Creating Data Centric Applications with JSP Servlet

- April 21, 2014
- By Manoj Debnath
- Bio »
- Send Email »
- More Articles »

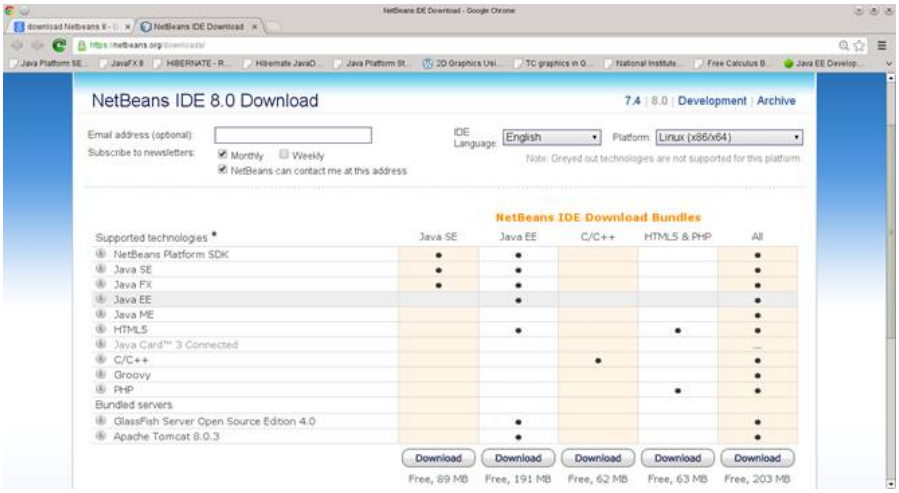
Servlets and Java Server Pages (JSP) are the highest level views of network programming in Java. Together they form the foundation of the request response model of communication. A servlet basically is a Java class that extends the functionality of a server, such as a Web Server that serves web pages to a user's browser using the HTTP protocol. JSP allows us to create pages that encapsulate Java functionality and even to write scriplets of actual Java code directly into the page. These features along with JDBC can be used effectively to create a data centric network application very easily.

Basic Requirements

The basic requirements of creating such an application are as follows:

- SDK: Java SDK
- IDE: NetBeans, Eclipse or any other IDE.
- Application Server: Tomcat, Glassfish, etc.
- Database package: MySQL, PostgreSQL, or Oracle, etc.
- Database driver such as JDBC Driver for MySQL, JDBC Driver for PostgreSQL, or JDBC Driver for Oracle.

Note: NetBeans version 8 IDE and its predecessor has inbuilt all the requirements as described above, except a database package such as MySQL. To start with this will be the best IDE, I believe. In this article we will be using Netbeans IDE, Tomcat Application Server and MySQL Database at the back end.

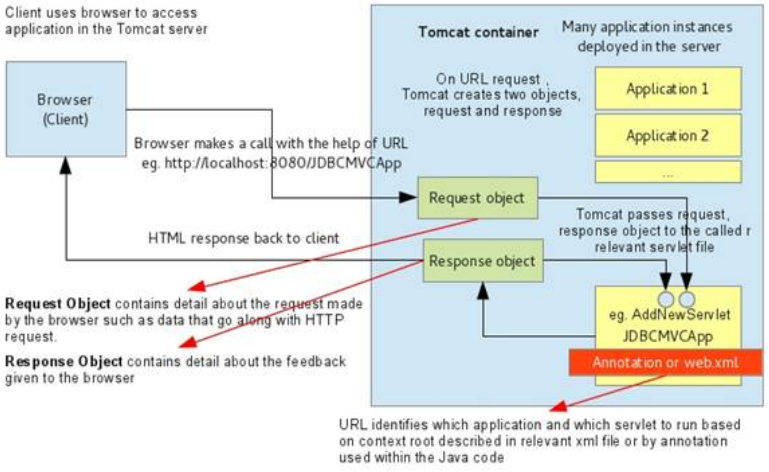


NetBeans IDE 8.0 Download

Web Programming Fundamentals

- Post a comment
- Email Article
- Print Article
- Share Articles
  - Digg
  - del.icio.us
  - Slashdot
  - DZone
  - Reddit
  - StumbleUpon
  - Facebook
  - FriendFeed
  - Furl
  - Newsvine
  - Google
  - LinkedIn
  - MySpace
  - Technorati
  - Twitter
  - YahooBuzz

When implementing the request response model of programming, it occurs between web browsers and web servers. When a user selects a web site to browse through the browser (the client application), a request is sent to the appropriate web server (the server application). The server normally responds to the client by sending the appropriate HTML web page. Let us view this from the perspective of a Tomcat Server.



Request Response Model

Tomcat refers to a servlet by the name given in the urlPatterns (if annotation is used rather than web.xml) and not by the servlet file name. The parameter urlPatterns is defined in the @WebServlet annotation as follows.

```
@WebServlet(name = "AddNewServlet", urlPatterns = {"/AddNewServlet"})
```

Related Articles

- Getting Started with Android Using Java
- Exception Handling in JNI
- Manipulating Java Objects in Native Code
- Desired Properties of a Big Data System
- JNI Data Type Mapping to C/C++

To write a servlet, we need to extend the HttpServlet class then use the annotation @WebServlet (See listing 3). The parameter, urlPatterns, is the key that tells Tomcat to execute the relevant classes whenever the requested string pattern matches with this urlPatterns. Observe that whenever we enter, http://localhost:8080, in the browser a Tomcat instance runs, as we add our project name such as http://localhost:8080/JDBCMVCApp, the first file mentioned in the web.xml runs. Let us create a small but complete data centric web application illustrating the use of servlet, JSP, JDBC using the Model View Controller (MVC) model of programming.

Our Application Design

The MVC model of programming grossly consists of four layers – Controller, Business Service, Model and View. Let's understand them from our project point of view.

- Controller
  - AddNewServlet, DeleteServlet, UpdateServlet
- Business Service
  - EmployeeBean
- Model
  - Employee
- View
  - empAddNew.jsp, empUpdate.jsp, empView.jsp

Controller, is a servlet in our case. When a request to add a new employee is initiated and the values for new employee are passed on to the Tomcat server, our controller servlet, AddNewServlet intercepts this request and collects the request parameter values for further processing. However, Controller servlet does not actually accesses the database. This is done through the Business Service class.

Business Service is a simple Java bean containing business logic and performs the actual processing function in the application. In our case, EmployeeBean contains all the CRUD logic for database access and provides service when called from the controller servlet.

Model contains the information and acts as an abstraction to the actual database schema. Obviously, this whole process needs some interface where the client can interact with the application – for example, forms for data collection, viewing, modifying, etc. We have used JSP to create all the view layer files, because, in JSP, along with HTML, we can write Java code seamlessly. This has a double advantage – one, the browser can easily render HTML and two, the programmer can write java code within HTML. As a result WYSIWYG HTML comes alive with dynamic content written in Java.

Java Codes

Let's start with the Model code in Employee.java:

Listing 1: Employee.java

```
package org.mano.model;

public class Employee {
    private int empId;
    private String empName;
    private String phone;
    private String email;
    private float salary;
    private String designation;

//...constructors, getters and setters

}
```

Then we write Business Service class: EmployeeBean.java. This class is the heart of our application. Observe the boilerplate code for database connection is repeated to retain simplicity. Once mastered the “way of the web programming in Java” :) the code can be further tuned to make it concise.

Listing 2: EmployeeBean.java

```
package org.mano.service;

//...import statements

public class EmployeeBean {

    private static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATABASE_URL = "jdbc:mysql://localhost/hr";
    private static final String USERNAME = "user1";
    private static final String PASSWORD = "secret";

    public void addNew(Employee e) {
        Connection con = null;
        PreparedStatement pstmt = null;
        try {
            Class.forName(JDBC_DRIVER);
            con = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);
            pstmt = con.prepareStatement("INSERT INTO emp(empId,empName,phone,email,salary,desig) VALUES(?,?,?,?,?,?)");
            pstmt.setInt(1, e.getEmpId());
            pstmt.setString(2, e.getName());
            pstmt.setString(3, e.getPhone());
            pstmt.setString(4, e.getEmail());
            pstmt.setFloat(5, e.getSalary());
            pstmt.setString(6, e.getDesignation());
            pstmt.execute();
        } catch (SQLException | ClassNotFoundException ex) {

        } finally {
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
                if (con != null) {
                    con.close();
                }
```

```
            } catch (SQLException ex) {
                Logger.getLogger(EmployeeBean.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

    public void delete(int id) {
        Connection con = null;
        Statement stmt = null;
        try {
            Class.forName(JDBC_DRIVER);
            con = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);
            stmt = con.createStatement();
            stmt.execute("DELETE FROM emp WHERE empId=" + String.valueOf(id));
        } catch (SQLException | ClassNotFoundException ex) {

        } finally {
            try {
                if (stmt != null) {
                    stmt.close();
                }
                if (con != null) {
                    con.close();
                }
            } catch (SQLException ex) {
                Logger.getLogger(EmployeeBean.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

    public void update(Employee e) {
        Connection con = null;
        PreparedStatement pstmt = null;
        try {
            Class.forName(JDBC_DRIVER);
            con = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);
            pstmt = con.prepareStatement("UPDATE emp SET empName=?, phone=?, email=?, salary=?, desig=? WHERE empId=?");
            pstmt.setString(1, e.getName());
            pstmt.setString(2, e.getPhone());
            pstmt.setString(3, e.getEmail());
            pstmt.setFloat(4, e.getSalary());
            pstmt.setString(5, e.getDesignation());
            pstmt.setInt(6, e.getEmpId());
            pstmt.executeUpdate();
        } catch (SQLException | ClassNotFoundException ex) {

        } finally {
            try {
                if (pstmt != null) {
                    pstmt.close();
                }
                if (con != null) {
                    con.close();
                }
            } catch (SQLException ex) {
                Logger.getLogger(EmployeeBean.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

    public Employee getEmployee(int id) {
        Employee emp = null;
        Connection con = null;
        Statement stmt = null;
        try {
            Class.forName(JDBC_DRIVER);
            con = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);
            stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM emp WHERE empId=" + id);
            if (rs.next()) {
                emp = new Employee();
                emp.setEmpId(rs.getInt(1));
                emp.setName(rs.getString(2));
                emp.setPhone(rs.getString(3));
                emp.setEmail(rs.getString(4));
                emp.setSalary(rs.getFloat(5));
                emp.setDesignation(rs.getString(6));
            }
        } catch (SQLException | ClassNotFoundException ex) {

        } finally {
            try {
                if (stmt != null) {
                    stmt.close();
                }
                if (con != null) {
                    con.close();
                }
            } catch (SQLException ex) {
                Logger.getLogger(EmployeeBean.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        return emp;
    }

    public List<Employee> getEmployees() {
        List<Employee> list = new ArrayList<>();
        Connection con = null;
        Statement stmt = null;
        try {
            Class.forName(JDBC_DRIVER);
            con = DriverManager.getConnection(DATABASE_URL, USERNAME, PASSWORD);
            stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM emp ORDER BY empId");
```

```
            while (rs.next()) {
                Employee emp = new Employee();
                emp.setEmpId(rs.getInt(1));
                emp.setName(rs.getString(2));
                emp.setPhone(rs.getString(3));
                emp.setEmail(rs.getString(4));
                emp.setSalary(rs.getFloat(5));
                emp.setDesignation(rs.getString(6));
                list.add(emp);
            }
        } catch (SQLException | ClassNotFoundException ex) {

        } finally {
            try {
                if (stmt != null) {
                    stmt.close();
                }
                if (con != null) {
                    con.close();
                }
            } catch (SQLException ex) {

            }
        }
        return list;
    }

}
```

Now, the controller classes: **AddNewServlet.java, UpdateServlet.java, DeleteServlet.java:**

Listing 3: AddNewServlet.java

```
package org.mano.controller;

//...import statements

@WebServlet(name = "AddNewServlet", urlPatterns = {"/AddNewServlet"})
public class AddNewServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        Employee emp = new Employee();
        emp.setEmpId(Integer.parseInt(request.getParameter("empId")));
        emp.setName(request.getParameter("empName"));
        emp.setPhone(request.getParameter("phone"));
        emp.setEmail(request.getParameter("email"));
        emp.setSalary(Float.parseFloat(request.getParameter("salary")));
        emp.setDesignation(request.getParameter("designation"));
        EmployeeBean eb = new EmployeeBean();
        eb.addNew(emp);
        response.sendRedirect("empView.jsp");

    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }
}
```

Listing 4: UpdateServlet.java

```
package org.mano.controller;

//...import statements

@WebServlet(name = "UpdateServlet", urlPatterns = {"/UpdateServlet"})
public class UpdateServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        Employee emp = new Employee();
        emp.setEmpId(Integer.parseInt(request.getParameter("empId")));
        emp.setName(request.getParameter("empName"));
        emp.setPhone(request.getParameter("phone"));
        emp.setEmail(request.getParameter("email"));
        emp.setSalary(Float.parseFloat(request.getParameter("salary")));
        emp.setDesignation(request.getParameter("designation"));
        EmployeeBean eb = new EmployeeBean();
        eb.update(emp);
        response.sendRedirect("empView.jsp");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
```

```
    }

       @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

     @Override
    public String getServletInfo() {
        return "Short description";
    }
}
```

Listing 5: DeleteServlet.java

```
package org.mano.controller;

//...import statements

@WebServlet(name = "DeleteServlet", urlPatterns = {"/DeleteServlet"})
public class DeleteServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        int id = Integer.parseInt(request.getParameter("delId"));
        EmployeeBean eb = new EmployeeBean();
        eb.delete(id);
        response.sendRedirect("empView.jsp");
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }
}
```

Now, the view JSP files: **empView.jsp, empAddNew.jsp, empUpdate.jsp**. These JSP files also include two additional files – one JSP file named **header.jsp,** which includes some demo header information and links and another, a cascading style sheet named **style.css** to configure HTML rendering in the browser.

Listing 6: header.jsp

```
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
    <center>
        <div id="mystyle" style="border: none;">
            <h1>JDBC, JSP, Servlet Tutorial</h1>
            <p><a href="http://www.developer.com">http://www.developer.com</a><br/>
                <b>Creating Data centric Application with JSP Servlet</b><br/>
                <%=new Date()%></br>  </br>
                <a href="empAddNew.jsp">Add New Employee</a> &NegativeThickSpace; |
                <a href="empView.jsp">View Employee</a>
            </p>
        </div>
    </center>
</body>
</html>
```

Listing 7: style.css

```
root {
    display: block;
}
body{
    font-family:"Lucida Grande", "Lucida Sans Unicode", Verdana, Arial, Helvetica, sans-serif;
    font-size:12px;
    background: bisque;
}

.spacer{clear:both; height:1px;}

table, tr, td{
    border-radius: 5px;
    border:solid 1px burlywood;
    padding: 5px;
    width: max-content;
    background-color: bisque;
```

```
}

.myform{
margin:0 auto;
width:400px;
padding:14px;
}


#mystyle{
border:solid 2px burlywood;
background: bisque;
border-radius: 5px;
}
#mystyle h1 {
font-size:14px;
font-weight:bold;
margin-bottom:8px;
}
#mystyle p{
font-size:11px;
color:#666666;
margin-bottom:20px;
border-bottom:solid 1px burlywood;
padding-bottom:10px;
}
#mystyle label{
display:block;
font-weight:bold;
text-align:right;
width:140px;
float:left;
}
#mystyle .small{
color:#666666;
display:block;
font-size:11px;
font-weight:normal;
text-align:right;
width:140px;
}
#mystyle input{
float:left;
font-size:12px;
padding:4px 2px;
border:solid 1px burlywood;
width:200px;
margin:2px 0 20px 10px;
border-radius: 5px;
}
#mystyle button{
clear:both;
margin-left:150px;
width:125px;
height:31px;
background:#666666;
text-align:center;
line-height:31px;
color:#FFFFFF;
font-size:11px;
font-weight:bold;
border-radius: 5px;
}
```

Listing 8: empAddNew.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link href="style.css" rel="stylesheet" type="text/css"/>
        <title>New Employee</title>
    </head>
    <body>
        <%@include file="header.jsp"%>
        <form method="post" action="AddNewServlet">
            <div id="mystyle" class="myform">
                <form id="form" name="form" action="AddNewEmployee" method="post">
                    <h1>Employee</h1>
                    <p>To add new Employee enter following information</p>
                    <label>Employee ID<span class="small">Enter Employee ID</span></label>
                    <input type="text" name="empId" id="empId" />
                    <label>Name<span class="small">Enter name</span></label>
                    <input type="text" name="empName" id="empName" />
                    <label>Phone<span class="small">Enter phone number</span></label>
                    <input type="text" name="phone" id="phone" />
                    <label>Email<span class="small">Enter email address</span></label>
                    <input type="text" name="email" id="email" />
                    <label>Salary<span class="small">Enter salary</span></label>
                    <input type="text" name="salary" id="salary" />
                    <label>Designation<span class="small">Enter designation</span></label>
                    <input type="text" name="designation" id="designation" />
                    <button type="submit">Add New Employee</button>
                    <div class="spacer"></div>
                </form>
            </div>
        </form>
    </body>
</html>
```

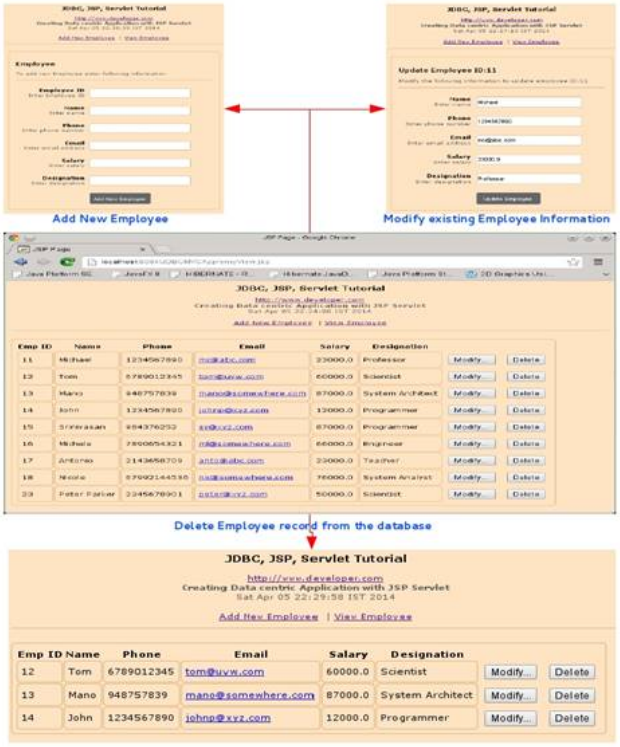
Listing 9: empUpdate.jsp

```
<%@page import="org.mano.model.Employee"%>
<%@page import="org.mano.service.EmployeeBean"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link href="style.css" rel="stylesheet" type="text/css"/>
        <title>Employee Update Page</title>
    </head>
    <body>
        <%@include file="header.jsp"%>
        <%
            int id = Integer.parseInt(request.getParameter("updateId"));
            EmployeeBean eb = new EmployeeBean();
            Employee e = eb.getEmployee(id);
        %>
        <div id="mystyle" class="myform">
            <form id="form" name="form" action="UpdateServlet" method="post">
                <h1>Update Employee ID:<%=e.getEmpId()%></h1>
                <p>Modify the following information to update employee ID:<%=e.getEmpId()%></p>
                <label><input type="hidden" name="empId" id="empId" value="<%=e.getEmpId()%>"/><span class="small"></span></label>
                <label>Name<span class="small">Enter name</span></label>
                <input type="text" name="empName" id="empName" value="<%=e.getName()%>"/>
                <label>Phone<span class="small">Enter phone number</span></label>
                <input type="text" name="phone" id="phone" value="<%=e.getPhone()%>"/>
                <label>Email<span class="small">Enter email address</span></label>
                <input type="text" name="email" id="email" value="<%=e.getEmail()%>"/>
                <label>Salary<span class="small">Enter salary</span></label>
                <input type="text" name="salary" id="salary" value="<%=e.getSalary()%>"/>
                <label>Designation<span class="small">Enter designation</span></label>
                <input type="text" name="designation" id="designation" value="<%=e.getDesignation()%>"/>
                <button type="submit">Update Employee</button>
                <div class="spacer"></div>
            </form>
        </div>
    </body>
</html>
```

Listing 10: empView.jsp

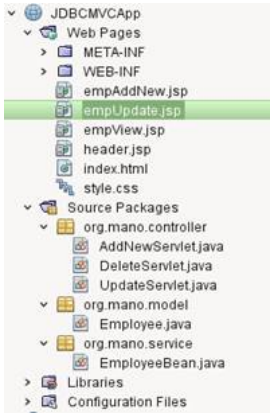
```
<%@page import="org.mano.service.EmployeeBean"%>
<%@page import="org.mano.model.Employee"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link href="style.css" rel="stylesheet" type="text/css"/>
        <title>JSP Page</title>
    </head>
    <body>
        <%@include file="header.jsp"%>
        <div>
            <table>
                <thead>
                    <tr>
                        <th>Emp ID</th>
                        <th>Name</th>
                        <th>Phone</th>
                        <th>Email</th>
                        <th>Salary</th>
                        <th>Designation</th>
                    </tr>
                </thead>
                <tbody>
                    <%
                        EmployeeBean eb = new EmployeeBean();
                        List<Employee> list = eb.getEmployees();
                        for (Employee e : list) {
                    %>

                    <tr>
                        <td><%=String.valueOf(e.getEmpId())%></td>
                        <td><%=e.getName()%></td>
                        <td><%=e.getPhone()%></td>
                        <td><a href="mailto:<%=e.getEmail()%>"><%=e.getEmail()%></a></td>
                        <td><%=String.valueOf(e.getSalary())%></td>
                        <td><%=e.getDesignation()%></td>
                        <td style="border: none;">
                            <div>
                                <form method="post" action="empUpdate.jsp">
                                    <input type="hidden" id="updateId" name="updateId" value="<%=String.valueOf(e.getEmpId())%>"/>
                                    <input type="submit" value="Modify..."/>
                                </form>
                            </div>
                        </td>
                        <td style="border: none;">
                            <div>
                                <form method="post" action="DeleteServlet">
                                    <input type="hidden" id="delId" name="delId" value="<%=String.valueOf(e.getEmpId())%>"/>
                                    <input type="submit" value="Delete"/>
                                </form>
                            </div>
                        </td>
```

```
                    </tr>
                    <%
                        }
                    %>
                </tbody>
            </table>
        </div>
    </body>
</html>
```



Configuring NetBeans

If you are unsure of how to create aweb application in Netbeans:



To create a web application in NetBeans do the following:

1. Open NetBeans, then select **File... → New Project.**
2. Select **Java Web** from the **Categories** list and **Web Application** from the **Projects** list. Click **Next.**
3. Provide the name of the Project and click **Next.**
4. Choose **Apache Tomcat Server** from the **Server** drop down list and click **Next**.
5. **Finish.**

Now, do not forget to include JDBC driver in the **Project → Libraries**. JDBC driver is not included by default in the project. As we have used MySQL JDBC Driver, to include it right click on the **Libraries →** choose **Add Library →** click on **import**... then select the appropriate JDBC driver from the list of available libraries and → click **Import Library**.

Creating a Sample Database for the Project

Create a sample table in the MySQL database for testing and realizing the above project. Open a MySQL prompt and type the following query:

CREATE TABLE emp (empId int, empName varchar(20), phone varchar(20), email varchar(20), salary float, desig varchar(20), primary key(empId));

Conclusion

Integrating JDBC, JSP and Servlet is pretty straight forward. This type of project is mainly used for creating multi-tiered online applications using request response mechanism. JSP technology is basically an extension of servlet technology and simplifies the process of creating pages by separating presentation form the content. JSP, Servlet and JDBC are a popular combination for creating dynamic web pages and applications with extensive data processing.

Tags: database, JDBC, JSP, Servlets, data processing

13 Comments (click to add your comment)

Comment Page: 1

2
By chaimaa June 25 2015 18:54 PDT

thank you so much for this tutorial , it was very helpfull =)

Reply to this commentBy lukas April 09 2015 08:34 PDT

could you please provide the code I will really appreciated. Thanks, Lukas

Reply to this commentBy R April 04 2015 20:15 PDT

on click of add new employee I am getting following exception: Pls help r 05, 2015 8:29:09 AM org.apache.catalina.startup.Catalina start INFO: Server startup in 4014 ms Apr 05, 2015 8:29:31 AM org.apache.catalina.core.StandardWrapperValve invoke SEVERE: Servlet.service() for servlet [AddNewServlet] in context with path [/Sample] threw exception java.lang.NumberFormatException: For input string: "nknkj" at sun.misc.FloatingDecimal.readJavaFormatString(FloatingDecimal.java:1241) at java.lang.Float.parseFloat(Float.java:452) at com.org.controller.AddNewServlet.processRequest(AddNewServlet.java:27) at com.org.controller.AddNewServlet.doPost(AddNewServlet.java:44) at javax.servlet.http.HttpServlet.service(HttpServlet.java:647) at javax.servlet.http.HttpServlet.service(HttpServlet.java:728) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:305) at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210) at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:243) at org.apache.catalina

Reply to this commentBy sokleng March 30 2015 20:43 PDT

Will u plz share the code for getters & setters??

Reply to this commentBy S K Rastogi March 01 2015 11:07 PST

Dear Sir, Thanks for such detailed example listings. Can JSP and servlet be used for developing mobile apps? If yes, can you please explain how to do it?

Reply to this commentBy Amit Soni December 11 2014 09:50 PST

Dear Sir, thnx for such a guiding example. I used this example as it is but it displays empty table without data. Same is happening to my project which I modified according to this example. Please help me with it Sir.

Reply to this commentBy Iqbal September 05 2014 09:34 PDT

Thank you so much :)

Reply to this commentBy Ovi September 03 2014 03:31 PDT

What will be its XML file ? plz share..

Reply to this commentBy thiru August 12 2014 06:04 PDT

Hi, This is really good example to learn to develop a basic applications with using sevlets and jsp's

Reply to this commentBy Saurabh August 08 2014 01:35 PDT

Will u plz share the code for getters & setters?? it showing some error in my project in netbeans.. & it showing error for execute() & executeQuery() methods too...

Reply to this comment**More comments ... Page:** 1
2

Comment and Contribute

Your name/nickname

Your email

Subject

(Maximum characters: 1200). You have 1200 characters left.

Type the text

Privacy & Terms

Submit Your Comment

Enterprise Development Update

Don't miss an article. Subscribe to our newsletter below.

Enter Email Address SIGN UP

Most Popular Developer Stories

- Today
- This Week
- All-Time

- 1 Using JDBC with MySQL, Getting Started
- 2 An Introduction to Java Annotations
- 3 MIDP Programming with J2ME
- 4 An Introduction to JSP Standard Template Library (JSTL)
- 5 Debugging a Java Program with Eclipse

- 1 Using JDBC with MySQL, Getting Started
- 2 An Introduction to Java Annotations
- 3 An Introduction to JSP Standard Template Library (JSTL)
- 4 MIDP Programming with J2ME
- 5 Debugging a Java Program with Eclipse

- 1 Using JDBC with MySQL, Getting Started
- 2 An Introduction to Java Annotations
- 3 An Introduction to JSP Standard Template Library (JSTL)
- 4 MIDP Programming with J2ME
- 5 Debugging a Java Program with Eclipse

Most Commented On

- This Week
- This Month
- All-Time

- 1 10 Experimental PHP Projects Pushing the Envelope
- 2 Day 1: Learning the Basics of PL/SQL
- 3 C# Tip: Placing Your C# Application in the System Tray
- 4 Logical Versus Physical Database Modeling
- 5 Is Ubuntu Contributing as Much as It Should to Free Software Projects?

- 1 Day 1: Learning the Basics of PL/SQL
- 2 The 5 Developer Certifications You'll Wish You Had in 2015
- 3 10 Experimental PHP Projects Pushing the Envelope
- 4 An Introduction to Struts
- 5 Inside Facebook's Open Source Infrastructure

- 1 Creating Use Case Diagrams
- 2 Day 1: Learning the Basics of PL/SQL
- 3 C# Tip: Placing Your C# Application in the System Tray
- 4 Using ASP.NET To Send Email
- 5 Using JDBC with MySQL, Getting Started

Recommended Partner Resources

- Cloud Computing Showcase for Developers
- Mobile Development Center
- HTML 5 Development Center