Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Tutorial PHP4 Parte II: Base de Datos y MySQL (143138 lectures)

Per **Ricardo Galli Granada**, ***gallir*** (http://mnm.uib.es/gallir/)

Creado el 15/06/2001 03:22 modificado el 15/06/2001 03:22

Esta es la segunda parte del artículo Tutorial PHP4 - Parte I[1]. En éste introduzco las nociones básicas de base de datos y SQL y luego se explican como crear tablas en MySQL y acceder a la base de datos desde PHP, tanto para listar los datos en HTML como para introducir datos nuevos desde un formulario. Los ejemplos sobre los que se explican son muy sencillos, sólo se usan dos tablas (artículos y secciones) que permiten almacenar artículos con título y texto y que además pertenecen a distintas secciones.

Tutorial PHP4 Parte II: Base de Datos y MySQL

mnm.uib.es/~gallir/php/BaseDatos.html

Ricardo Galli (gallir@uib.es)

La relación entre PHP y bases de datos, especialmente MySQL y Postgres, es muy estrecha y beneficiosa. De hecho, cuando se habla de Web y PHP, es muy difícil que no se mencione también a una base de datos. Después de todo, el Web está pensado para almacenar y permitir los accesos a cantidades enormes de información. Mientras mayor sea la cantidad de información y más alta la frecuencia de actualización de un sitio web, mayor es su valor y sus ventajas sobre otros medios.

Tal vez la mayor ventaja de PHP sobre sus competidores es la integración con los sistemas de bases de datos y el soporte nativo a las distintas bases de datos existentes, libres y comerciales. Las razones principales para usar una base de datos son:

- Evitar redundancias.
- Evitar programas complicados.
- Búsquedas.
- Seguridad.
- Arquitectura *n-tier*

Arquitectura *n-tier*

Una arquitectura cliente/servidor es una *2-tier*, una *n-tie*r desagrega aún más las funciones, por ejemplo en web tenemos una *3-tier*:

1. Presentación: Navegador Web.
2. Lógica: Servidor web + programas o *scripts* PHP.
3. Almacenamiento de Datos: base de datos.

La comunicación entre el *tier-1* y el *tier-2* es a través de HTTP como soporte de comunicación y HTML como representación de datos. La comunicación entre el *tier-2* y el *tier-3* es a través del *middleware*, en nuestro caso PHP y las funciones de MySQL que se conectan al servidor. También puede hacerse mediante ODBC:

SQL

SQL, "Structured Query Language" representa un método estricto y más general de almacenamiento de datos que estándares anteriores. SQL es un estándar ANSI (www.ansi.org) y ECMA (www.ecma.ch).

La estructura básica de una base de datos relacional con SQL es muy simple. Una instalación de base de datos puede contener múltiples bases de datos, cada base de datos puede contener un conjunto de tablas. Cada tabla está compuesta de un conjunto de columnas cuidadosamente diseñadas y cada elemento (o entrada) de la tabla es una fila.

Hay cuatro sentencias de manipulación de datos soportado por la mayoría de los servidores SQL y que constituyen una gran parte de todas las cosas que se pueden hacer sobre una base de datos.

1. SELECT
2. INSERT
3. UPDATE
4. DELETE

Los cuatro tipos de sentencias permiten la manipulación de datos, pero no de la estructura de la base de datos. En otras palabras, se pueden usar para agregar o modificar la información almacenada en la base de datos, pero no para definir o construir una nueva base de datos. Para modificar la estructura, o agregar tablas y base de datos se usan las sentencias DROP, ALTER y CREATE:

```
alter table articulos add secciones int after id;

alter table articulos modify secciones int not null;
```

Base de Datos Simple

Trabajaremos con una base de datos muy sencilla (simple) que nos servirá para almacenar artículos compuestos de un título y un texto. Cada artículo estará identificado por un código numérico único. Además cada artículo pertenecerá a una sección determinada, que a su vez estará identificada por un código entero y tendrá un nombre:

```
create table secciones (id int default '0' not null auto_increment, nombre varchar(20),

         primary key (id), unique id(id));
```

```
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| id     | int(11)     |      | PRI | NULL    | auto_increment |
| nombre | varchar(20) | YES  |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
```

```
create table articulos (id int default '0' not null auto_increment,

seccion int, titulo text, texto text, primary key (id), unique id(id));
```

```
+-----------+---------+------+-----+---------+----------------+
| Field     | Type    | Null | Key | Default | Extra          |
+-----------+---------+------+-----+---------+----------------+
| id        | int(11) |      | PRI | NULL    | auto_increment |
| seccion   | int(11) |      |     | 0       |                |
| titulo    | text    | YES  |     | NULL    |                |
| texto     | text    | YES  |     | NULL    |                |
+-----------+---------+------+-----+---------+----------------+
```

```
+------------------+
| Tables_in_simple |
+------------------+
| articulos        |
| secciones        |
+------------------+
```

Cambiar Privilegios

El comando anteriormente citado para crear las tablas fue ejecutado desde el usuario privilegiado de la instalación del MySQL. Es muy mala idea trabajar con ese usuario desde las páginas PHP. Lo que se hace normalmente, a efectos de no desvelar las claves en el código PHP, es permitir el acceso a las base de datos a el usuario con el que es ejecutado el servidor web (Apache en este caso).

Si el Apache está ejecutándose como el usuario "wmaster", podemos darle la mayoría de los privilegios (seleccionar, insertar, modificar, borrar, crear y eliminar tablas) con el comando GRANT:

```
GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP

ON simple

TO wmaster@localhost;
```

Luego ejecutamos el comando flush para asegurarnos que los demás servidores MySQL actualizan esta información.

```
flush privileges;
```

Select

SELECT es el comando principal para obtener información de una base de datos. La sintaxis es muy sencilla:

```
SELECT campo1, campo2 FROM tabla WHERE condición;
```

En algunos casos, aunque no está recomendado, se pueden seleccionar todos las columnas de la tabla usando la siguiente sintaxis.

```
SELECT * FROM tabla WHERE secciones = 1;
```

```
select id, nombre from secciones;

+----+---------+
| id | nombre  |
+----+---------+
| 1  | Primera |
| 2  | Segunda |
| 4  | Tercera |
+----+---------+
```

```
+----+---------+----------------+-----------------------------------------------------+
| id | seccion | titulo         | texto                                               |
+----+---------+----------------+-----------------------------------------------------+
|  1 |    NULL | Primer articulo | Es el cuerpo del primer articulo y deberia estar bien |
+----+---------+----------------+-----------------------------------------------------+
```

Uniones

Una de las cosas que complica un poco el SELECT son las *uniones*, aunque es una de las propiedades más útiles de los *selects*. Un select sobre una base de datos que no soporte uniones, puede ser imaginada como una fila de una hoja de cálculo. Pero una base de datos SQL es por definición relacional Las uniones son definidas mediante una cláusula WHERE:

```
SELECT seccion, nombre, titulo FROM secciones, articulos

WHERE secciones.id = articulos.id;

+---------+---------+------------------+
| seccion | nombre  | titulo           |
+---------+---------+------------------+
|   1     | Primera | Primer articulo  |
|   2     | Segunda | Segundo articulo |
+---------+---------+------------------+
```

Insert

El comando para insertar nuevos datos a una base de datos es el *insert*.

```
INSERT INTO tabla (col1, col2, col3) VALUES (val1, val2, val3);

INSERT INTO tabla VALUES (val1, val2, val3);



insert into secciones values(1, 'Primera');

insert into secciones values(2, 'Segunda');



insert into articulos values (1, 'Primer articulo',

  'Es el cuerpo del primer articulo y deberia estar bien');

insert into articulos (seccion, titulo, texto) values (2, 2, 'Segundo articulo',

  'Es el cuerpo del segundo articulo.');
```

Update

Se utiliza para modificar datos previamente almacenados en la base de datos.

```
UPDATE table SET campo1='valor1, campo2='valor2'

WHERE condición

update articulos set seccion=1

  where id=2;
```

La clausula *where* es similar a la usada en el *select* para las uniones.

Delete

```
DELETE FROM tabla WHERE condición;

delete from articulos where seccion=1;
```

Es muy importante especificar la cláusula *where*, de otra forma se borrarán todos los datos almacenados en la tabla especificada.

Funciones Básicas de PHP/MySQL

Conexión a la base de datos

Si se quiere acceder a una base de datos desde PHP, primero debemos conectarnos al servidor MySQL:

```
$enlace = mysql_connect($hostname, $user, $password);
```

si se usan variables, o

```
$enlace = mysql_connect('localhost', 'master', 'laclave');
```

en el caso de constantes.

Los argumentos son opcionales, si no se especifican se supone "localhost" para el servidor, el mismo usuario que el servidor web para el usuario y clave vacía. La función **`mysql_connect`**retorna una conexión o enlace (*link*) a la base de datos que luego será usado para ejecutar los comandos SQL. El uso y almacenamiento de dicho valor es opcional, ya que la mayoría de las funciones de MySQL usan por defecto la último conexión. En caso de que se tengan varios enlaces o conexiones, hay que especificar cuál usar.

Luego de establecida la conexión hay que seleccionar una base de datos:

```
mysql_select_db($database);
```

si se usa variable o

```
mysql_select_db('simple');
```

si se usan constantes.

```
<?

    $con=mysql_connect();
    mysql_select_db("simple", $con);
    mysql_close($con);
?>
```

Conexiones Persistentes

Cada vez que un cliente se conecta y solicita una página PHP que se conectará a una base de datos, el proceso del Apache que lo ejecuta debe establecer la conexión al momento que se ejecuta el `mysql_connect`. Esta operación normalmente involucra arrancar un nuevo proceso del servidor MySQL, por lo que se produce un retardo (latencia) en la ejecución y visualización de la página.

Para evitar que se arranque un nuevo proceso por cada conexión a la base de datos, el MySQL del PHP permite especificar conexiones persistentes. En este tipo de conexiones, la conexión se mantiene abierta y podrá ser reutilizada más tarde en otra ejecución. Por lo tanto el servidor MySQL no acaba, sino que espera por nuevas peticiones, lo que acelera muchísimo la ejecución de los programas. Para ello no hay nada más que usar la función equivalente **`mysql_pconnect`**:

```
$enlace = mysql_pconnect($hostname, $user, $password);
```

Consultas e inserciones a la base de datos

A continuación veremos unos ejemplos muy sencillos pero representativo de las operaciones más comunes con bases de datos y páginas PHP:

- Recuperar datos desde una base de datos y listarlos en la página web.
- Permitir que un usuario inserte nuevos datos en la base de datos mediante un formulario HTML.

Con estos pequeños ejemplos se muestran las funciones más importantes y ya seréis capaces de realizar las mayoría de las operaciones comunes. Las operaciones de modificación son muy similares al INSERT, salvo que se usa el comando SQL UPDATE con una cláusula WHERE similar a las usadas en los SELECT.

En los ejemplos trabajaremos con las tablas previamente creadas de artículos y secciones.

Obtener la Lista de Bases de Datos

Lo primero que haremos es conectarnos a la base de datos y seleccionar las base de datos existentes en el servidor. Ello se logra con la función **`mysql_list_dbs`** que retorna un resultado consistente en la lista de base de datos. Luego sólo queda recorrer toda la lista con la función `mysql_fetch_object` e imprimir el nombre de cada base de datos.

```
<?

    $con=mysql_pconnect()
        or die ("No pude conectarme");
    echo "<B>Connectado</B><P>";

    $db_list = mysql_list_dbs($con);
    while ($row = mysql_fetch_object($db_list)) {
        echo $row->Database . "<br>";
    }
    mysql_close($con);
?>
```

Listar el contenido de una tabla

En el siguiente ejemplo listaremos todo el contenido de la tabla secciones y también el número secciones que existen. Las funciones que se usan son:

- **`mysql_select_db`**: Esta función selecciona una base de datos para ser usada en las siguientes funciones. Si la función se ejecuta correctamente (en enlace es correcto y la base de datos existe) devuelve TRUE, caso contrario FALSE.
- **`mysql_query`**: Ejecuta un comando SQL y retorna un resultado (cursor) que mantiene la información necesaria para "recorrer" el conjunto de filas que cumplen con la condición especificada en el SELECT. Si el comando se ejecuta correctamente (la sintaxis es correcta, las tablas y columnas existen y tenemos permiso para hacer un SELECT) devuelve TRUE, caso contrario devuelve FALSE.
- **`mysql_affected_rows`**: Esta función nos devuelve la cantidad de filas afectadas o seleccionadas con el comando anterior. Es similar a la función **`mysql_num_rows`**, pero además puede ser usada para UPDATE o DELETE.
- **`mysql_fetch_array`**: Esta es una función muy potente y eficiente, ya que recupera de la base de datos la siguiente fila que nos interesa y pone todos los datos en un array que puede ser accedido de dos formas: con un índice numérico como un array normal, o como un array asociativo usando los nombres de las columnas especificados en el SELECT: En el ejemplo lo usamos como un array asociativo con los nombres de las columnas "id" y "nombre".

```
<?

    $con=mysql_pconnect()
```

```
        or die ("No pude conectarme");

    mysql_select_db("simple", $con)
        or die("No puedo acceder a la base de datos");
    echo "Seleccionando de simple<br><hr>";

    $resultado=mysql_query("select * from secciones", $con);
    $items = mysql_affected_rows($con);
    echo "Número de filas: $items <p>";

    while( ($fila=mysql_fetch_array($resultado)) ) {
        echo $fila["id"] . ": " . $fila["nombre"] . "<p>";
    }

    mysql_close($con);

?>
```

Insertar filas desde un formulario

En el siguiente ejemplo mostramos como definir un formulario sencillo y luego insertar los valores del formulario en una tabla. En el ejemplo lo que hacemos es permitir insertar campos a la tabla "secciones". Como se puede observar, no empleamos ninguna función nueva, sino que hacemos uso de la instrucción SQL INSERT en la función `mysql_query` ya usada en el ejemplo anterior.

El funcionamiento del programa es muy sencillo, define un formulario que un solo campo (nombre) y la acción del formulario es el mismo *script* PHP. Al principio del programa verificamos si la llamada es desde el formulario (mediante la verificación de la variable `$submit`), si es así, insertamos el valor de la variable "nombre" (definida en el formulario) en una nueva fila de la tabla "secciones".

Notar que no especificamos el valor de "id" ya que este valor es generado automáticamente por el MySQL (ver el `autoincrement` del `create`).

```
<?php
if ($submit) {

  $db = mysql_pconnect("localhost");
  mysql_select_db("simple",$db);
  $sql = "INSERT INTO secciones (nombre) VALUES ('$nombre')";
  $result = mysql_query($sql);
  echo "Información introducida.\n";

} else{
?>
  <form method="post" action="<?php echo $PHP_SELF?>">
  Nombre:<input type="Text" name="nombre"><br>
  <input type="Submit" name="submit" value="Enter information">
  </form>

<?php
} // end if
?>
```

Unión e inserción más compleja

En el siguiente ejemplo lo que hacemos es permitir insertar nuevos artículos, pero hay un requerimiento nuevo: cada artículo debe tener una sección asociada (en el campo "seccion"), por lo que es importante que el usuario especifique un valor correcto. Es decir, que el código de sección (y su nombre) exista en la tabla "secciones". La solución es sencilla si no hay demasiados valores posibles: poner un campo de entrada del tipo "SELECT" donde listaremos los códigos y nombres de las secciones existentes.

También mostramos un ejemplo de UNION. Nos interesa mostrar en la pagina un listado de todos los artículos y la sección a la que pertenecen. Para poder obtener el nombre de la sección y los títulos de los artículos debemos hacer una unión entre la tabla de artículos y la de secciones y seleccionar las columnas que nos interesan (título del artículo y nombre de la sección).

```
<?php
$db = mysql_pconnect("localhost");
mysql_select_db("simple",$db);
if ($submit) {

  $sql = "INSERT INTO articulos (seccion, titulo, texto)
          VALUES ($seccion, '$titulo', '$texto')";
  $result = mysql_query($sql);
  if($result) {
      echo "<h3>Información introducida.</h3>\n";
  } else {
      echo "<h3>No se pudo introducir el artículo</h3>\n";
  }

}

// Listamos todos los artículos y sus secciones (UNION)
$result = mysql_query("select nombre, titulo from articulos, secciones
        where articulos.seccion=secciones.id", $db);
while ( ($datos = mysql_fetch_array($result)) ) {
    print $datos["titulo"] . " (<i>" . $datos["nombre"] . "</i)<br>\n";
}

// Damos la opcion de recargar la página
echo "<CENTER><A HREF=\"$PHP_SELF\">Actualizar</A></CENTER>";

?>
  <H1>Insertar nuevo artículo</H1>
  <form method="post" action="<?php echo $PHP_SELF?>">
  Sección: <select name=seccion>
<? // Ahora seleccionamos de la BD las secciones existentes
  $result = mysql_query("select id, nombre from secciones
         order by nombre asc", $db);
  while ( ($datos = mysql_fetch_array($result)) ) {
    printf("<option value=%d>%s</option>\n", $datos["id"], $datos["nombre"]);
  }



?>
  </select><br>
  Título:<textarea cols="40" rows="2" name="titulo"><? echo $titulo ?></textarea><br>
  Texto:<textarea cols="40" rows="10" name="texto"><? echo $texto ?></textarea><br>
  <input type="Submit" name="submit" value="Enter information">
  </form>
```

Lista de enlaces de este artículo:

1. http://bulma.net/body.phtml?nIdNoticia=655

E-mail del autor: gallir _ARROBA_ uib.es