1 Write a java program using JDBC Connection technique to create table and perform insert operation using My sql database.

```
import java.sql.*;

public class JDBCInsertDemo {
   public static void main(String[] args) {
      String url = "jdbc:mysql://localhost:3306/testdb"; // Your DB
      String user = "root"; // Your MySQL username
      String pass = "password"; // Your MySQL password

      try (Connection conn = DriverManager.getConnection(url, user, pass);
           Statement stmt = conn.createStatement()) {

         // Create table
         stmt.execute("CREATE TABLE IF NOT EXISTS Employee (id INT AUTO_INCREMENT PRIMARY
KEY, name VARCHAR(50), role VARCHAR(50))");
         System.out.println("Table created.");

         // Insert data
         int rows = stmt.executeUpdate("INSERT INTO Employee (name, role) VALUES ('John Doe',
'Manager')");
         System.out.println(rows + " row(s) inserted.");

      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

Output –

Table created.

1 row(s) inserted.

2 Write a java program using JDBC Connection technique to create table and perform update operation using My sql database.

```
import java.sql.*;

public class JDBCUpdateDemo {
   public static void main(String[] args) {
      String url = "jdbc:mysql://localhost:3306/testdb"; // Your database name
      String user = "root"; // MySQL username
      String pass = "password"; // MySQL password

      try (Connection conn = DriverManager.getConnection(url, user, pass);
           Statement stmt = conn.createStatement()) {
```

```
        // Create table if not exists
        stmt.execute("CREATE TABLE IF NOT EXISTS Employee (id INT AUTO_INCREMENT PRIMARY
KEY, name VARCHAR(50), role VARCHAR(50))");
        System.out.println("Table ensured.");

        // Insert initial data
        stmt.executeUpdate("INSERT INTO Employee (name, role) VALUES ('Alice', 'Developer')");

        // Update operation
        int rows = stmt.executeUpdate("UPDATE Employee SET role='Senior Developer' WHERE
name='Alice'");
        System.out.println(rows + " row(s) updated.");

      } catch (SQLException e) {
        e.printStackTrace();
      }
   }
}
```

Output –

Table ensured.

1 row(s) updated.

3 Write a java program using JDBC Connection technique to create table and perform delete operation using My sql database.

```
import java.sql.*;

public class JDBCDeleteDemo {
   public static void main(String[] args) {
      String url = "jdbc:mysql://localhost:3306/testdb"; // Your DB name
      String user = "root"; // Your MySQL username
      String pass = "password"; // Your MySQL password

      try (Connection conn = DriverManager.getConnection(url, user, pass);
           Statement stmt = conn.createStatement()) {

        // Create table
        stmt.execute("CREATE TABLE IF NOT EXISTS Employee (id INT AUTO_INCREMENT PRIMARY
KEY, name VARCHAR(50), role VARCHAR(50))");
        System.out.println("Table ready.");

        // Insert sample data
        stmt.executeUpdate("INSERT INTO Employee (name, role) VALUES ('Bob', 'Analyst')");

        // Delete record
        int rows = stmt.executeUpdate("DELETE FROM Employee WHERE name='Bob'");
        System.out.println(rows + " row(s) deleted.");
```

```
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Output –

Table ready.

1 row(s) deleted.

4 Write a java program using JDBC Connection technique to create table and perform retrieve operation using My sql database.

```
import java.sql.*;

public class JDBCRetrieveDemo {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/testdb";  // your DB
        String user = "root";       // your MySQL username
        String pass = "password";   // your MySQL password

        try (Connection conn = DriverManager.getConnection(url, user, pass);
             Statement stmt = conn.createStatement()) {

            // Create table
            stmt.execute("CREATE TABLE IF NOT EXISTS Employee (id INT AUTO_INCREMENT PRIMARY
KEY, name VARCHAR(50), role VARCHAR(50))");
            System.out.println("Table ready.");

            // Insert sample data
            stmt.executeUpdate("INSERT INTO Employee (name, role) VALUES ('Charlie', 'HR'), ('Diana',
'Engineer')");

            // Retrieve data
            ResultSet rs = stmt.executeQuery("SELECT * FROM Employee");

            // Display data
            System.out.println("ID | Name     | Role");
            System.out.println("-------------------------");
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                String role = rs.getString("role");
                System.out.printf("%2d | %-8s | %-10s%n", id, name, role);
            }

        } catch (SQLException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Output –

Table ready.

ID | Name | Role

---

1 | Charlie | HR

2 | Diana | Engineer

5 Write a java program which demonstrate the difference between statement interface and prepared statement interface.

```
import java.sql.*;

public class StatementVsPreparedStatementDemo {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/testdb"; // Use your DB name
        String user = "root"; // Your username
        String pass = "password"; // Your password

        try (Connection conn = DriverManager.getConnection(url, user, pass)) {

            // Create table
            try (Statement stmt = conn.createStatement()) {
                stmt.execute("CREATE TABLE IF NOT EXISTS Users (id INT AUTO_INCREMENT PRIMARY
KEY, name VARCHAR(50), age INT)");
                System.out.println("Table created.");
            }

            // Insert using Statement (unsafe – prone to SQL injection)
            try (Statement stmt = conn.createStatement()) {
                String name = "John'; DROP TABLE Users; --";
                int age = 25;
                String unsafeSQL = "INSERT INTO Users (name, age) VALUES ('" + name + "', " + age + ")";
                stmt.executeUpdate(unsafeSQL);
                System.out.println("Inserted using Statement (Unsafe).");
            }

            // Insert using PreparedStatement (safe – prevents SQL injection)
            String safeSQL = "INSERT INTO Users (name, age) VALUES (?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(safeSQL)) {
                pstmt.setString(1, "Alice");
                pstmt.setInt(2, 30);
```

```
            pstmt.executeUpdate();
            System.out.println("Inserted using PreparedStatement (Safe).");
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

Otuutput –

Table created.

Inserted using Statement (Unsafe).

Inserted using PreparedStatement (Safe).

6 Write a java program to accept the data from command line arguments and insert the data in a table present in mysql database using JDBC.

```
import java.sql.*;

public class InsertFromArgs {
   public static void main(String[] args) {
      if (args.length != 2) {
         System.out.println("Usage: java InsertFromArgs <name> <age>");
         return;
      }

      String name = args[0];
      int age = Integer.parseInt(args[1]);

      String url = "jdbc:mysql://localhost:3306/testdb";
      String user = "root";
      String pass = "password";

      String sql = "INSERT INTO Users (name, age) VALUES (?, ?)";

      try (Connection conn = DriverManager.getConnection(url, user, pass);
           PreparedStatement pstmt = conn.prepareStatement(sql)) {

         pstmt.setString(1, name);
         pstmt.setInt(2, age);
         int rows = pstmt.executeUpdate();
         System.out.println(rows + " row(s) inserted.");

      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
```

```
}
```

Output –

javac InsertFromArgs.java

java InsertFromArgs "Bob" 28

1 row(s) inserted.

##

Assignment – 2

1 WAP to create a simple Web application using Servlet interface and print hello on browser.

HelloServlet.java

```
import java.io.*;
import javax.servlet.*;

public class HelloServlet implements Servlet {
   ServletConfig config;

   public void init(ServletConfig config) {
      this.config = config;
   }

   public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      out.println("<h1>Hello from Servlet Interface</h1>");
   }

   public void destroy() {}
   public ServletConfig getServletConfig() { return config; }
   public String getServletInfo() { return "Basic Servlet Implementation"; }
}
```

web.xml

```
<web-app>
   <servlet>
      <servlet-name>HelloServlet</servlet-name>
      <servlet-class>HelloServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>HelloServlet</servlet-name>
      <url-pattern>/helloServlet</url-pattern>
   </servlet-mapping>
</web-app>
```

Output on browser:

Hello from Servlet Interface

2 WAP to create a simple Web application using Generic Servlet class and print hello on browser.

```
import java.io.*;
import javax.servlet.*;

public class HelloGenericServlet extends GenericServlet {
   public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      out.println("<h1>Hello from GenericServlet</h1>");
   }
}
```

```
<web-app>
   <servlet>
      <servlet-name>HelloGeneric</servlet-name>
      <servlet-class>HelloGenericServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>HelloGeneric</servlet-name>
      <url-pattern>/helloGeneric</url-pattern>
   </servlet-mapping>
</web-app>
```

Output –

Hello from GenericServlet

3 WAP to create a simple Web application using HttpServlet class and print hello on browser.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloHttpServlet extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      out.println("<h1>Hello from HttpServlet</h1>");
   }
}
```

```
<web-app>
   <servlet>
      <servlet-name>HelloHttp</servlet-name>
      <servlet-class>HelloHttpServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>HelloHttp</servlet-name>
      <url-pattern>/helloHttp</url-pattern>
   </servlet-mapping>
</web-app>
```

Output –

Hello from HttpServlet

4 Implement a JAVA Servlet Program to implement a dynamic HTML using Servlet (user name and password should be accepted using HTML and displayed using a Servlet)

Index.html

```
<!DOCTYPE html>
<html>
<head><title>Login Form</title></head>
<body>
   <form action="LoginServlet" method="post">
      Username: <input type="text" name="username" /><br/>
      Password: <input type="password" name="password" /><br/>
      <input type="submit" value="Login" />
   </form>
</body>
</html>
```

Java file –

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginServlet extends HttpServlet {
   protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();

      String username = req.getParameter("username");
      String password = req.getParameter("password");

      out.println("<h2>Welcome, " + username + "</h2>");
      out.println("<p>Your password is: " + password + "</p>");
   }
}
```

Web.xml

```
<web-app>
   <servlet>
      <servlet-name>LoginServlet</servlet-name>
      <servlet-class>LoginServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>LoginServlet</servlet-name>
      <url-pattern>/LoginServlet</url-pattern>
   </servlet-mapping>
</web-app>
```

Output-

Welcome, Alice

Your password is: secret123

5 WAP to create a simple Web application using HttpServlet and demonstrate the concept of Get and Post request method.

Form.html

```
<!DOCTYPE html>
<html>
<head><title>GET vs POST</title></head>
<body>
```

```
  <form action="GetPostServlet" method="get">
    <input type="submit" value="Send GET Request" />
  </form>
  <form action="GetPostServlet" method="post">
    <input type="submit" value="Send POST Request" />
  </form>
</body>
</html>
```

GetPostServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class GetPostServlet extends HttpServlet {
  protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
    res.setContentType("text/html");
    res.getWriter().println("<h2>This is a GET request</h2>");
  }

  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
    res.setContentType("text/html");
    res.getWriter().println("<h2>This is a POST request</h2>");
  }
}
```

Web.xml

```
<web-app>
  <servlet>
    <servlet-name>GetPostServlet</servlet-name>
    <servlet-class>GetPostServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>GetPostServlet</servlet-name>
    <url-pattern>/GetPostServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Output on Browser:

- **Click GET button** → This is a GET request
- **Click POST button** → This is a POST request

6 WAP to create a simple Web application using HttpServlet and demonstrate the concept of forward and include method of RequestDispacter.

MainServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MainServlet extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
      String action = req.getParameter("action");
      if ("forward".equals(action)) {
         RequestDispatcher rd = req.getRequestDispatcher("TargetServlet");
         rd.forward(req, res);
      } else if ("include".equals(action)) {
         PrintWriter out = res.getWriter();
         out.println("<h3>Before Include</h3>");
         RequestDispatcher rd = req.getRequestDispatcher("TargetServlet");
         rd.include(req, res);
         out.println("<h3>After Include</h3>");
      } else {
         res.getWriter().println("Specify ?action=forward or ?action=include");
      }
   }
}
```

TargetServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class TargetServlet extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
      res.getWriter().println("<p>This is the TargetServlet</p>");
   }
}
```

Web.xml

```
<web-app>
```

```
  <servlet>
    <servlet-name>MainServlet</servlet-name>
    <servlet-class>MainServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MainServlet</servlet-name>
    <url-pattern>/MainServlet</url-pattern>
  </servlet-mapping>

  <servlet>
    <servlet-name>TargetServlet</servlet-name>
    <servlet-class>TargetServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>TargetServlet</servlet-name>
    <url-pattern>/TargetServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

Output:

- Visit /MainServlet?action=forward

This is the TargetServlet

- Visit /MainServlet?action=include

Before Include

This is the TargetServlet

After Include

7 Create Servlet for login page, if the username and password is correct then prints message "Hello username" else a message"login failed".

login.html (HTML Form)

```
<!DOCTYPE html>
<html>
<head><title>Login Page</title></head>
<body>
  <form action="LoginCheckServlet" method="post">
    Username: <input type="text" name="username" /><br/>
    Password: <input type="password" name="password" /><br/>
    <input type="submit" value="Login" />
  </form>
</body>
```

```
</html>
```

LoginCheckServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginCheckServlet extends HttpServlet {
   protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();

      // Retrieve username and password from form
      String username = req.getParameter("username");
      String password = req.getParameter("password");

      // Simple validation logic
      if ("admin".equals(username) && "1234".equals(password)) {
         out.println("<h2>Hello, " + username + "!</h2>");
      } else {
         out.println("<h2>Login failed. Invalid username or password.</h2>");
      }
   }
}
```

Web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
   <servlet>
      <servlet-name>LoginCheckServlet</servlet-name>
      <servlet-class>LoginCheckServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>LoginCheckServlet</servlet-name>
      <url-pattern>/LoginCheckServlet</url-pattern>
   </servlet-mapping>
</web-app>
```

Output on Browser:

- Input Username: admin, Password: 1234 → **Hello, admin!**
- Input Username: bob, Password: 1111 → **Login failed. Invalid username or password.**

8 Create Servlet that uses cookies to store the number of times a user has visited the servlet.

VisitCounterServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class VisitCounterServlet extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();

      // Default visit count is 1
      int visitCount = 1;
      Cookie[] cookies = req.getCookies();

      // If the cookie exists, get the visit count value and increment
      if (cookies != null) {
         for (Cookie c : cookies) {
            if (c.getName().equals("visitCount")) {
               visitCount = Integer.parseInt(c.getValue()) + 1;
            }
         }
      }

      // Create new cookie for visit count
      Cookie visitCookie = new Cookie("visitCount", String.valueOf(visitCount));
      visitCookie.setMaxAge(60 * 60 * 24); // 1 day expiry time
      res.addCookie(visitCookie);

      // Display the visit count
      out.println("<h2>You have visited this page " + visitCount + " time(s).</h2>");
   }
}
```

Web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
   <servlet>
      <servlet-name>VisitCounterServlet</servlet-name>
      <servlet-class>VisitCounterServlet</servlet-class>
   </servlet>
```

```
    <servlet-mapping>
        <servlet-name>VisitCounterServlet</servlet-name>
        <url-pattern>/VisitCounterServlet</url-pattern>
    </servlet-mapping>
</web-app>
```

Output on Browser:

- **First visit:** You have visited this page 1 time(s).
- **Second visit:** You have visited this page 2 time(s).
- **Third visit:** You have visited this page 3 time(s).

##

Assignment – 3

1 Write a Java program to implement a simple Client Server Application using RMI.

Step 1: Create the Remote Interface (CalculatorInterface.java)

java

CopyEdit

```
import java.rmi.*;

public interface CalculatorInterface extends Remote {
    // Remote method for adding two numbers
    public int add(int a, int b) throws RemoteException;
}
```

The remote interface defines the method add which takes two integers and returns an integer. This method will be invoked remotely.

Step 2: Implement the Remote Interface (CalculatorImpl.java)

java

CopyEdit

```
import java.rmi.*;
import java.rmi.server.*;

public class CalculatorImpl extends UnicastRemoteObject implements CalculatorInterface {

    public CalculatorImpl() throws RemoteException {
        super();
    }

    // Implementation of the add method
    public int add(int a, int b) throws RemoteException {
        return a + b;
    }
}
```

The CalculatorImpl class extends UnicastRemoteObject (which makes the object remotely accessible) and implements the CalculatorInterface interface.

📄 Step 3: Create the Server Program (CalculatorServer.java)

java

CopyEdit

```
import java.rmi.*;
import java.rmi.registry.*;

public class CalculatorServer {
    public static void main(String[] args) {
        try {
            // Create the object of CalculatorImpl
```

```
            CalculatorImpl calc = new CalculatorImpl();

            // Bind the object to the RMI registry
            Naming.rebind("rmi://localhost:5000/CalculatorService", calc);

            System.out.println("Calculator Server is running...");
        } catch (Exception e) {
            System.out.println("Server exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

This server program binds the CalculatorImpl object to the RMI registry, making it available for remote clients.

📄 Step 4: Create the Client Program (CalculatorClient.java)

java

CopyEdit

```
import java.rmi.*;

public class CalculatorClient {
    public static void main(String[] args) {
        try {
            // Lookup the remote object in the RMI registry
            CalculatorInterface calc = (CalculatorInterface)
Naming.lookup("rmi://localhost:5000/CalculatorService");

            // Call the remote method
            int result = calc.add(10, 20);
```

```
            System.out.println("The result of 10 + 20 is: " + result);
        } catch (Exception e) {
            System.out.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

💻 **Expected Output:**

- **On Server Terminal:**

arduino

CopyEdit

Calculator Server is running...

- **On Client Terminal:**

csharp

CopyEdit

The result of 10 + 20 is: 30

==========

2 Design a Java JSP Program to implement verification of a particular user login and display a welcome page.

Step 1: Design the Login Form (login.jsp)

jsp

CopyEdit

<!DOCTYPE html>

<html>

<head>

```
    <title>User Login</title>
</head>
<body>
    <h2>Login Page</h2>
    <form action="loginServlet" method="post">
        Username: <input type="text" name="username" /><br/><br/>
        Password: <input type="password" name="password" /><br/><br/>
        <input type="submit" value="Login" />
    </form>
</body>
</html>
```

The form sends the entered username and password to loginServlet.

Step 2: Create the Login Servlet (LoginServlet.java)

java

CopyEdit

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Hardcoded user credentials
        if ("admin".equals(username) && "password123".equals(password)) {
            // Redirect to welcome page on successful login
            response.sendRedirect("welcome.jsp");
```

```
        } else {
            // Login failed, show message
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<h3>Login failed. Please check your username and password.</h3>");
        }
    }
}
```

This servlet performs simple validation. If the username is admin and the password is password123, the user is redirected to the welcome.jsp page. Otherwise, it shows a "login failed" message.

Step 3: Create the Welcome Page (welcome.jsp)

jsp

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
    <title>Welcome</title>
</head>
<body>
    <h2>Welcome, Admin!</h2>
    <p>You have successfully logged in.</p>
</body>
</html>
```

This page is shown after a successful login.

Step 4: Create web.xml (Servlet Mapping)

xml

CopyEdit

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
  <servlet>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>LoginServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/loginServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

This file maps the servlet LoginServlet to the URL /loginServlet.

💻 **Expected Output:**

1. **Login Page (login.jsp)**:
    - Input Username: admin, Password: password123 → Redirects to **welcome.jsp**.
2. **Welcome Page (welcome.jsp)**:

pgsql

CopyEdit

Welcome, Admin!

You have successfully logged in.

3. **Login Failed**:
    - If the wrong credentials are entered, the message **"Login failed. Please check your username and password."** will appear.

==========

3 Design and implement a Java JSP Program to get student information through a HTML and create a JAVA Bean Class, populate Bean and display the same information through another JSP.

Step 1: Create the HTML Form (studentForm.jsp)

This form will collect student information.

jsp

CopyEdit

```
<!DOCTYPE html>
<html>
<head>
   <title>Student Information Form</title>
</head>
<body>
   <h2>Enter Student Information</h2>
   <form action="saveStudent.jsp" method="post">
      Name: <input type="text" name="name" /><br/><br/>
      Age: <input type="text" name="age" /><br/><br/>
      Roll No: <input type="text" name="rollNo" /><br/><br/>
      <input type="submit" value="Submit" />
   </form>
</body>
</html>
```

Step 2: Create the Java Bean (Student.java)

This class will represent the student data.

java

CopyEdit

```
public class Student {
   private String name;
```

```
    private int age;
    private String rollNo;

    // Getter and Setter methods
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getRollNo() {
        return rollNo;
    }

    public void setRollNo(String rollNo) {
        this.rollNo = rollNo;
    }
}
```

Step 3: Process the Form Data and Set Student Data (saveStudent.jsp)

In this page, we will populate the Java Bean with data from the HTML form.

jsp

CopyEdit

```
<%@ page import="yourPackage.Student" %>
<%@ page contentType="text/html; charset=ISO-8859-1" language="java" %>
<%
   // Get data from form
   String name = request.getParameter("name");
   String ageStr = request.getParameter("age");
   String rollNo = request.getParameter("rollNo");

   // Create Student Bean and populate with data
   Student student = new Student();
   student.setName(name);
   student.setAge(Integer.parseInt(ageStr));
   student.setRollNo(rollNo);

   // Set the bean in the request scope
   request.setAttribute("student", student);
%>

<!-- Redirect to displayStudent.jsp to show the populated student data -->
<jsp:forward page="displayStudent.jsp"/>
```

Here, we retrieve the data from the form, create a Student object (Java Bean), and populate it with the data. Then, we forward the data to the next JSP page (displayStudent.jsp).

Step 4: Display the Student Information (displayStudent.jsp)

This page will display the information of the student that was passed to the request scope.

jsp

CopyEdit

```
<%@ page import="yourPackage.Student" %>
<%@ page contentType="text/html; charset=ISO-8859-1" language="java" %>
<%
   // Get the Student object from the request scope
   Student student = (Student) request.getAttribute("student");
%>

<!DOCTYPE html>
<html>
<head>
   <title>Student Information</title>
</head>
<body>
   <h2>Student Information</h2>
   <p>Name: <%= student.getName() %></p>
   <p>Age: <%= student.getAge() %></p>
   <p>Roll No: <%= student.getRollNo() %></p>
</body>
</html>
```

In this page, we retrieve the Student object from the request and display its properties (name, age, and rollNo).

💻 **Expected Output:**

1. **studentForm.jsp** (Input Form):
    - A simple form to input student details like name, age, and roll number.
2. **displayStudent.jsp** (Output Page):
    - After submitting the form, the student details are displayed:

yaml

CopyEdit

Name: John Doe

Age: 20

Roll No: 12345

==========

4 Develop a sample web application using MVC design pattern in Struts framework.

Step 1: Create the Action Class (HelloWorldAction.java)

The action class handles the request from the view and performs the necessary business logic.

java

CopyEdit

```
package com.example.struts;

import com.opensymphony.xwork2.ActionSupport;

public class HelloWorldAction extends ActionSupport {
   private String name;

   // Getter and Setter
   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   // Action method that will be called
   public String execute() {
```

```
        if (name == null || name.isEmpty()) {
            return ERROR;
        }
        return SUCCESS;
    }
}
```

Here, the HelloWorldAction class extends ActionSupport, which provides some built-in methods. The execute method handles the request and determines the outcome (SUCCESS or ERROR).

Step 2: Create the JSP View (helloWorld.jsp)

This is the view that will display the form and output.

jsp

CopyEdit

```
<!DOCTYPE html>
<html>
<head><title>Hello World</title></head>
<body>
    <h2>Enter your name:</h2>
    <form action="helloWorld" method="post">
        Name: <input type="text" name="name" /><br/><br/>
        <input type="submit" value="Submit" />
    </form>
</body>
</html>
```

This JSP page provides a form for the user to input their name.

Step 3: Create the Success View (success.jsp)

This view will be shown when the action is successful.

jsp

CopyEdit

```
<!DOCTYPE html>
<html>
<head><title>Success</title></head>
<body>
  <h2>Welcome, <%= request.getAttribute("name") %>!</h2>
</body>
</html>
```

Step 4: Create the Struts Configuration (struts.xml)

This file configures the action mappings.

xml

CopyEdit

```
<struts>
  <package name="default" extends="struts-default">
    <action name="helloWorld" class="com.example.struts.HelloWorldAction">
      <result name="success">/success.jsp</result>
      <result name="error">/helloWorld.jsp</result>
    </action>
  </package>
</struts>
```

This XML file configures the helloWorld action to use the HelloWorldAction class, with different views for success and error.

💻 **Expected Output:**

1. **On Initial Page (helloWorld.jsp)**:
    - The user inputs their name and submits the form.
2. **On Success (success.jsp)**:
    - Displays the message "Welcome, [name]!" after submitting the form.