

# Sesiones en PHP

1 - ¿Qué son las sesiones? .....	1
2 - ¿Dónde se usan? .....	2
3 - Un poco de historia .....	2
4 - ¿Para qué sirven las sesiones? .....	2
5 - ¿Dónde y cómo se guarda la información de la sesión en PHP? .....	2
6 - ¿Cómo sabe php cual es la sesión activa? .....	3
7 - ¿Por qué son seguras? .....	3
8 - ¿Todas las versiones de PHP tienen soporte para sesiones? .....	4
9 - ¿Hay que configurar algo para usarlas en PHP? .....	4
10 - ¿Qué variables hay y que hacen en el archivo PHP.INI? .....	4
11 - ¿Cuales són, qué hacen y cómo se utilizan las funciones de manejo de sesiones en PHP? .....	5
*12* - ¿Mi script puede acceder a las sesiones de otros clientes? .....	7
----- EJEMPLOS PRÁCTICOS ----- .....	8

---

## 1 - ¿Qué son las sesiones?

Se pueden definir como:

Un conjunto de variables almacenadas en el servidor; única por cada entidad que accede a la página.

o también como:

Estructura de datos almacenadas en el servidor, que ofrecen información del seguimiento del cliente, ésta información es individual para cada cliente.

-----

Generalmente las sesiones tienen 5 métodos principales:

- Abrir sesión
- Definir variable de sesión
- Definir el valor de una variable en sesión
- Obtener el valor de una variable en sesión
- Cerrar sesión

-----

Ejemplos:

FTP - El usuario (cliente) se conecta a una máquina remota (servidor), ofreciendo un usuario y un password, a partir de aquí se crea una sesión la cual contendrá valores tales como:

- ¿Está logado?
- Directorio actual
- Grupo al que pertenece
- Permisos
- IP
- ...

Los servidores tienen un tiempo de sesión, por ejemplo, si tiene 120 segundos de duración, al caducar el usuario será desconectado. Cada vez que se realiza una operación la sesión se refresca, y el tiempo de la sesión se pone a 0.

APLICACIÓN HTTP EN PHP - El usuario (cliente) se conecta al servidor, y se loga en un formulario, el cual contiene un nombre de usuario y un password, al ofrecer éstos, el script verifica los mismos, y obtiene los permisos que tiene este usuario; un ejemplo de datos almacenados en la sesión podrían ser:

- ¿Está logado? false-true
- Nombre de usuario
- E-Mail del usuario
- Permisos que tiene

- Poner noticias
- Añadir descargas
- Editar/Modificar una encuesta
- Subir fichero
- Poder poner mensajes en un foro

Una vez dentro el usuario podría ver, por ejemplo un mensaje que pusiese Hola, [NOMBRE], (Siempre y cuando ¿Está logado? valga true), y todas las operaciones deberían mirar los valores de ¿Estás logado? y de los permisos.

---

## 2 - ¿Dónde se usan?

Las sesiones son estructuras que se usan en aplicaciones cliente-servidor, tales como en ftp, telnet... aplicaciones web.

---

## 3 - Un poco de historia

- Las sesiones en PHP fueron incluidas en la versión 4 del mismo.
    - Antes de este acontecimiento, para usar sesiones en php tocaba realizar funciones de sesiones propias; había una librería que permitía usarlas 'LIBPHP';
- 

## 4 - ¿Para qué sirven las sesiones?

Las sesiones se utilizan como método para conservar ciertos datos a lo largo de los subsiguientes accesos.  
En las páginas webs permite construir aplicaciones más personalizadas e incrementar el atractivo de una página web.

---

## 5 - ¿Dónde y cómo se guarda la información de la sesión en PHP?

En php las sesiones son ficheros de texto guardados en disco, los cuales están en el directorio indicado en el fichero 'PHP.INI': en el apartado [Session] hay un atributo que es 'session.save\_path' y que contiene, como valor, el directorio utilizado para almacenar los datos de la sesión. Cada sesión tiene un nombre de fichero temporal semejante a: '<PATH SESIÓN>/sess\_<ID>' ... El formato de éste es:

- Nombre de la variable
- Tipo de la variable
- s - string (cadena)
- b - boolean (0/1)
- i - integer (entero)
- a - array (matriz)
- d - float ( )
- ...
- (entre otros)
- Tamaño del valor
- "Valor"

```
nombre|Tipo:Tamaño:"Valor";  
nombre|Tipo:Tamaño:"Valor";  
...
```

[[EJEMPLO]]

FICHERO: sess\_74e9c92b92e60b5a2a5c2de075135ff1

---

```
array|a:2:{s:8:"element1";s:4:"key1";s:8:"element2";s:4:"key2";}integer|i:100;string|s:15:"Cadena de texto";char|s:1:"a";bool|b:1;float|d:123.000323;
```

::::CADENA ESTRUCTURADA::::

```
array|a:2:{
s:8:"element1";
s:4:"key1";
s:8:"element2";
s:4:"key2";
}
integer|i:100;
string|s:15:"Cadena de texto";
char|s:1:"a";
bool|b:1;
float|d:123.000323;
```

FICHERO: sesiones.php

```
<?php
session_start ();
// Registra las variables de sesión si no están registradas
session_register ("array", "integer", "string", "char", "bool", "float");
$array = array ("element1" => "key1", "element2" => "key2");
$integer = (int)100;
$string = "Cadena de texto";
$char = "a";
$bool = true;
$float = (float)123.000323;
?>
```

---

## 6 - ¿Cómo sabe php cual es la sesión activa?

PHP utiliza dos métodos para identificar la sesión, mediante una cookie con un identificador único de la sesión, o bien mediante un parámetro: SID, el cual contiene éste identificador. Si el navegador no dispone de cookies éste último será el método utilizado por PHP; si está activado ([Session]->session.use\_trans\_sid),

PHP reescribirá los enlaces de tipo:

- a:href
- area:href
- frame:src
- input:src
- form:fakeentry

Estos valores son los que hay por defecto en el apartado [Session]-> del fichero PHP.INI, "url\_rewriter.tags"...

---

## 7 - ¿Por qué son seguras?

A diferencia de las cookies, las cuales guardan la información en el cliente, los valores, se guardan en el servidor, de forma inaccesible al cliente. Por ejemplo; puedes tener una variable en la sesión actual que sea 'logado', con el valor 'true', ésta información será accesible a todas las páginas PHP, y cada sesión tendrá sus propios valores, (cada explorador abierto en cada ordenador tiene su sesión propia (siempre y cuando se abra)), con ésto tienes la certeza de estar logado, ya que si fueran cookies el cliente podría manipular el valor y acceder a ellas, otra forma de saber si se está logado es meter en las cookies dos valores con el nombre del usuario y el password, y verificar cada vez estos valores,... a parte de que es mucho mas rápido obtenerlos de la sesión, esta información pasaría constantemente de una forma poco segura y además se quedaría guardado en el cliente, y otras personas podrían verlo.

## 8 - ¿Todas las versiones de PHP tienen soporte para sesiones?

NO. Las funciones de soporte de sesiones aparecieron a partir de PHP 4, sin embargo, para las personas que utilicen PHP 3, pueden recurrir a librerías hechas por ellos o a la librería PHPLIB.

---

## 9 - ¿Hay que configurar algo para usarlas en PHP?

No, a partir de la versión 4.0 están disponibles, al menos que en la configuración se defina lo contrario. Ver apartado 10.

---

## 10 - ¿Qué variables hay y que hacen en el archivo PHP.INI?

En el fichero .INI hay un apartado dedicado a las sesiones; en el cual se definen ciertos valores que determinarán el comportamiento de tales.

- session.save\_handler
  - Indica el handler que usará PHP para guardar las variables de sesión
  - defecto: files
  - MAS INFO: función ini\_set ();
- session.save\_path
  - Indica el path dónde se guardan los ficheros de las sesiones
  - defecto: /tmp
- session.use\_cookies
  - Indica si se usarán las cookies para almacenar el SID
  - defecto: 1
- session.name
  - Indica el nombre de la cookie que indicará el SID
  - defecto: PHPSESSID
- session.auto\_start
  - Indica si la sesión se inicializa al empezar el script automáticamente
  - defecto: 0
- session.cookie\_lifetime
  - Indica el tiempo de vida de la cookie (si aplicable)
  - Si 0, caduca al cerrar el explorador.
  - defecto: 0
- session.cookie\_path
  - Indica el path de la cookie para que sea válida
  - defecto: /
- session.cookie\_domain
  - Indica el dominio de la cookie para que sea válida
  - defecto:
- session.serialize\_handler
  - Indica el handler usado para serializar la sesión
  - defecto: php
- session.gc\_probability
  - Indica el número de veces de cada 100 que se ejecuta el recolector de basura de las sesiones. Éste proceso borra los ficheros de las sesiones que hallan caducado ya.
  - defecto: 1 cada 100 (1%)
- session.gc\_maxlifetime
  - Indica el número de segundos que deben haber pasado desde la última actualización para que se considere basura.
  - defecto: session.gc\_maxlifetime
- session.referer\_check
  - Indica la cadena que debe contener la variable \$HTTP\_REFERER (variable que contiene la url que ha referenciado a ésta página (No todos los navegadores envían esta información))
  - defecto:
- session.entropy\_length
  - Indica el número de bytes que se leerán del fichero
- session.entropy\_file
  - Indica como crear el fichero
  - defecto: /dev/urandom
- session.cache\_limiter

- Determina el tipo de caché HTTP
- \* nocache
- \* private
- \* public
- defecto: nocache
- session.cache\_expire
- Indica el tiempo en minutos para que caduque el documento
- defecto: 180
- session.use\_trans\_sid
- Indica si se utilizará el transid, que modificará automáticamente los enlaces por GET.
- defecto: 1
- url\_rewriter.tags
- Se relaciona con session.use\_trans\_sid, e indica los tags y los parámetros los cuales reescribirán sus urls.
- defecto: "a=href,area=href,frame=src,input=src,form=fakeentry"

## 11 - ¿Cuales són, qué hacen y cómo se utilizan las funciones de manejo de sesiones en PHP?

### ----- **FUNCIONES** -----

- session\_id ( [ CADENA ] ) devuelve CADENA
- Obtiene/Fija el valor del 'id' de la sesión actual.
  
- session\_cache\_expire ( [ ENTERO ] ) devuelve ENTERO (minutos)
- Obtiene/Fija el valor en minutos que tienen que pasar para que la sesión actual caduque.
- \* La variable (PHP.INI->[Session]->session.cache\_expire) contiene el valor por defecto indicado en minutos.
  
- session\_cache\_limiter ( [ CADENA ] ) devuelve CADENA
- Obtiene/Fija el valor de una variable que indica el nivel de caché que habrá en el cliente:
- nocache (valor por defecto)
- public
- private (puede confundir algunos navegadores)
- private\_no\_expire
  
- session\_decode (CADENA) devuelve BOOL
- Decodifica la CADENA en el formato de las sesiones de PHP, y introduce los valores en la sesión activa.
- Al finalizar:
- Devuelve 'true' si la operación se ha realizado correctamente
- Devuelve 'false' si la ha habido algún tipo de error y por consiguiente no se ha podido introducir los valores en la sesión activa.
  
- session\_destroy (nada)
- Destruye las variables de sesión con sus valores
- Al finalizar:
- Devuelve 'true' si la operación se ha realizado correctamente
- Devuelve 'false' si la ha habido algún tipo de error y por consiguiente no se han podido borrar los valores de la sesión
  
- session\_encode (nada) devuelve CADENA
- Devuelve una cadena con la sesión actual codificada, en el formato indicado en el apartado '¿Dónde y cómo se guarda la información de la sesión?'
- \* Esta cadena puede utilizarse para almacenar sesiones y mas tarde reusarlas con la función 'session\_decode'

- `session_is_registered (CADENA)` devuelve `BOOL`
- Devuelve `TRUE` si existe una variable (en la sesión actual) con el nombre `CADENA`
- Devuelve `FALSE` en caso contrario

- `session_name ( [ CADENA ] )` devuelve `CADENA`
- Si se pasa como parámetro una cadena, modifica el nombre de de la sesión actual se cambia a este valor; si se define un valor debe de usarse antes de `session_start ( )` y `session_register ( )`, y hazelo en caad petición.

- `session_readonly (nada)` devuelve nada
- Inicia la sesión, pero con la particularidad de que al finalizar el script las variables de sesión se quedan como estaban, és decir, no se guardan los nuevos valores de la sesión.

- `session_register ( MIXTO [, MIXTO ...])` devuelve `BOOLEANO`
- Registra una variable de sesión como tal, esta variable se puede acceder globalmente, por ello para acceder desde una función se debe de usar el comando global `$variable`;

Atención:

-----

Si usas las arrays `$_SESSION` o `$HTTP_SESSION_VARS`, no uses las funciones `session_register ( )`, `session_is_registered ( )` y `session_unregister ( )`. Estas funciones devolveran `VERDADERO` cuando todas las variables se hallan registrado en la sesión. Si `session_start ( )` no ha sido llamada antes de usar esta función, se puede hacer una llamada a esta función sin parámetros para hacerlo. Puedes crear variables de sesión (tan bien) poniendo valores en las matrices `$_SESSION` (a partir de la versión 4.1) o en `$HTTP_SESSION_VARS` (antes de ésta versión)

- `string session_save_path ( [ CADENA ] )` devuelve `CADENA`
- Esta función permite obtener o definir el path del directorio dónde deben guardarse los ficheros de las sesiones.

- `session_set_cookie_params ( INT [, path CADENA [, dominio CADENA [, n/seguro BOOL]]])`
- Cambia los parámetros de configuración para usar las cookies, éstos están definidos por defecto en el fichero `PHP.INI`

- `session_set_save_handler ( string abrir, string cerrar, string leer, string escribir, string destruir, string gc)`

- Más info.: `'http://www.php.net/manual/es/function.session-set-save-handler.php'`

- Define las funciones de usuario respecto a la sesión:
- Abrir (`$save_path`, `$session_name`) - (al abrir la sesión)
- Cerrar ( ) - (al cerrar la sesión)
- Leer (`$id`) - (???)
- Escribir (`$id`, `$sess_data`) - (???)
- Destruir (`$id`) - (???)
- rb (`$maxfiletime`) - (Recolector de basura)

Ver las variables de `PHP.INI`

¿Cómo funciona y qué es el recolector de basura de las sesiones?

-----

- Esta función indica que funciones del usuario se deben de ejecutar en determinadas ocasiones.

Ej:

```
- function FCabrir ($path, $name) {  
}  
session_set_save_handler ("FCabrir" ...);
```

- `session_start (nada)` -> No devuelve nada

- Inicializa la sesión:
- Crea una nueva sesión

o bien

- Continúa la anterior basandose en el session id pasado por GET o mediante una cookie). Carga las variables del fichero como globales y permite empezar a usar las funciones principales de sesión.

- Si estás usando sesiones basadas en las cookies, debes llamar a la función antes de que haya ninguna salida al navegador.

- Más info.: <http://www.php.net/manual/es/function.session-start.php>  
- `session_start()` registrará un manejador de salida interno para la reescritura de las URL's si `trans-sid` está activado. Si un usuario utiliza `ob_gzhandler` o algo como `ob_start()`, el orden del manejador de salida es importante para que la salida sea la adecuada. Por ejemplo, el usuario debe registrar `ob_gzhandler` antes de iniciar la sesión.

Nota: Se recomienda utilizar `zlib.output_compression` en lugar de `ob_gzhandler`

- `session_unregister ( CADENA )` devuelve BOOLEANO  
- Quita de la sesión y destruye la variable indicada como parámetro.  
- Al finalizar  
- Devuelve true si se hizo correctamente  
- Devuelve false si no se pudo eliminar

- Más info.: <http://www.php.net/manual/es/function.session-unregister.php>  
Nota: Si utiliza `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o inferior), use `unset()` para eliminar una variable de la sesión actual.

- `session_unset (nada)` devuelve nada  
- Destruye todas las variables de la sesión

- Más info.: <http://www.php.net/manual/es/function.session-unset.php>  
Nota: Si utiliza `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o inferior), use `unset()` en su lugar para desregistrar una variable de la sesión. p.ej. `$_SESSION = array();`

- `session_write_close (nada)` devuelve nada  
- Finaliza la sesión actual y guarda sus datos

- Más info.: <http://www.php.net/manual/es/function.session-write-close.php>  
Los datos de la sesión se suelen guardar tras finalizar la ejecución de su script sin necesidad de llamar a `session_write_close()`, pero como los datos de la sesión están bloqueados para evitar escrituras concurrentes, sólo un script puede trabajar con una sesión a la vez. Cuando se usan framesets junto con sesiones, podrá comprobar que los frames se cargan uno a uno debido a este bloqueo. Puede reducir el tiempo necesario para cargar los frames finalizando la sesión tan pronto como haya terminado los cambios a las variables de la sesión.

---

## **\*12\* - ¿Mi script puede acceder a las sesiones de otros clientes?**

Con las funciones que ofrece el sistema de sesiones de PHP no...  
Sin embargo hay 1 forma...  
Que es obteniendo el directorio de las sesiones y editando el fichero de otra sesión; (éste método no debería ser utilizado).

Las sesiones fueron creadas para la almacenación persistente de información relevante de los usuarios que accedan a un sistema, esta información es individual y no requiere ver otras sesiones.

## EJEMPLOS PRÁCTICOS

### Formulario de varios pasos

Si quieres dividir un formulario en varias partes tienes varias opciones:

- Coger los datos que quieras que 'persistan' y crear el siguiente formulario usando elementos 'hidden' para almacenarlos hasta que se acabe el proceso y meterlo todo junto en una base de datos por ejemplo.
- Como es evidente esta forma no es recomendable, por diversos motivos.
- Meter la información en la base de datos por ejemplo, y así tener toda la información al final en la base de datos. Esto está bien, pero ... ¿Qué pasa si no se acaba de completar el formulario?
- Por último usando las sesiones :), vas metiendo la información en la sesión, y al acabar el proceso de registro, introduces los datos almacenados en la sesión en la base de datos por ejemplo.

```
form.php
.....

<?php
// Inicia la sesión
// Si existe se abre y se cargan los datos de la sesión,
// en caso contrario se crea.

session_start ();

while (list($key, $val) = @each($_HTTP_POST_VARS)) {
if ($key != "step" && $key != "Astep") {
session_register ("$key");
$GLOBALS[$key] = $val;
}
}

if ($step == -1) session_unset ();
?>
<html><head></head>
<body>
<table>
<form name="Mform" action="ses.php" method="post">
<?php
if (!isset ($step) || $step < 1 || $step > 3) $step = 1;

switch ($Astep) {
case 1:
if (!vNick ($name)) $step = 1;
if (!vPassword ($password)) $step = 1;
if (!vEmail ($email)) $step = 1;
if ($password != $password2) $step = 1;
break;
case 2:
break;
case 3:
break;
}
if ($Astep == $step) FRMparagraph ("Verifique que los campos están correctamente escritos");

switch ($step) {
case 1:
FRMtexto ("Nombre", "name", $name);
FRMpassword ("Password", "password", $password);
FRMpassword ("Repita Password", "password2", $password2);
FRMtexto ("E-Mail", "email", $email);
```

```

FRMgoto (2);
?>
<?php
break;
case 2:
$years = array ();
for ($n = 1900; $n <= ((int)date ("Y")); $n++) array_push ($years, $n);
$years = array_reverse ($years);
FRMselect ("Año", "year", $years, $year);
$months = array (
"Enero",
"Febrero",
"Marzo",
"Abril",
"Mayo",
"Junio",
"Julio",
"Agosto",
"Septiembre",
"Octubre",
"Noviembre",
"Diciembre");
FRMselect ("Mes", "month", $months, $month);
$days = array ();
for ($n = 1; $n <= 31; $n++) array_push ($days, $n);
FRMselect ("Día", "day", $days, $year);
FRMgoto (3);
?>
<?php
break;
case 3:
echo "Fin del formulario, gracias por su colaboración...";
// Ahora se meterían los datos, por ejemplo en una base de datos y
// se ofrecería un link para aceptar los datos
?>
<?php
default;
?>
<?php
}

```

```

// Reglas:
// tener una @ en la cadena
// tener al- un . en la cadena
// usar sólo los caracteres: 'A-Z', 'a-z', '@', '.' y '-'
function vEmail ($email) {
if (substr_count ($email, "@") == 1 &&
substr_count ($email, ".") > 0 &&
ereg ("^([A-Za-z@.-]{1,})$", $email)) {
return true;
} else {
return false;
}
}

```

```

// Reglas:
// letras y números
// nº caracteres ... < o = a 16
function vNick ($nick) {
if (strlen ($nick) <= 16 &&
ereg ("^([A-Za-z0-9]{1,})$", $nick)) {
return true;
} else {
return false;
}
}

```

```

// Reglas:
// nº caracteres ... > o = a 6 && < o = 32
//
function vPassword ($password) {
if (strlen ($password) >= 6 &&
strlen ($password) <= 32) {
return true;
} else {
return false;
}
}

```

```

}
}

function FRMgoto ($ses) {
global $step, $Astep;
echo "<tr><td colspan=2>";
echo "<input type=hidden name=step value=$ses/>n";
echo "<input type=hidden name=Astep value=$step/>n";
if ($step > 1) {
echo "<input type=button onClick=Mform.Astep.value= . ($step) . ";Mform.step.value= . ($step - 1) . ";Mform.submit();" value=Anterior &lt;&lt;"/>n";
}
echo "<input type=submit value=Siguiete &gt;&gt;"/>n";
echo "<input type=button onClick=Mform.step.value=-1;Mform.Astep.value=0;Mform.submit();" value=BORRAR"/>n";
echo "</td></tr>n";
}

function FRMtexto ($text, $name, $value) {
echo "<tr>";
echo "<td>$text:</td>";
echo "<td><input type=text name=$name value=$value/></td>";
echo "</tr>n";
}

function FRMparagraph ($text) {
echo "<tr><td colspan=2>$text</td></tr>n";
}

function FRMpassword ($text, $name, $value) {
echo "<tr>";
echo "<td>$text:</td>";
echo "<td><input type=password name=$name value=$value/></td>";
echo "</tr>n";
}

function FRMselect ($text, $name, $array, $value) {
echo "<tr>";
echo "<td>$text:</td>";
echo "<td><select name=$name>n";
for ($n = 0; $n < sizeof ($array); $n++) {
echo $value . ", " . $array[$n] . "<br/>n";
if ($value == $array[$n]) {
echo "<option name=$n selected>" . $array[$n] . "</option>n";
} else {
echo "<option name=$n>" . $array[$n] . "</option>n";
}
}
echo "</select>";
echo "</td></tr>n";
}
?>
</form>
</table>
</body>

```

-----  
Más ayuda en <http://php-hispano.net>