Manejo de datos BLOB con PHP y MySQL

programacion.net/articulo/manejo_de_datos_blob_con_php_y_mysql_194

Cedido por MySQL Hispano.

Sin duda alguna, una de las preguntas más frecuentes en relación a MySQL es, "¿Cómo puedo almacenar archivos en una base de datos?". La respuesta es, se tienen que usar los tipos de datos BLOB. Estos BLOBs (Binary Large Objects) pueden almacenar prácticamente cualquier tipo de datos, incluyendo documentos de MS Word, imagenes gif/jpeg, archivos PDF, MP3s, etc.

En este artículo vamos a mostrar como crear un repositorio de archivos binarios usando PHP y MySQL para poder almacenar diferentes tipos de archivos. Veremos como almacenar cada uno de los archivos en una base de datos, para posteriormente recuperarlos.

Para probar los ejemplos en este artículo, es necesario tener acceso a un servidor con soporte para PHP, además de contar con un servidor MySQL. Se asume que se cuentan con los privilegios apropiados para crear la base de datos, y que MySQL está corriendo en el mismo servidor que PHP.

Creando la base de datos

Nuestro repositorio de documentos usará una base de datos que contenga únicamente una tabla para almacenar dichos documentos.

```
[[email protected]]$ mysqladmin create repositorio
[[email protected]]$
[[email protected]]$ mysql repositorio
Welcome to the MySQL monitor.  Commands end with ; or g.
Your MySQL connection id is 139 to server version: 3.23.41

Type 'help;' or 'h' for help. Type 'c' to clear the buffer.

mysql> CREATE TABLE archivos(
    -> id int not null auto_increment primary key,
    -> nombre varchar(50),
    -> titulo varchar(50),
    -> contenido mediumblob,
    -> tipo varchar(50));
Query OK, 0 rows affected (0.01 sec)

mysql> grant all on repositorio.* to [email protected] identified by 'holahola'
Query OK, 0 rows affected (0.46 sec)
```

Ahora tenemos una base de datos llamada repositorio, la cual contiene una tabla llamada `archivos`.

Los detalles de cada campo en la tabla archivos se muestra a continuación:

- **id** - Un número entero que nos proporcionará un identificador único para cada archivo que iremos almacenando. Éste se irá incrementando automáticamente cada vez que se vaya agregando un nuevo registro.
- **nombre** - El nombre original del archivo, por ejemplo, foto.gif, curriculum.doc, etc
- **titulo** - Una breve descripción de cada archivo que se irá guardando en la tabla, por ejemplo "Cartita para mi novia", o "La foto de mi perro". Este título será usado posteriormente al estar revisando los archivos de nuestro repositorio a través de una página web.
- **contenido** - Un campo de tipo binario (blob) para guardar el contenido de cada archivo. En nuestra tabla hemos usado un tipo mediumblob, el cuál puede almacenar archivos hasta de 16MB.
- **tipo** - Como veremos más adelante, cada archivo (ya sea un .doc, .gif, .pdf, etc) tiene un tipo único. Cuando se envía un archivo hacía el servidor web a través de una página web, el navegador le envía al servidor la información acerca del tipo del archivo, o bien, del tipo de contenido del archivo. Los tipos de contenido son simples cadenas. El tipo de contenido para un archivo MS Word es "application/msword", el tipo de contenido para una imagen GIF es "image/gif", etc.

Agregando archivos a la base de datos

Ahora que tenemos lista la base de datos que servirá como repositorio de nuestros archivos, vamos a crear un simple página web que permita seleccionar un archivo desde el navegador para posteriormente enviarlo a un script en PHP que se va a encargar de almacenarlo en nuestra base de datos. La página web será nombrada escoger_archivo.html y el script en PHP se llamará guardar_archivo.php.

La página web puede contener todo el código HTML que se desee, pero es necesario que se incluya el siguiente formulario para que se tenga la opción de escoger un archivo y enviarlo al servidor.

```
<form enctype="multipart/form-data" action="guardar_archivo.php" method="post">
Descripción <input type="text" name="titulo" size="30">
Ubicación <input type="file" name="archivito">
<input type="submit" value="Enviar archivo">
</form>
```

Del código que hemos escrito tenemos que subrayar lo siguiente:

- El formulario necesita un atributo `enctype` con un valor `multipart/form-data`.
- El método de envío tiene que ser POST.

- Se tiene que usar por lo menos un campo del tipo `file`

Estos son los tres requerimientos básicos que debe cumplir la página HTML que servirá para subir los archivos al servidor. Antes de escribir el código del script `guarda_archivo.php`, vamos a comentar algo acerca de la forma en como se tomaran los datos del archivo que se va a recibir.

Desde la versión 4.1 de PHP se recomienda se utilice el arreglo `$_FILES` para leer los datos del archivo que se está subiendo al servidor. A continuación se listan los elementos que contiene este arreglo. Nótese que el nombre de los índices del arreglo depende de como se haya nombrado el campo del tipo file en el formulario.

$_FILES['archivito']['name']
Es el nombre original del archivo.

$_FILES['archivito']['type']
El tipo MIME del archivo,.. image/gif, application/pdf, application/msword,.. etc

$_FILES['archivito']['size']
El tamaño del archivo en bytes.

$_FILES['archivito']['tmp_name']
La ubicación del archivo temporal que se crea cuando se sube un archivo al servidor. Es en esta variable de donde se leen los datos del archivo en sí. Si estos datos no son copiados o movidos a otro lugar, o en nuestro caso, almacenados en una base de datos, se pueden perder, ya que PHP elimina este archivo después de un determinado tiempo.

Por ejemplo, estos son los posibles valores para un archivo j-odbc.zip que se ha subido al servidor:

```
$_FILES["archivito"][name] => j-jdbc.zip
$_FILES["archivito"][type] => application/zip
$_FILES["archivito"][tmp_name] => /tmp/phpvXQpqP
$_FILES["archivito"][size] => 337945
```

Cabe mencionar que se tienen que revisar, y en su caso modificar las siguientes variables del archivo de configuración de PHP para que se puedan subir los archivos al servidor, y puedan ser correctamente manejados por un script de PHP.

- **file_uploads** - Le dice a PHP si se pueden, ó no, subir archivos al servidor. Esta variable debe tener el valor "On".
- **upload_max_filesize** - Le indica a PHP cuál es el tamaño máximo del archivo que se puede aceptar. Se puede utilizar el sufijo "M" para indicar el valor en MegaBytes, por ejemplo, con un valor 2M se acepta un archivo máximo de 2 MegaBytes.
- **upload_tmp_dir** - Es el directorio en el que se copia temporalmente el contenido del archivo cuando se sube al servidor.

Ahora es el momento de mostrar el código del script en PHP que va a guardar el archivo en nuestra base de datos.

```
/* guardar_archivo.php */

require("dbconnect.inc.php");

$archivo = $_FILES["archivito"]["tmp_name"];
$tamanio = $_FILES["archivito"]["size"];
$tipo    = $_FILES["archivito"]["type"];
$nombre  = $_FILES["archivito"]["name"];
$titulo  = $_POST["titulo"];

if ( $archivo != "none" )
{
   $fp = fopen($archivo, "rb");
   $contenido = fread($fp, $tamanio);
   $contenido = addslashes($contenido);
   fclose($fp);

   $qry = "INSERT INTO archivos VALUES
           (0,'$nombre','$titulo','$contenido','$tipo')";

   mysql_query($qry);

   if(mysql_affected_rows($conn) > 0)
      print "Se ha guardado el archivo en la base de datos.";
   else
      print "NO se ha podido guardar el archivo en la base de datos.";
}
else
   print "No se ha podido subir el archivo al servidor";
```

El archivo dbconnect.inc.php contiene únicamente las instrucciones para conectarse a MySQL y seleccionar la base de datos que se va a utilizar. El código de este programita se muestra a continuación.

```
/* dbconnect.inc.php */

$conn = mysql_connect("localhost","bingo","holahola");
mysql_select_db("repositorio");
```

Listando los archivos de la base de datos

Ya que hemos guardado algunos archivos en nuestro repositorio, ahora podemos listar la información de éstos, para posteriormente descargarlos. Esta es la labor del script que se muestra a continuación.

```
/* listar_archivos.php */

require("dbconnect.inc.php");

$qry = "SELECT id, nombre, titulo, tipo FROM archivos";
$res = mysql_query($qry);

while($fila = mysql_fetch_array($res))
{
print "$fila[titulo]
<br>
$fila[nombre] ($fila[tipo])
<br>
<a href='descargar_archivo.php?id=$fila[id]'>Descargar</a>
<br>
<br>";
}
```

Descargando archivos de la base de datos

Como se puede observar en el código del listado anterior, se ha puesto una liga a un archivo descargar_archivo.php. La funcionalidad de este programita será la de leer los datos de los archivos que se encuentran en la base de datos y mandarlos al navegador. Dependiendo del tipo de archivo del cuál se trate, el navegador podrá mostrar por él mismo el contenido del archivo.

```
/* Script descargar_archivo.php */

require("dbconnect.inc.php");

$qry = "SELECT tipo, contenido FROM archivos WHERE id=$id";
$res = mysql_query($qry);
$tipo = mysql_result($res, 0, "tipo");
$contenido = mysql_result($res, 0, "contenido");

header("Content-type: $tipo");
print $contenido;
```

Comentarios finales

Como se puede observar, el código PHP de los programitas es demasiado simple, y no tienen ningún grado de dificultad. El código se ha hecho a propósito lo más simple posible, sin embargo puede adecuarse o tomarse como base para realizar una aplicación real más completa.

COMPARTIR EN FACEBOOK

COMPARTIR EN TWITTER

COMPARTIR EN GOOGLE +