Índice

Sesiones

Introducción

Una de las limitaciones de las páginas web es que cada página web es un documento independiente. Eso hace que dos programas PHP no puedan, en principio, compartir información.

En principio, la única manera de enviar información de una página a otra es a través de un formulario y se trata de la información que ha introducido el usuario en ese formulario. Como mucho podemos enviar información adicional mediante controles ocultos, o podemos "imitar" a un formulario redirigiendo a otra página enviando información en la dirección (como se comenta con la función header), pero esto no puede considerarse compartir información, puesto que la página que recibe la información no puede saber si la información ha sido manipulada por el camino ni quién se la envía.

Por compartir información nos estamos refiriendo a utilizar variables comunes en programas distintos, sin que la información salga del servidor.

El ejemplo siguiente muestra dos páginas: en la primera se crea una variable, que se puede utilizar en esa misma página. La segunda página, que se encuentra en el mismo servidor, no tiene acceso a la variable creada en el primer programa y por eso se produce un aviso al ejecutarla.

```
<?php
  $nombre = "Antonio";
  print "<p>El nombre es $nombre</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

```
<?php
  print "<p>El nombre es $nombre</p>";
?>
```

```
Notice: Undefined variable: nombre in ejemplo.php on line 2

El nombre es
```

PHP puede superar esta limitación mediante las sesiones. Las sesiones permiten que páginas distintas puedan acceder a una variable común, la matriz $_SESSION.

El ejemplo siguiente muestra dos páginas: en la primera se crea la variable en la matriz $_SESSION, que se puede utilizar en esa misma página. La segunda página, que se

encuentra en el mismo servidor, tiene acceso a la variable creada en el primer programa (si se ha ejecutado antes el primer programa, obviamente).

```
<?php
  session_start();
  $_SESSION["nombre"] = "Antonio";
  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

```
<?php
  session_start();
  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

Las sesiones no deben confundirse con las cookies. Las cookies es un método que permite guardar información en el ordenador del cliente para recuperarla en el futuro, mientras que en las sesiones la información se mantiene en el servidor hasta que se cierra la sesión (por intervención del usuario o por tiempo).

El trabajo con sesiones tiene tres partes:

- Creación o apertura de la sesión
  Cuando alguna página crea una sesión utilizando la función correspondiente, el servidor asocia al navegador del usuario un identificador de usuario único. El identificador se guarda en el usuario en forma de cookie o, si el navegador del usuario no permite la creación de cookies, añadiendo el identificador en la dirección de la página.
- Utilización de la sesión
  Si ya se ha creado la sesión, las páginas solicitadas por el mismo navegador pueden guardar y recuperar información en el servidor, información que se asocia al identificador de usuario, por lo que no es accesible a otros usuarios. La información se conserva hasta que el usuario o el servidor destruyan la sesión.
- Destrucción o cierre de la sesión
  Tanto el usuario como el servidor pueden cerrar la sesión. El usuario puede destruir la sesión cerrando el navegador. El servidor puede destruir la sesión cuando alguna página utilice la función correspondiente o al cabo de un tiempo determinado (definido mediante la función **session_set_cookie_params()**).

Crear la sesión

En PHP, 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();
  $_SESSION["nombre"] = "Antonio";
  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

```
<?php
  session_start();
  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

Utilizar la sesión

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.
La matriz $_SESSION es una matriz asociativa en la que se pueden definir valores como en cualquier otra matriz. La diferencia es que $_SESSION es accesible desde páginas diferentes (siempre que esas páginas tengan asociada la misma sesión), manteniéndose los valores de una página a otra.
El ejemplo siguiente muestra dos páginas. La primera página guarda información en $_SESSION y la segunda la utiliza.

```
<?php
  session_start();
  $_SESSION["nombre"] = "Antonio";
  print "<p>El nombre es $_SESSION["nombre"]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

```
<?php
  sesion_start();
  print "<p>El nombre es $_SESSION["nombre"]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

Los nombres de los primeros índices de la matriz $_SESSION tienen que cumplir las mismas reglas que los nombres de las variables, es decir, que el primer carácter debe ser una letra o un guión bajo (_). En particular, no deben ser números ni contener caracteres no alfanuméricos.

```
<?php
  session_start();
  $_SESSION[] = "Antonio";
  print "<p>Se ha guardado su nombre.</p>";
  print "<pre>";
  print_r($_SESSION);
  print "</pre>";
?>
```

```
<p>Se ha guardado su nombre</p>

Notice:  Unknown: Skipping numeric key 0 in Unknown on line 0
```

```
<?php
  session_start();
  $_SESSION["nombres"][] = "Antonio";
  print "<p>Se ha guardado su nombre.</p>";
  print "<pre>";
  print_r($_SESSION);
  print "</pre>";
?>
```

```
<p>Se ha guardado su nombre</p>

Array
(
    [nombres] => Array
        (
            [0] => Antonio
        )
)
```

Cerrar la sesión

Cerrar una sesión es destruir la matriz $_SESSION y el identificador de la sesión.
Las sesiones se pueden cerrar de varias maneras:

- El usuario puede cerrar la sesión simplemente cerrando el navegador (no basta con cerrar las pestañas).

- Un programa puede cerrar la sesión mediante la función **session_destroy()**.
- En general, las cookies tienen una duración establecida en la directiva **session.cookie_lifetime** (y el servidor puede borrar la información cuando ha pasado el tiempo indicado en segundos en la directiva **session.gc_maxlifetime**), pero la duración de una sesión en particular puede establecerse en el momento de su creación mediante la función **session_set_cookie_params()** (tiempo que se puede modificar posteriormente).

Cuando se destruye una sesión, el programa que ha destruido la sesión sigue teniendo acceso a los valores de $_SESSION creados antes de la destrucción de la sesión, pero las páginas siguientes no. Si se ejecuta el primero de los ejemplos siguientes y después el segundo, se obtienen los resultados indicados:

```php
<?php
  session_start();
  $_SESSION["nombre"] = "Antonio";
  session_destroy();
  if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>";
  } else {
    print "<p>No sé su nombre.</p>";
  }
?>
```

```
<p>Su nombre es Antonio</p>
```

```php
<?php
  session_start();

  if (isset($_SESSION["nombre"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>";
  } else {
    print "<p>No sé su nombre.</p>";
  }
?>
```

```
<p>No sé su nombre.</p>
```

Nombre de sesión

En principio, cuando el navegador se conecta a un servidor, la sesión es única, es decir, todas las páginas del mismo dominio compartirán la misma matriz $_SESSION. La función **session_name()** permite establecer un nombre de sesión específico, de manera que todas las páginas que declaren el mismo nombre de sesión accederán a la misma matriz $_SESSION y las que tengan nombres de sesión distintos accederán a matrices $_SESSION diferentes.

En el ejemplo siguiente, los dos primeros programas crean la misma variable en $_SESSION, pero como se ha utilizado la función **session_name()** con nombres diferentes, realmente lo hacen en matrices $_SESSION distintas.

```
<?php
  session_name("ejemplo1");
  session_start();
  $_SESSION["nombre"] = "Antonio";

  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

```
<?php
  session_name("ejemplo2");
  session_start();
  $_SESSION["nombre"] = "Juan";

  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Juan</p>
```

Los dos programas siguientes acceden después a las matrices $_SESSION, pero cada uno accede a la matriz correspondiente al nombre de sesión.

```
<?php
  session_name("ejemplo1");
  session_start();

  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Antonio</p>
```

```
<?php
  session_name("ejemplo2");
  session_start();

  print "<p>El nombre es $_SESSION[nombre]</p>";
?>
```

```
<p>El nombre es Juan</p>
```

El nombre de sesión distingue entre minúsculas y mayúsculas, es decir, dos sesiones con el mismo nombre, pero uno en minúsculas y otro en mayúsculas, son dos sesiones distintas.
El nombre de la sesión no puede contener únicamente números, ni tampoco puede contener los caracteres espacio ( ), punto (.), ampersand (&), más (+), corchete izquierdo ([) ni almohadilla (#).

Borrar elementos de la sesión

Los valores de $_SESSION se borran como en cualquier otra matriz mediante la función **unset()**.

```
<?php
  session_start();
  $_SESSION["nombre"] = "Antonio";

  print "<p>Su nombre es $_SESSION[nombre]</p>";

  unset($_SESSION["nombre"]);

  if (isset($_SESSION["nombres"])) {
    print "<p>Su nombre es $_SESSION[nombre].</p>";
  } else {
    print "<p>No sé su nombre.</p>";
  }
?>
```

```
<p>Su nombre es Antonio.</p>
<p>No sé su nombre.</p>
```

Para borrar todos los valores de $_SESSION se pueden borrar uno a uno o utilizar la función **session_unset()**, pero normalmente no se debe utilizar **unset()**:

- **unset($_SESSION)** impide que el resto de la página escriba o lea valores en $_SESSION, pero la sesión conserva los valores, por lo que otras páginas seguirán viendo esos valores.

```
<?php
  session_start();
  $_SESSION["nombre"] = "Antonio";
  $_SESSION["apellidos"] = "López";
  $a = &$_SESSION;
  unset($_SESSION);
  $a["apellidos"] = "García"; // $a hacer referencia a $_SESSION
original
```

```
  $_SESSION["apellidos"] = "Pérez";
  header("Location:pagina.php");
?>
```

pagina.php

```
<?php
  session_start();
  $_SESSION["saludo"] = "Hola";
  print "<pre>";
  print_r($_SESSION);
  print "</pre>";
?>
```

```
<pre>
Array
(
    [nombre] => Antonio
    [apellidos] => García
    [saludo] => Hola
)
</pre>
```

- La función **session_unset()** borra todos los valores pero permite que el resto de la página (y otras páginas) escriba o lea valores en $_SESSION.

```
<?php
  session_start();
  $_SESSION["nombre"] = "Antonio";
  session_unset();
  $_SESSION["apellidos"] = "López";

  header("Location:pagina.php");
?>
```

pagina.php

```
<?php
  session_start();
  $_SESSION["saludo"] = "Hola";
  print "<pre>";
  print_r($_SESSION);
  print "</pre>";
?>
```

```
<pre>
Array
```

```
(
    [apellidos] => López
    [saludo] => Hola
)
</pre>
```

Directiva session.save_handler

Para utilizar sesiones mediante el mecanismo propio de PHP (es decir, sin necesidad de crear funciones propias), la directiva session.save_handler del archivo de configuración php.ini debe tener el valor files.

```
session.save_handler = files      ; Valor recomendado en este
curso
```

Esta configuración es la más habitual, pero algunos gestores de contenidos (CMS) tienen sus propias funciones de gestión de sesiones y requieren que esta directiva tome el valor user.

```
session.save_handler = user    ; Valor definido en algunos
servidores
```

Si se ha necesitado modificar php.ini, pero queremos ejecutar otros programas que no incluyen sus propias funciones de gestión de sesiones, se puede incluir en las páginas PHP la función ini_set() antes de abrir la sesión. Ese cambio sólo afectará a la página que incluya la llamada a la función.

```
<?php
  ini_set("session.save_handler", "files"); // Necesario
únicamente cuando session.save_handler = user en php.ini
  session_start();
?>
```

Seguridad en las sesiones

A pesar de su simplicidad, hay en determinadas ocasiones que el uso de sesiones puede salir mal. Vamos a ver algunas técnicas sencillas para evitar riesgos con las sesiones:

- **Sesión Time-Outs**. Establecer un tiempo de vida a las sesiones es algo importante para lidiar con las sesiones de usuarios en aplicaciones. Si un usuario inicia sesión en un lugar y se olvida de cerrarla, otros pueden continuar con la misma, por eso es mejor establecer un tiempo máximo de sesión:

```
<?php
  session_start();

  // Establecer tiempo de vida de la sesión en segundos
  $inactividad = 60;

  // Comprobar si $_SESSION["timeout"] está establecida
  if(isset($_SESSION["timeout"])){
  // Calcular el tiempo de vida de la sesión (TTL = Time To Live)
    $sessionTTL = time() - $_SESSION["timeout"];
    if($sessionTTL > $inactividad){
      session_destroy();
      header("Location: fin.php");
    }
  }

  // La siguiente clave se crea cuando se inicia sesión (llamada
a la página)
  $_SESSION["timeout"] = time();

?>
```

El código asegura que si no hay ninguna actividad en 1 minuto, cualquier request en adelante redirigirá a la página de fin.

- **Regenerar el Session ID**. La función **session_regenerate_id()** crea un nuevo ID único para representar la sesión actual del usuario. Esto se debe realizar cuando se realizan acciones importantes como logearse o modificar los datos del usuario. Darle a las sesiones un nuevo ID reduce la probabilidad de ataques **session hijacking**:

```
<?php
  session_start();

  if($_POST["usuario"] = "admin"
     && $_POST["password"] == sha1($password)){
        $_SESSION["autorizado"] = true;
        session_regenerate_id();
  }
?>
```

- **Destruir la sesión.** Como ya se ha mencionado antes, es importante utilizar **session_destroy()** cuando ya no se vaya a hacer uso de la sesión.
- **Utilizar almacenamiento permanente**. Utilizar una base de datos para almacenar los datos que se cree que serán persistentes. Dejarlos en la sesión por demasiado tiempo abre de nuevo puertas a ataques. Depende del desarrollador decidir qué datos serán almacenados o se mantendrán en $_SESSION.