

Upload files to database (Servlet + JSP + MySQL)

Java Performance

Clean Code

The Clean Coder

The Pragmatic Programmer

Last Updated on 29 November 2014 |

Print  Email

This tutorial shows how to implement a Java web application that uploads files to server and save the files into database.

Table of content:

1. [Creating MySQL database table](#)
2. [Coding upload form page](#)
3. [Coding file upload servlet](#)
4. [Coding message page](#)
5. [Testing the application and verifying file stored in database](#)

The application applies the following technologies:

- **Servlet 3.0:** Using Servlet 3.0 we can write code to handle file upload easily. For detailed explanation of how to upload file with Servlet 3.0, read the tutorial: [How to write upload file servlet with Servlet 3.0 API](#).
- **MySQL database 5.5:** We will store uploaded files in MySQL database. For more details about how to store files in MySQL database, read the article: [Insert file data into MySQL database using JDBC](#).

The application will consist of the following source files:

- `Upload.jsp`: presents a form which allows users entering some information (first name and last name), and picking up a file (a portrait image).
- `FileUploadDBServlet`: captures input from the upload form, saves the upload file into database, and forwards the users to a message page.
- `Message.jsp`: shows either successful or error message.

Now, let's go through each part of the application in details.

1. Creating MySQL database table

First, let's create a database and a table in MySQL. Execute the following script using either *MySQL Command Line Client* or *MySQL Workbench*:

```
1 create database AppDB;
2
3 use AppDB;
4
5 CREATE TABLE `contacts` (
6   `contact_id` int(11) NOT NULL AUTO_INCREMENT,
7   `first_name` varchar(45) DEFAULT NULL,
8   `last_name` varchar(45) DEFAULT NULL,
9   `photo` mediumblob,
10  PRIMARY KEY (`contact_id`)
11 ) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

The script will create a database named *AppDB* and a table named *contacts*. File will be stored in the column *photo* which is of type `mediumblob` which can store up to 16 MB of binary data. For larger files, use `longblob` (up to 4 GB).



2. Coding upload form page

Write code for the upload form as follows (`Upload.jsp`):

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2     pageEncoding="ISO-8859-1"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4     "http://www.w3.org/TR/html4/loose.dtd">
5  <html>
6  <head>
7  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8  <title>File Upload to Database Demo</title>
9  </head>
10 <body>
11     <center>
12         <h1>File Upload to Database Demo</h1>
13         <form method="post" action="uploadServlet" enctype="multipart/form-data">
14             <table border="0">
15                 <tr>
16                     <td>First Name: </td>
17                     <td><input type="text" name="firstName" size="50"/></td>
18                 </tr>
19                 <tr>
20                     <td>Last Name: </td>
21                     <td><input type="text" name="lastName" size="50"/></td>
22                 </tr>
23                 <tr>
24                     <td>Portrait Photo: </td>
25                     <td><input type="file" name="photo" size="50"/></td>
26                 </tr>
27                 <tr>
28                     <td colspan="2">
29                         <input type="submit" value="Save">
30                     </td>
31                 </tr>
32             </table>
33         </form>
34     </center>
35 </body>
36 </html>
```

This page shows two text fields (first name and last name) and a file field which allows the users choosing a file to upload.

The `action` attribute of this form is set to `uploadServlet` which is URL mapping of the servlet we will create in the next section.

[Servlet and JSP \(A Tutorial\) \[Kindle Edition\]](#) - This book covers everything in Servlet and JSP in a concise and easy to understand way. Every Java programmer should understand the pillar stones of Servlet and JSP before diving into other web frameworks which are built on top of Servlet and JSP.

3. Coding file upload servlet

Create a servlet class named `FileUploadDBServlet.java` with the following code:

```
1  package net.codejava.upload;
2
3  import java.io.IOException;
4  import java.io.InputStream;
5  import java.sql.Connection;
6  import java.sql.DriverManager;
7  import java.sql.PreparedStatement;
8  import java.sql.SQLException;
9
10 import javax.servlet.ServletException;
11 import javax.servlet.annotation.MultipartConfig;
12 import javax.servlet.annotation.WebServlet;
13 import javax.servlet.http.HttpServlet;
14 import javax.servlet.http.HttpServletRequest;
15 import javax.servlet.http.HttpServletResponse;
16 import javax.servlet.http.Part;
17
18 @WebServlet("/uploadServlet")
19 @MultipartConfig(maxFileSize = 16177215) // upload file's size up to 16MB
20 public class FileUploadDBServlet extends HttpServlet {
21
22     // database connection settings
23     private String dbURL = "jdbc:mysql://localhost:3306/AppDB";
24     private String dbUser = "root";
25     private String dbPass = "secret";
26
27     protected void doPost(HttpServletRequest request,
28         HttpServletResponse response) throws ServletException, IOException {
29         // gets values of text fields
30         String firstName = request.getParameter("firstName");
31         String lastName = request.getParameter("lastName");
32
33         InputStream inputStream = null; // input stream of the upload file
34
35         // obtains the upload file part in this multipart request
36         Part filePart = request.getPart("photo");
37         if (filePart != null) {
38             // prints out some information for debugging
39             System.out.println(filePart.getName());
40             System.out.println(filePart.getSize());
41             System.out.println(filePart.getContentType());
42
43             // obtains input stream of the upload file
44             inputStream = filePart.getInputStream();
45         }
46
47         Connection conn = null; // connection to the database
48         String message = null; // message will be sent back to client
49
50         try {
51             // connects to the database
52             DriverManager.registerDriver(new com.mysql.jdbc.Driver());
53             conn = DriverManager.getConnection(dbURL, dbUser, dbPass);
54
55             // constructs SQL statement
```

```

56     String sql = "INSERT INTO contacts (first_name, last_name, photo) values (?, ?, ?)";
57     PreparedStatement statement = conn.prepareStatement(sql);
58     statement.setString(1, firstName);
59     statement.setString(2, lastName);
60
61     if (inputStream != null) {
62         // fetches input stream of the upload file for the blob column
63         statement.setBlob(3, inputStream);
64     }
65
66     // sends the statement to the database server
67     int row = statement.executeUpdate();
68     if (row > 0) {
69         message = "File uploaded and saved into database";
70     }
71 } catch (SQLException ex) {
72     message = "ERROR: " + ex.getMessage();
73     ex.printStackTrace();
74 } finally {
75     if (conn != null) {
76         // closes the database connection
77         try {
78             conn.close();
79         } catch (SQLException ex) {
80             ex.printStackTrace();
81         }
82     }
83     // sets the message in request scope
84     request.setAttribute("Message", message);
85
86     // forwards to the message page
87     getServletContext().getRequestDispatcher("/Message.jsp").forward(request, response);
88 }
89 }
90 }

```

In this servlet, we use two annotations:

- `@WebServlet`: marks this servlet so that the servlet container will load it at startup, and map it to the URL pattern `/uploadServlet`.
- `@MultipartConfig`: indicates this servlet will handle multipart request. We restrict maximum size of the upload file up to 16 MB.

The `doPost()` method carries out all the details. Here, there are three noticeable points:

- Obtaining the part of upload file in the request:

```
1 | Part filePart = request.getPart("photo");
```

The name "photo" is name of the file input field in the `Upload.jsp` page.

- Obtaining input stream of the upload file:

```
1 | inputStream = filePart.getInputStream();
```

- And pass the input stream into the prepared statement:

```
1 | statement.setBlob(3, inputStream);
```

4. Coding message page

Create a JSP page named as `Message.jsp` with the following code:

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
4   "http://www.w3.org/TR/html4/loose.dtd">
5 <html>
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
8 <title>Message</title>
9 </head>
10 <body>
11     <center>
12         <h3><%=request.getAttribute("Message")%></h3>
13     </center>
14 </body>
15 </html>
```

This page simply displays value of the variable “Message” in the request scope.

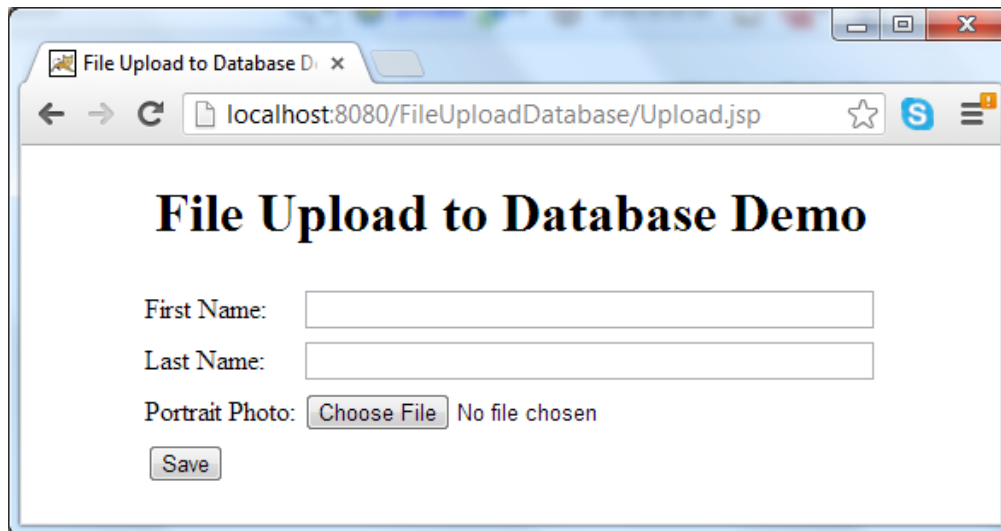
Learn how to create dynamic websites using the Java programming language with this video tutorial: [Servlets and JSPs: Creating Web Applications With Java](#)

5. Testing the application and verifying file stored in database

Supposing the application is deployed on localhost at port 8080, under the context root `/FileUploadDatabase`, type the following URL:

`http://localhost:8080/FileUploadDatabase/Upload.jsp`

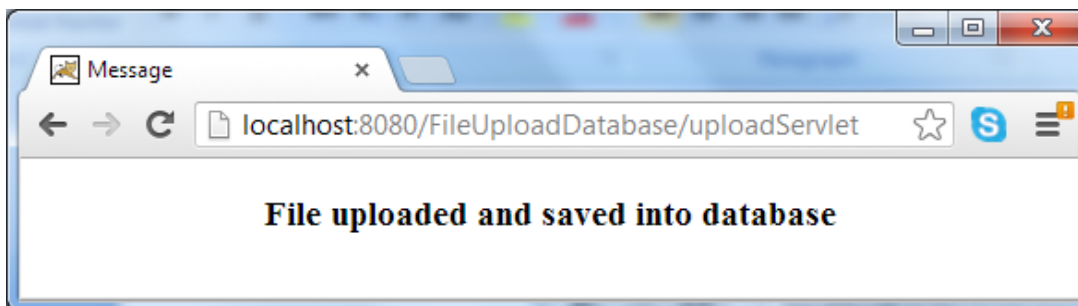
The following upload form is displayed:



The screenshot shows a web browser window with the title "File Upload to Database Demo". The address bar shows the URL "localhost:8080/FileUploadDatabase/Upload.jsp". The form contains the following elements:

- First Name:** A text input field.
- Last Name:** A text input field.
- Portrait Photo:** A section containing a "Choose File" button and the text "No file chosen".
- Save:** A button at the bottom of the form.

Type first name, last name, and pick up an image file. Click **Save** button, if everything is going well, this message appears:



To verify that the file is stored successfully in the database, open a new SQL Editor in MySQL Workbench and execute the following query:

```
1 | select * from contacts;
```

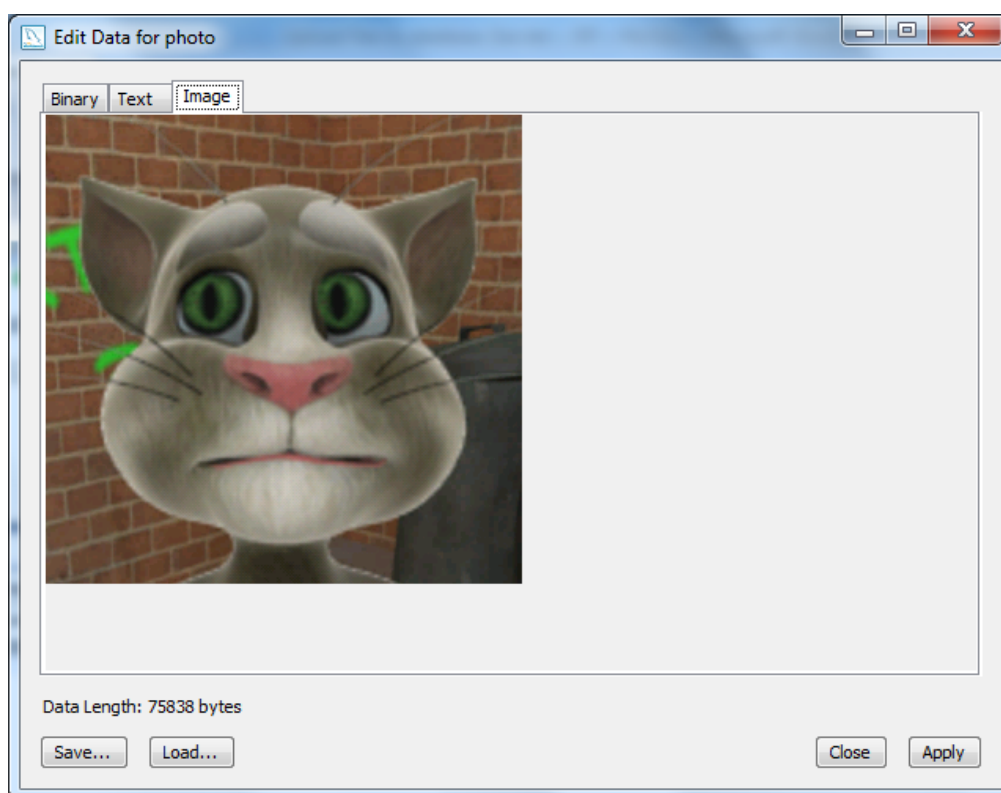
The query would return the newly inserted record, right click on the `BLOB` cell of the column `photo`, and select **Open Value in Editor** from the context menu:

	contact_id	first_name	last_name	photo
▶	3	Tom	Cat	BLOB
*	NULL	NULL	NULL	NULL

Open Value in Editor

Set Field(s) to NULL

A dialog appears and we can see the image in the Image tab:



If you want to retrieve the stored file programmatically, read the tutorial: [Read file data from database using JDBC](#).

NOTE: By default, MySQL restricts the size of data packet can be sent in a query to only 1 MB. So you may get an error if trying to upload a file bigger than this limit. To increase this size limit, set the `max_allowed_packet` variable in MySQL, as discussed in the tutorial [Insert file data into MySQL database using JDBC](#).

Practical Database Programming with Java - This book is for those who are beginner in Java database programming and want to catch up with the main database programming techniques with straightforward and easy to learn example code.



Set of 2 5w LED Bulbs

Buy Now

Only
Rs. 99

Do you want to be expert in Java programming? If you do, why not join our mailing list to get advices from the professionals everyday? Just click here: <http://newsletter.codejava.net> - It's FREE, Quick and Awesome!

Attachments:

 [FileUploadDatabase.zip](#) [Eclipse project] 676 kB