MySQL y Sesiones en PHP

Área de Ingeniería Telemática

Contenido

- **Repaso de SQL**
- PHP y MySQL
- Sesiones en PHP

Repaso SQL: tipo de datos

- Numéricos
  - Standard:
    - INTEGER o INT, SMALLINT, DECIMAL o DEC, NUMERIC
    - FLOAT, REAL, DOUBLE PRECISION
    - BIT
  - Extras:
    - TINYINT, MEDIUMINT, BIGINT
- Fecha y hora
  - DATETIME, DATE, TIMESTAMP, TIME, YEAR
  - Las fechas siempre deben darse en el orden **año-mes-día**

Repaso SQL: tipo de datos

Tipo de Columna	"Cero" Valor
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	00000000000000
TIME	'00:00:00'
YEAR	0000

- Cadena de caracteres
  - CHAR y VARCHAR
    - Char: longitud fija (0-255). Asigna espacios si la cadena es más corta
    - Varchar: longitud variable (0-255 v5.0.3-, 0-65535 MySQLv5.0.3+)

Repaso SQL: tipo de datos

- BINARY y VARBINARY
  - Similares a CHAR y VARCHAR
  - Contienen cadenas de caracteres binarias (cadenas de bytes) en lugar de cadenas de carácteres no binarias (cadenas de caracteres)
  - Principalmente se usan para almacenar ficheros
- BLOB y TEXT
  - No pueden tener valores DEFAULT
  - Su tamaño está prefijado
  - TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
  - TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT
  - BLOB: cadenas de carácteres binarias
  - TEXT: cadenas de carácteres no binarias

Repaso SQL: tipo de datos

- ENUM
  - Objeto de cadenas de caracteres con un valor elegido de una lista de valores permitidos que se enumeran explícitamente en la especificación de columna en tiempo de creación de la tabla
  - Cada valor de la enumeración tiene un índice
  - ENUM('uno', 'dos', 'tres') => 'uno' 'dos' 'tres'
- SET
  - objeto de cadenas de caracteres que tiene cero o más valores, cada uno de ellos debe elegirse de una lista de valores posibles especificada cuando se crea la tabla.
  - Los miembros de SET no pueden contener comas
  - Puede tener un máximo de 64 miembros distintos
  - SET('one', 'two') NOT NULL => '' 'one' 'two' 'one,two'

Repaso SQL: SELECT

- SELECT * FROM tabla
- SELECT nombre_campo as nuevo_nombre FROM tabla
- SELECT * FROM tabla WHERE nombre_campo='valor_campo'
- SELECT * FROM tabla WHERE nombre_campo='valor_campo' AND/OR nombre_campo='valor_campo'
- SELECT * FROM tabla GROUP BY nombre_campo
- SELECT DISTINCT nombre_campo FROM tabla
- SELECT * FROM tabla nombre_campo LIKE '%valor_campo?'
- SELECT * FROM tabla ORDER BY nombre_campo ASC/DESC

Repaso SQL: INSERT INTO

- INSERT INTO tabla (campo_1, campo_2, ...) VALUES ('valor_1', 'valor_2', ...)
- Recomendación: siempre poner los nombres de los campos

Repaso SQL: UPDATE

- UPDATE tabla SET campo_1='valor_1', campo_2='valor_2', ...
- UPDATE tabla SET campo_1='valor_1', campo_2='valor_2', ... WHERE campo_x='valor_x'

Repaso SQL: DELETE

- DELETE FROM tabla WHERE
- DELETE FROM tabla WHERE nombre_campo =‘valor_campo’

Repaso SQL: Otros

- TRUNCATE TABLE tabla => reinicia tabla
- SELECT * FROM tabla_1 WHERE nombre_campo=(SELECT valor FROM tabla_2 WHERE nombre_campo_2='valor_campo_2')

- Referencias:
  - http://www.w3schools.com/sql/default.asp
  - http://dev.mysql.com/doc/refman/5.0/es/index.html

Contenido

- Repaso de SQL
- **PHP y MySQL**
- Sesiones en PHP

PHP y MySQL

- Veamos algunas funciones para conectarse y realizar queries a un servidor de MySQL:
  - `mysql_connect()`
  - `mysql_list_dbs(), mysql_select_db()`
  - `mysql_select_db()`
  - `mysql_query()`
  - `mysql_fetch_array()`
  - `mysql_num_rows(), mysql_insert_id()`
  - `mysql_error(), mysql_errno()`

mysql_connect()

- Establecer una conexión con un servidor MySQL

```
resource mysql_connect ( [string server [, string username
   [, string password [, bool new_link [, int
   client_flags]]]]])
```

- Devuelve un recurso que identifica a esa conexión si se lleva a cabo con éxito, si no devuelve FALSE
- Ejemplo:

```
$link = mysql_connect("localhost", "mysql_user",
                      "mysql_password");
```

- Para cerrar esa conexión se usa `mysql_close()`, aunque no es necesario puesto que se cerrará al terminar el script de PHP. Es recomendable cerrarla.

mysql_list_dbs()

- Lista las bases de datos disponibles en el servidor (si se tiene permiso)

```
resource mysql_list_dbs ( [resource link_identifier])
```

- El argumento es el recurso que identifica a la conexión con la base de datos (`$link` en el ejemplo anterior)
- El resultado se puede recorrer como cualquier resultado de un query

mysql_select_db()

- Selecciona una base de datos para todos los queries siguientes

```
bool mysql_select_db ( string database_name [, resource link_identifier])
```

- Si se le pasa el segundo argumento (la conexión a la BD), pasa a estar activa la base de datos seleccionada en el resource dado
- Sin segundo argumento, activa esa base de datos en la conexión más reciente al servidor

mysql_query()

- Envía una query al servidor

```
resource mysql_query ( string query [, resource
                    link_identifier])
```

- Envía una query (primer argumento) al servidor al que hace referencia el segundo argumento, a la base de datos que tenga activa
- Si no hay segundo argumento se emplea la última conexión con base de datos que se haya creado
- Para `SELECT`, `SHOW`, `EXPLAIN` y `DESCRIBE` devuelve un identificador de recurso o FALSE
- Para otras sentencias SQL devuelve TRUE o FALSE

mysql_fetch_array()

- Extrae una fila del resultado de una query

```
array mysql_fetch_array ( resource result [, int result_type])
```

- El primer argumento es el resultado de la query
- El segundo argumento nos permite especificar cómo queremos que nos devuelva la fila:
  - `MYSQL_ASSOC` para que sea un array asociativo cuyas claves son los nombres de las columnas
  - `MYSQL_NUM` para que sea un array indexado por el número de columna
  - `MYSQL_BOTH` para que tenga ambas claves (nombres e índice)
- Cada vez que se llama a la función devuelve la siguiente fila del resultado
- Devuelve FALSE cuando no quedan más filas

mysql_num_rows()

- Devuelve el número de filas que tiene el resultado de una query de tipo SELECT

```
int mysql_num_rows ( resource result)
```

- El argumento es el resultado de la query

mysql_insert_id()

- Si la última operación de tipo INSERT fue en una tabla con una columna con el atributo AUTO_INCREMENT devuelve el ID nuevo que se le asignó

```
int mysql_insert_id ( [resource link_identifier])
```

- El argumento es la conexión con la base de datos y si no se especifica se empleará el último que se haya creado.

mysql_error()

- Devuelve el texto asociado al error que produjo la última función de MySQL

```
string mysql_error ( [resource link_identifier])
```

- Si no se produjo un error devuelve la cadena vacía.

mysql_errno()

- Devuelve el código numérico del error que produjo la última función de MySQL

```
int mysql_errno ( [resource link_identifier])
```

PHP+MySQL: recomendaciones

- Crear un fichero "config.php" donde guardar los parámetros de conexión MySQL
- Mediante "include" usar ese fichero en estos los scripts => actualizar parámetros MySQL tan sencillo como modificar "config.php"
- Opcional:
  - Crear en "config.php" una función que devuelva una conexión MySQL a la BBDD => ahorrar tener que escribir las mismas líneas en todos los scripts

Lectura sencilla de MySQL

```
<?php
session_start();
include("inicio.php");
?>
<html><head><title>Uso PHP, MySQL y HTML</title></head>
<body>
<table border="0">
  <tr><td>ID</td><td>Usuario</td></tr>
<?php
$query = "SELECT * FROM Usuarios";
$result = mysql_query($query, $link);
while($row = mysql_fetch_array($result))
  {
  print "<tr><td>".$row['id']."</td><td>".$row['usuario']."</td></tr>\n";
  }
?>
</table>
</body>
<?php
include("fin.php");
?>
```

Contenido

- Repaso de SQL
- PHP y MySQL
- **Sesiones en PHP**
  - Estado de la conexión con el navegador
  - Cookies
  - Sesiones

Estado de la conexión con el navegador

- Cuando el usuario solicita una página que es en verdad un PHP éste empieza a ejecutarse, manteniendo la conexión establecida para poder mandar el resultado del script
- Puede que el usuario aborte la conexión (botón STOP en el navegador)
- Para reconocer esta circunstancia un script PHP puede encontrarse en diferentes estados:
  - NORMAL: Mientras el script se ejecuta con normalidad como se ha descrito
  - ABORTED: Si el usuario corta la conexión el script pasa a este estado
  - TIMEOUT: Se puede configurar un máximo tiempo que puede ejecutarse el script, si se alcanza este tiempo pasa a este estado

Estado de la conexión con el navegador

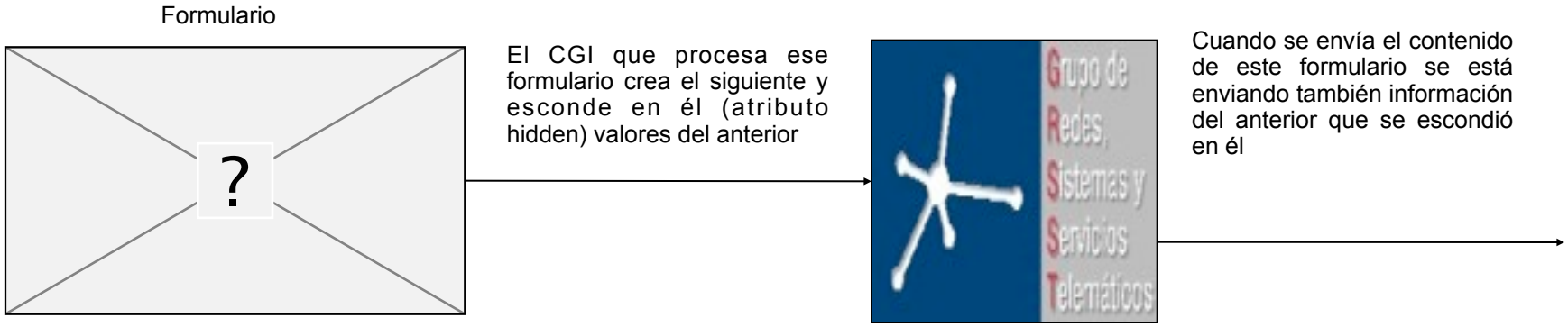
- Normalmente cuando el usuario corta la conexión y el script pasa al estrado ABORTED termina abruptamente la ejecución del script
- Se puede cambiar este modo de funcionamiento para que los scripts se ejecuten siempre hasta finalizar (por ejemplo llamando a la función `ignore_user_abort()`)
- El tiempo máximo típico que está configurado que pueda ejecutarse un script sin ser abortado por TIMEOUT es de 30 segundos pero puede cambiarse por ejemplo con la función `set_time_limit()`

Contenido

- Repaso de SQL
- PHP y MySQL
- **Sesiones en PHP**
  - Estado de la conexión con el navegador
  - **Cookies**
  - Sesiones

Persistent Client State HTTP Cookies

- Una de las limitaciones de la Web a la hora del desarrollo de aplicaciones/interfaces es que su funcionamiento es sin estado
- Cada petición de un URI es independiente de los anteriores y hay poca o ninguna información de lo que ha hecho el usuario anteriormente
- Eso quiere decir que formularios que ocupen varias páginas HTML son difíciles de implementar
- Un técnica clásica ha sido, al generar un CGI una página como resultado de un formulario esconder en esa página, en el siguiente formulario, la información que se quiere conservar...



Persistent Client State HTTP Cookies

- Las Cookies son el mecanismo más cómodo para almacenar información de estado en el cliente
- Al enviar una página Web el servidor puede indicar al cliente que almacene cierta información
- Ese cliente, cuando solicite otras páginas de ese servidor enviará en la solicitud esa información que se le pidió almacenar (la cookie)
- Cuando el cliente solicita un URL envía las cookies que pertenezcan a ese dominio y dentro del camino (path) especificado
- Las cookies se envían al servidor como parte de la cabecera HTTP

```
Cookie: NAME1=OPAQUE_STRING1 ;NAME2=OPAQUE_STRING2 ...
```

Persistent Client State HTTP Cookies

- Se introduce una cookie en el cliente mediante `Set-Cookie` en la cabecera HTTP:

  `Set-Cookie:` *`NAME`* `=`*`VALUE`* `; expires=` *`DATE`* `;path=` *`PATH`* `; domain=` *`DOMAIN_NAME`* `; secure`

  - El contenido de la cookie sigue el formato `NAME=VALUE` (ni punto-y-coma ni coma ni espacios).
  - Se le puede indicar una fecha máxima de validez a la cookie
  - Si se ha indicado el atributo `domain` el cliente, cuando haga una petición, enviará la cookie solo si en el URL al que se le solicita el nombre de dominio de la máquina está dentro de ese dominio
  - Con el atributo `path` se puede restringir el subconjunto de URLs del dominio a los que se les enviará la cookie al solicitar una página
  - Si se indica el atributo `secure` esta cookie solo se enviará si la conexión es segura (sobre SSL)

Contenido

- Repaso de SQL
- PHP y MySQL
- **Sesiones en PHP**
  - Estado de la conexión con el navegador
  - Cookies
  - **Sesiones**

Sesiones en PHP

- Sucede lo mismo con los scripts PHP que con CGIs: no se guarda estado
- PHP nos permite guardar el contenido de unas variables asociándolas a una sesión
- En realidad lo que hará será guardar esas variables localmente y mandar al usuario un identificador de sesión asociado a ese conjunto de variables en forma de una cookie
- Cuando el cliente solicita otro script PHP envía su cookie con el identificador de sesión
- Se accede al fichero correspondiente recuperando esas variables de forma que parece que conservar el contenido asignado por el anterior script...

Un script guarda la información obtenida del usuario en la pagina anterior en variables de sesión. Al mandar la pagina siguiente manda una cookie con el ID de la sesión

Al llamar a un nuevo script se le pasa el ID de la sesión con lo que este puede recuperar las variables guardadas

Variables y algunas funciones

- `$_SESSION`
  - Variable superglobal
  - Array con las variables que se guardan en la sesión (la clave es el nombre de la variable) => la clave tiene que cumplir las normas de una variable
- `bool session_start(void)`
  - Crea una nueva sesión o recupera las variables de una
  - En sesiones implementadas con cookies hay que llamar a esta función antes de que se envíe nada al navegador (para que se pueda poner la cookie en la cabecera HTTP)
- `bool session_destroy(void)`
  - Destruye los datos asociados a esta sesión (fichero donde guarda el servidor las variables de la sesión)
  - No invalida (unset) las variables ni la sesión
- `string session_encode(void)`
  - Devuelve una cadena con en contenido de la sesión codificado
- `session_decode(string data)`
  - Decodifica la cadena que se le pasa creando las variables que indica

PHP+Session: uso

- El identificador de la sesión va en la cabecera =>antes de enviar ningún dato hay que generar la sesión

```
<?php
session_start();
?>
HTML
<?php
$_SESSION['var'] = valor;
?>
Más HTML
```

```
<?php
session_start();
?>
HTML
<?php
$_SESSION['var'] = valor;
?>
Más HTML
```

Conclusiones

- Repaso de SQL
- PHP y MySQL
- Sesiones en PHP
  - Estado de la conexión con el navegador
  - Cookies
  - Sesiones

- Con estos mecanismos tenemos la base de hacer un interfaz interactivo con estado