Upload de imagens para banco de dados (mysql) com Java Server Pages

Neste artigo explicarei os métodos para se trabalhar com imagens em Banco de Dados. Mais especificamente estarei abordando este tema utilizando o MySQL e um componente do projeto jakarta chamado FileUpload.

Pra começar precisamos instalar e configurar o ambiente. Você precisará do MySQL, que pode ser encontrado em: http://www.mysql.com ou, se preferir, poderá criar uma conta em http://www.freesql.com que é um servidor MySQL free.

Pronto, feito isso, é preciso baixar e instalar o componente FileUpload do projeto Jakarta. Este pode ser encontrado em http://jakarta.apache.org/commons/fileupload. Para instalar o componente, basta descompactá-lo em uma pasta qualquer e copiar o arquivo *commons-fileupload-1.0.jar* para a pasta $CATALINA_HOME/commons/lib (para disponibilizar para todos os webapps), ou para a pasta seu_web_app/WEB-INF/lib (para que somente a sua aplicação tenha acesso ao componente).

Feito isso, o ambiente está devidamente preparado para começarmos. Para facilitar utilizarei neste exemplo uma tabela simples com apenas três campos (sendo um autonumerável, um texto e, por fim, um do tipo Blob).

Para criar a Tabela:

```
create table foto (
      id tinyint(4) not null auto_increment,
      comentario longtext not null,
      foto blob not null,
      primary key (id) );
```

Pronto, nossa tabela está criada.
Atente para o tipo do campo foto (BLOB – Binary Large Object) , que é o tipo perfeito para armazenarmos fotos, já que uma foto (aqui, em especial, JPEG) é um arquivo binário e geralmente grande (200kb em média).

Bom, agora teremos que construir o nosso formulário que irá chamar a página JSP para inclusão da imagem no banco de dados. Neste tutorial não farei os testes de consistência, para ver se todos os (dois) campos foram preenchidos, mesmo porque essa não é a idéia deste artigo.

```
<html>
<head>
<title>Inclusao de Fotos</title>
</head>
<body bgcolor="#FFFFFF">
```

```
<h2>Incluir foto no banco de dados</h2><br>
<form action="gravaFoto.jsp" method="post" enctype="multipart/form-
data" name="foto_up" id="foto_up">

<input type="text" name="comentario">
<br>
<input type="file" name="foto">
<br>
<input type="submit" value="enviar">
</form>
</body>
</html>
```

Até aqui normal, a única diferença foi o atributo *enctype="multipart/form-data"* na tag <form>. Este atributo informa que o formulário irá enviar na requisição, além de puro texto, dados binários (imagens, arquivos, etc.).

Bom, chegou a hora de codificarmos!!!! Aqui abordarei como fazer usando paginas JSP, para fazer com servlet o método é praticamente o mesmo (só irá mudar o nome de alguns objetos, caso você os declare com nome diferente), aliás, no fim tudo vira servlet mesmo !!! Bom, mas antes de começar a codificar você precisa incluir no seu projeto o *commons-fileupload-1.0.jar*. Para isso (usando o eclipse) selecione o seu projeto na janelinha Package Explorer e depois vá em Project -> Properties. Feito isso uma janela se abrirá. Clique agora em Java Build Path (que fica a esquerda na janela), depois clique na aba Libraries. Clique no no botão Add External Jars... E selecione o *commons-fileupload-1.0.jar*
Agora sim ta tudo pronto para começarmos de fato!

Aqui segue um código de uma classe Java bem simples só para automatizar a criação de conexão com o banco de dados:

Arquivo MyConnection.java:

```
package mypackage;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 * @author Darkseid
 *
 * Classe utilizada para estabelecer conexao com o banco de dados
 */
public class MyConnection {
	static Connection con=null;


	/**
	 *	Construtor
	 */
	public MyConnection() {
		super();
	}

	//cria conexao com o banco de dados
	public static Connection createConnection(String banco) throws Exception {
```

```
                try {
                        // instanciando o driver
                        Class.forName("com.mysql.jdbc.Driver");
                        con = DriverManager.getConnection
("jdbc:mysql://www.freesql.org/"+banco, "usuario", "senha");
                }catch (ClassNotFoundException e) {
                        throw new Exception("Driver nao encontrado");
                }catch (SQLException e) {
                        throw new Exception("Erro com o banco de dados");
                }

                return con;
        }

        // fecha a conexao
        public static void closeConnection () throws DAOException {

                try {
                        if (!con.isClosed())
                                con.close();
                } catch (SQLException e) {
                        throw new DAOException ("Nao foi possivel fechar a conexao");
                }


        }

}
```

//----------

Abaixo segue o código do arquivo gravaFoto.jsp

```
<%@ page language="java" import="org.apache.commons.fileupload.*, java.util.*, java.sql.*" %
>
<html>
<body bgcolor="#FFFFFF">

<%
// Verificando se o form possui campo(s) com dados binários
if (FileUpload.isMultipartContent(request)) {

        // criando o objeto para cuidar do upload
        DiskFileUpload fu = new DiskFileUpload();
        // setando o tamanho maximo em bytes para upload
        fu.setSizeMax(800000);

        try {
                // parseando a requisição e retornando uma lista com os campos
                // encontrando, tanto textos, quanto dados binários (arquivos binários)
                List items = fu.parseRequest(request);
                Iterator i = items.iterator();
                FileItem fi;
                String cmt=null;

                // laço para pegar todos os campos do form
                while (i.hasNext()) {
                        fi = (FileItem)i.next();
                        // teste para ver se o campo em questão é campo do formulario
                        // ou um arquivo
                        if (fi.isFormField()) {
                                // pegando o valor do campo do formulário (comentário)
                                cmt = fi.getString();
                        }
                        else{
                                // obtendo o tamanho da foto
                                int size = (int) fi.getSize();
                                // array de byte para armazenar a foto
                                byte [] imagem = new byte[size];
```

```
                                // carregando a foto no array
                                upload.read(imagem, 0, size);

                                // chamado o método estático para conectar com o banco
                                Connection con = MyConnection.createConnection("bolao");
                                // onde tem “?”, será substituído por valores posteriormente.
                                PreparedStatement pstm = con.prepareStatement("INSERT INTO
foto (comentario, foto) VALUES (?, ?)");

                                // inserindo o comentário no lugar da primeira interrogação
                                pstm.setString(1, cmt);

                                // inserindo a foto no lugar da segunda interrogação
                                pstm.setBytes(2, imagem);
                                // executando a query
                                pstm.executeUpdate();

                        }


                }


        } catch (FileUploadException e) {
                out.print(“Erro no upload do arquivo”);
        } catch (SQLException e) {
                out.print(“Erro na consulta com o banco de dados”);
        } catch (Exception e) {
                out.print(e.getMessage());

}else {
        out.print(“O Formulario nao possui dados binários”)
}

} %>

<h2>Fonto inserida com sucesso no banco de dados!</h2>

</body>
</html>
```

Bom, agora falta exibir a foto na pagina JSP, certo?! Para isso vou criar uma classe que converte uma seqüência de bytes em JPEG. Ah, antes que eu me esqueça esse código é do **Farnetani**!

```
public class JPGManager {

        static public void encodeJPG(OutputStream out, byte [] image) throws IOException{

                int BUFFER = image.length;
                InputStream fs = new ByteArrayInputStream(image);
                JPEGImageDecoder decoder = JPEGCodec.createJPEGDecoder(fs);
                BufferedImage bImage = decoder.decodeAsBufferedImage();
                JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
                encoder.encode(bImage);

                fs.close();
                fs = null;
                encoder = null;
                bImage = null;
        }
}
```

O que resta a fazer agora é simplesmente criar uma JSP para pegar no banco o conteúdo do campo BLOB e passa-lo como parâmetro para a função acima, juntamente com o

OutputStream do servlet (pois na verdade uma pagina jsp é transformada em servlet antes de ser compilada).
Arquivo verFoto.jsp

```
<%@ page language="java" import="java.lang.*" %>
<%

try {
    // criando a conexao com o banco de dados
    Connection con = MyConnection.createConnection("bolao");
    Statement stm = con.createStatement();

    int codigo;
    // pegando o codigo da foto a ser exibida
    if (request.getParameter("numero") == null)
        codigo = 1;
    else
         codigo = Integer.parseInt(request.getParameter("numero"));

    // procurando pela foto na tabela
    ResultSet rs = stm.executeQuery("SELECT * FROM foto WHERE codigo = "+codigo);
    if (rs.next()) {
         // pegando o conteudo do campo foto (BLOB, binario)
         Blob blob = rs.getBlob("foto");
          // Codificando para JPG e enviando para o browser, através do Output do servlet
         JPEGManager.encodeJPG(response.getOutputStream(), blob.getBytes(1, (int)
blob.length()) );

}catch (Exception e){
    // erro
    %>Erro encontrado<%
}

%>
```

Terminamos! Para ver suas fotos, basta chamar
http://localhost:8080/sua_aplicacao/verFoto.jsp?numero=x
Onde x é o numero da foto que deseja visualizar.

Bom, é isso, espero que tenham gostado! Qualquer dúvida, sugestão ou crítica:
darkseid@click21.com.br