



**UNIVERSITY MALAYSIA TERENGGANU**  
**FACULTY OF COMPUTER SCIENCE & MATHEMATICS**

**Web Based Application Development**  
**CSM 3023**

**LAB REPORT 8:**  
**JSP: Perform Create, Update, Retrieve and Delete (CRUD)**

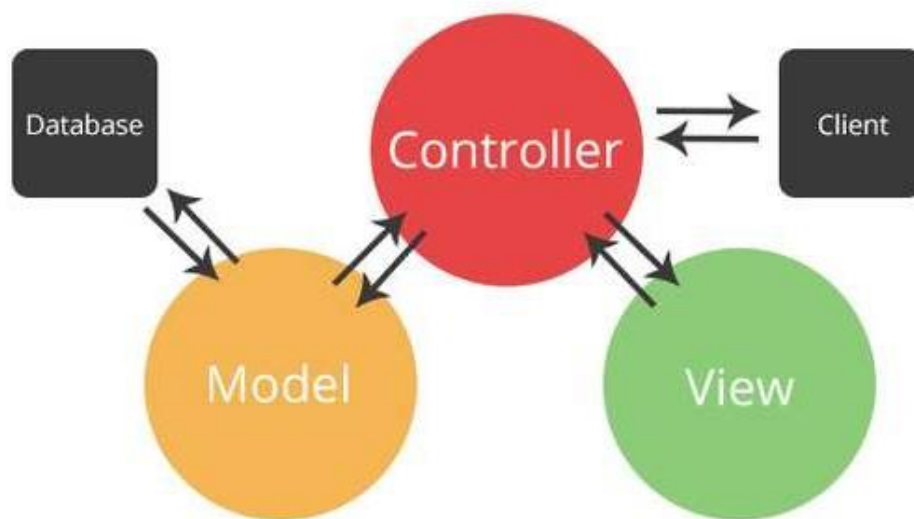
Prepared by:  
**NUR ARIFAH BINTI MOHD HANAFIAH**  
**S66428**

Prepared for:  
**DR MOHAMAD NOR BIN HASSAN**

**BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH HONORS**  
**SEMESTER II 2023/2024**

## LAB 8

# An MVC Example with Servlets and JSP



# Table of Contents

## Arahan:

Manual makmal ini adalah untuk kegunaan pelajar-pelajar Fakulti Teknologi Kejuruteraan Kelautan dan Informatik - FTKKI, Universiti Malaysia Terengganu (UMT) sahaja. Tidak dibenarkan mencetak dan mengedar manual ini tanpa kebenaran rasmi daripada penulis.

Sila ikuti langkah demi langkah sebagaimana yang dinyatakan di dalam manual. Tandakan (✓) setiap langkah yang telah selesai dibuat dan tulis kesimpulan bagi setiap aktiviti yang telah selesai dijalankan.

## Instruction:

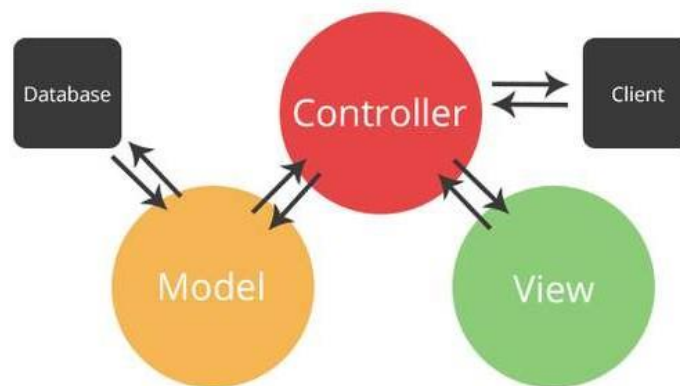
This laboratory manual is for use by the students of the Faculty of Ocean Engineering Technology and Informatics (FTKKI), Universiti Malaysia Terengganu only.

It is not permissible to print and distribute this manual without the official authorisation of the author. Please follow step by step as described in the manual. Tick (✓) each step completed and write the conclusions for each completed activity.

# Creating MVC Database Web Application in JSP and Servlets – for Create, Read, Update, Delete

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

- Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
- View - View represents the visualization of the data that model contains.
- Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.



## Benefits of MVC in JSP and Servlet Web Application

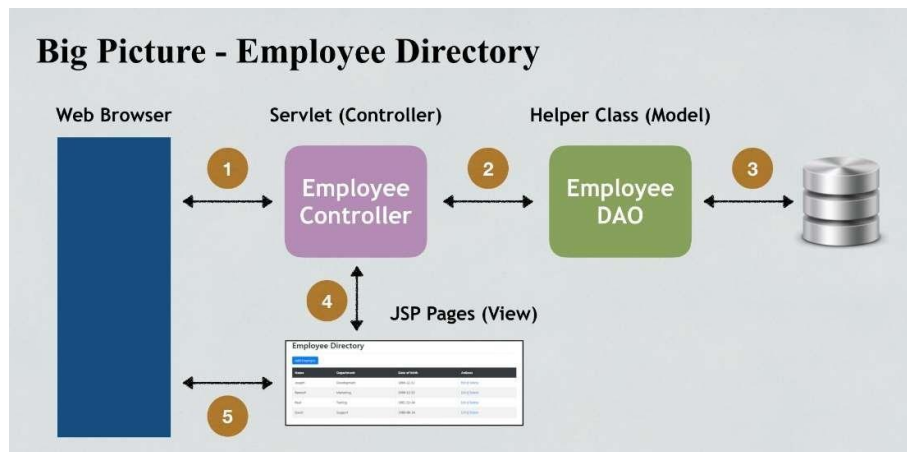
- Minimizes HTML code in Servlet no more: `out.println(...)` in Servlet code.
- Minimize Java business logic in JSPs no more large scriptlets in JSP code
- It separates the presentation layer from the business layer
- The Controller performs the action of invoking the Model and sending data to View
- The Model is not even aware that it is used by some web application or a desktop application

In this Lab8 activity, student will follow step by step to create a MVC application using JSP, Servlet and MySQL to create, read, update, and delete (CRUD) the student records into the database.

## Steps Involved in the Application

Basically, there are 4 main steps involved in this application that are given below:

- Capture the employee records and store it into the database.
- Fetch the employee records from the database and display it on the JSP.
- Update the existing employee records into the database.
- Delete the employee records from the database.



Step by step of Application development:

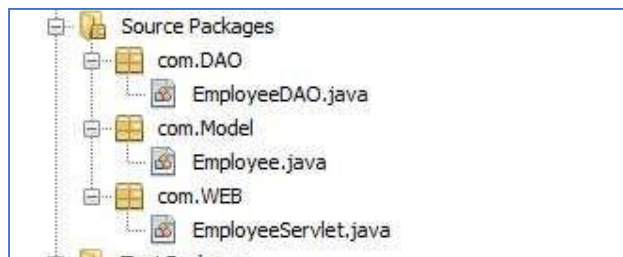
**Step 1** - Create table employees in COMPANY database schema.

```

1 CREATE DATABASE IF NOT EXISTS Company;
2 USE Company;
3
4 CREATE TABLE IF NOT EXISTS employees (
5     id INT NOT NULL AUTO_INCREMENT,
6     Name VARCHAR(60),
7     Email VARCHAR(50),
8     Position VARCHAR(15),
9     PRIMARY KEY (id)
10 )
  
```

**Step 2** - Create new web application project, named as Employee\_Management.

**Step 3** - Create three Java class that representing :



- EmployeeDAO.java (act as a Data Access Object (DAO) and to open /close database connection),
- Employee.java (act as a JavaBeans to represent business object), and
- EmployeeServlet.java (act to perform CRUD process)

## EmployeeDAO.java

Name the package as com.DAO

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package com.DAO;
7
8   import java.sql.Connection;
9   import java.sql.DriverManager;
10  import java.sql.PreparedStatement;
11  import java.sql.ResultSet;
12  import java.sql.SQLException;
13  import java.util.ArrayList;
14  import java.util.List;
15
16  import com.Model.Employee;
17
18  public class EmployeeDAO {
19      Connection connection = null;
20      private String jdbcURL = "jdbc:mysql://localhost:3306/company";
21      private String jdbcUsername = "yusro";
22      private String jdbcPassword = "admin";
23
24      private static final String INSERT_EMPLOYEES_SQL = "INSERT INTO employees (name, email, position) VALUES " +
25          " ( ?, ?, ?)";
26
27      private static final String SELECT_EMPLOYEE_BY_ID = "select id,name,email,position from employees where id=?";
28      private static final String SELECT_ALL_EMPLOYEES = "select * from employees";
29      private static final String DELETE_EMPLOYEES_SQL = "delete from employees where id = ?";
30
31      private static final String UPDATE_EMPLOYEES_SQL = "update employees set name = ?,email= ?, position=? where id = ?";
32
33      public EmployeeDAO() {}
34
35      protected Connection getConnection() {
36          Connection connection = null;
37          try {
38              Class.forName("com.mysql.jdbc.Driver");
39              connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
40          } catch (SQLException e) {
41              // TODO Auto-generated catch block
42              e.printStackTrace();
43          } catch (ClassNotFoundException e) {
44              // TODO Auto-generated catch block
45              e.printStackTrace();
46          }
47          return connection;
48      }
49
50      public void insertEmployee(Employee employee) throws SQLException {
51          System.out.println(INSERT_EMPLOYEES_SQL);
52          // try-with-resource statement will auto close the connection.
53          try (Connection connection = getConnection(); PreparedStatement preparedStatement =
54              connection.prepareStatement(INSERT_EMPLOYEES_SQL)) {
55              preparedStatement.setString(1, employee.getName());
56              preparedStatement.setString(2, employee.getEmail());
57              preparedStatement.setString(3, employee.getPosition());
58              System.out.println(preparedStatement);
59              preparedStatement.executeUpdate();
60          } catch (SQLException e) {
61              printSQLException(e);
62          }
63      }
64
65      public Employee selectEmployee(int id) {
66          Employee employee = null;
67          // Step 1: Establishing a Connection
68          try (Connection connection = getConnection();
69              // Step 2: Create a statement using connection object
70              PreparedStatement preparedStatement = connection.prepareStatement(SELECT_EMPLOYEE_BY_ID);) {
71              preparedStatement.setInt(1, id);
72              System.out.println(preparedStatement);
73              // Step 3: Execute the query or update query
74              ResultSet rs = preparedStatement.executeQuery();
75
76              // Step 4: Process the ResultSet object.
77              while (rs.next()) {
78                  String name = rs.getString("name");
79                  String email = rs.getString("email");
80                  String position = rs.getString("position");
81                  employee = new Employee(id, name, email, position);
82              }
83          } catch (SQLException e) {
84              printSQLException(e);
85          }
86          return employee;
87      }
88  }
```

```

87
88 public List < Employee > selectAllEmployees() {
89
90     // using try-with-resources to avoid closing resources (boiler plate code)
91     List < Employee > employees = new ArrayList < > ();
92     // Step 1: Establishing a Connection
93     try (Connection connection = getConnection());
94
95         // Step 2: Create a statement using connection object
96         PreparedStatement preparedStatement =
97             connection.prepareStatement(SELECT_ALL_EMPLOYEES);) {
98         System.out.println(preparedStatement);
99         // Step 3: Execute the query or update query
100         ResultSet rs = preparedStatement.executeQuery();
101
102         // Step 4: Process the ResultSet object.
103         while (rs.next()) {
104             int id = rs.getInt("id");
105             String name = rs.getString("name");
106             String email = rs.getString("email");
107             String position = rs.getString("position");
108             employees.add(new Employee(id, name, email, position));
109         }
110     } catch (SQLException e) {
111         printSQLException(e);
112     }
113     return employees;
114 }

```

```

116 public boolean deleteEmployee(int id) throws SQLException {
117     boolean rowDeleted;
118     try (Connection connection = getConnection(); PreparedStatement statement =
119         connection.prepareStatement(DELETE_EMPLOYEES_SQL);) {
120         statement.setInt(1, id);
121         rowDeleted = statement.executeUpdate() > 0;
122     }
123     return rowDeleted;
124 }
125
126 public boolean updateEmployee(Employee employee) throws SQLException {
127     boolean rowUpdated;
128     try (Connection connection = getConnection(); PreparedStatement statement =
129         connection.prepareStatement(UPDATE_EMPLOYEES_SQL);) {
130         statement.setString(1, employee.getName());
131         statement.setString(2, employee.getEmail());
132         statement.setString(3, employee.getPosition());
133         statement.setInt(4, employee.getId());
134
135         rowUpdated = statement.executeUpdate() > 0;
136     }
137     return rowUpdated;
138 }
139

```

```

139
140 private void printSQLException(SQLException ex) {
141     for (Throwable e: ex) {
142         if (e instanceof SQLException) {
143             e.printStackTrace(System.err);
144             System.err.println("SQLState: " + ((SQLException) e).getSQLState());
145             System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
146             System.err.println("Message: " + e.getMessage());
147             Throwable t = ex.getCause();
148             while (t != null) {
149                 System.out.println("Cause: " + t);
150                 t = t.getCause();
151             }
152         }
153     }
154 }
155
156

```

## Employee.java

Name the package as com.Model

```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package com.Model;
7
8  public class Employee {
9      protected int id;
10     protected String name;
11     protected String email;
12     protected String position;
13
14     public Employee() {}
15
16     public Employee(String name, String email, String position) {
17         super();
18         this.name = name;
19         this.email = email;
20         this.position = position;
21     }
22
23     public Employee(int id, String name, String email, String position) {
24         super();
25         this.id = id;
26         this.name = name;
27         this.email = email;
28         this.position = position;
29     }
30

```

```

30
31     public int getId() {
32         return id;
33     }
34     public void setId(int id) {
35         this.id = id;
36     }
37     public String getName() {
38         return name;
39     }
40     public void setName(String name) {
41         this.name = name;
42     }
43     public String getEmail() {
44         return email;
45     }
46     public void setEmail(String email) {
47         this.email = email;
48     }
49     public String getPosition() {
50         return position;
51     }
52     public void setPosition(String position) {
53         this.position = position;
54     }
55 }
56

```

## EmployeeServlet.java

Name the package as com.WEB



```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package com.WEB;
7
8   import java.io.IOException;
9   import java.sql.SQLException;
10  import java.util.List;
11
12  import javax.servlet.RequestDispatcher;
13  import javax.servlet.ServletException;
14  import javax.servlet.annotation.WebServlet;
15  import javax.servlet.http.HttpServlet;
16  import javax.servlet.http.HttpServletRequest;
17  import javax.servlet.http.HttpServletResponse;
18
19  import com.DAO.EmployeeDAO;
20  import com.Model.Employee;
21
22  @WebServlet("/")
23  public class EmployeeServlet extends HttpServlet {
24      // private static final long serialVersionUID = 1 L;
25      private EmployeeDAO employeeDAO;
26
27      public void init() {
28          employeeDAO = new EmployeeDAO();
29      }
30  }
```

```
31  protected void doPost(HttpServletRequest request, HttpServletResponse response)
32      throws ServletException, IOException {
33      doGet(request, response);
34  }
35
36  protected void doGet(HttpServletRequest request, HttpServletResponse response)
37      throws ServletException, IOException {
38      String action = request.getServletPath();
39
40      try {
41          switch (action) {
42              case "/new":
43                  showNewForm(request, response);
44                  break;
45              case "/insert":
46                  insertEmployee(request, response);
47                  break;
48              case "/delete":
49                  deleteEmployee(request, response);
50                  break;
51              case "/edit":
52                  showEditForm(request, response);
53                  break;
54              case "/update":
55                  updateEmployee(request, response);
56                  break;
57              default:
58                  listEmployee(request, response);
59                  break;
60          }
61      }
```

```
62      } catch (SQLException ex) {
63          throw new ServletException(ex);
64      }
65  }
66
67  private void listEmployee(HttpServletRequest request, HttpServletResponse response)
68      throws SQLException, ServletException, IOException {
69      List<Employee> listEmployee = employeeDAO.selectAllEmployees();
70      request.setAttribute("listEmployee", listEmployee);
71      RequestDispatcher dispatcher = request.getRequestDispatcher("employeeList.jsp");
72      dispatcher.forward(request, response);
73  }
74
75  private void showNewForm(HttpServletRequest request, HttpServletResponse response)
76      throws ServletException, IOException {
77      RequestDispatcher dispatcher = request.getRequestDispatcher("employeeForm.jsp");
78      dispatcher.forward(request, response);
79  }
80
81  private void showEditForm(HttpServletRequest request, HttpServletResponse response)
82      throws SQLException, ServletException, IOException {
83      int id = Integer.parseInt(request.getParameter("id"));
84      Employee existingEmployee = employeeDAO.selectEmployee(id);
85      RequestDispatcher dispatcher = request.getRequestDispatcher("employeeForm.jsp");
86      request.setAttribute("employee", existingEmployee);
87      dispatcher.forward(request, response);
88  }
```

```

90 private void insertEmployee(HttpServletRequest request, HttpServletResponse response)
91     throws SQLException, IOException {
92     String name = request.getParameter("name");
93     String email = request.getParameter("email");
94     String position = request.getParameter("position");
95     Employee newEmployee = new Employee(name, email, position);
96     employeeDAO.insertEmployee(newEmployee);
97     response.sendRedirect("list");
98 }
99
100 private void updateEmployee(HttpServletRequest request, HttpServletResponse response)
101     throws SQLException, IOException {
102     int id = Integer.parseInt(request.getParameter("id"));
103     String name = request.getParameter("name");
104     String email = request.getParameter("email");
105     String position = request.getParameter("position");
106
107     Employee employee = new Employee(id, name, email, position);
108     employeeDAO.updateEmployee(employee);
109     response.sendRedirect("list");
110 }
111
112 private void deleteEmployee(HttpServletRequest request, HttpServletResponse response)
113     throws SQLException, IOException {
114     int id = Integer.parseInt(request.getParameter("id"));
115     employeeDAO.deleteEmployee(id);
116     response.sendRedirect("list");
117 }
118 }
119 }

```

## Step 4 - Create these files:



### 1. File web.xml

Java web applications use a deployment descriptor file to determine how URLs map to servlets, which URLs require authentication, and other information. This file is named web.xml, and resides in the app's WAR under the WEB-INF/ directory. web.xml is part of the servlet standard for web applications.

### 2. File EmployeeForm.jsp (used for Add and Edit/Update process)

```

1  <% page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
3
4  <!DOCTYPE html>
5  <html>
6  <head>
7      <title>Employee Management Application</title>
8      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
9          integrity="sha384-ggOyR0iXCbMQV3Eipema34ND+dH/1fQ784/16cY/1JTQUCohcWz7x9Jv0Rxt2M2w1F" crossorigin="anonymous">
10  </head>
11  <body>
12      <header>
13          <nav class="navbar navbar-expand-md navbar-dark" style="background-color: tomato">
14              <div>
15                  <a href="#" class="navbar-brand"> Employee Management App </a>
16              </div>
17              <ul class="navbar-nav">
18                  <li><a href="<%=request.getContextPath()%>/list" class="nav-link">Employees</a></li>
19              </ul>
20          </nav>
21      </header>
22      <br>
23      <div class="container col-md-5">
24          <div class="card">
25              <div class="card-body">
26                  <c:if test="${employee != null}">
27                      <form action="update" method="post">
28                          </c:if>
29                      <c:if test="${employee == null}">

```

```

31 <form action="insert" method="post">
32 </c:if>
33
34 <h2>
35 <c:if test="{employee != null}">
36 Edit Employee
37 </c:if>
38 <c:if test="{employee == null}">
39 Add New Employee
40 </c:if>
41 </h2>
42
43 <c:if test="{employee != null}">
44 <input type="hidden" name="id" value="{c:out value='{employee.id}' />" />
45 </c:if>
46
47 <fieldset class="form-group">
48 <label>Employee Name</label> <input type="text" value="{c:out value='{employee.name}' />"
49 class="form-control" name="name" required="required">
50 </fieldset>
51
52 <fieldset class="form-group">
53 <label>Employee Email</label> <input type="text" value="{c:out value='{employee.email}' />"
54 class="form-control" name="email">
55 </fieldset>
56
57 <fieldset class="form-group">
58 <label>Employee Position</label>
59 <input type="text" value="{c:out value='{employee.position}' />" class="form-control" readonly >
60 <input list="positionList" id="position" class="form-control" name="position" >
61 <datalist id="positionList">
62 <option value="Manager">
63 <option value="Head of Dept">
64 <option value="Supervisor">
65 <option value="Director">
66 </datalist>
67 </fieldset>
68
69 <button type="submit" class="btn btn-success">Save</button>
70 </form>
71 </div>
72 </div>
73 </body>
74 </html>

```

### 3. File EmployeeList.jsp (used for displaying all employee records)

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
3
4 <!DOCTYPE html>
5 <html>
6
7 <head>
8 <title>Employee Management Application</title>
9 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
10 integrity="sha384-ggOyR0iXCbMQV3Xipma34MD+dR/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
11 </head>
12
13 <body>
14
15 <header>
16 <nav class="navbar navbar-expand-md navbar-dark" style="background-color: tomato">
17 <div>
18 <a href="" class="navbar-brand"> Employee Management App </a>
19 </div>
20 <ul class="navbar-nav">
21 <li><a href="{%=request.getContextPath()%}/list" class="nav-link">Employees</a></li>
22 </ul>
23 </nav>
24 </header>
25 <br>
26
27

```

```

27 <div class="row">
28 <!-- <div class="alert alert-success" ngIf='message'>{{message}}</div> -->
29
30 <div class="container">
31 <h3 class="text-center">List of Employees</h3>
32 <hr>
33 <div class="container text-left">
34 <a href="{%=request.getContextPath()%}/new" class="btn btn-success">Add New Employee</a>
35 </div>
36 <br>
37 <table class="table table-bordered">
38 <thead>
39 <tr>
40 <th>ID</th>
41 <th>Name</th>
42 <th>Email</th>
43 <th>Position</th>
44 <th>Actions</th>
45 </tr>
46 </thead>
47
48 <tbody>
49 <!-- for (Todo todo: todos) { -->
50 <c:forEach var="employee" items="${listEmployee}">
51 <tr>
52 <td>
53 <c:out value="${employee.id}" />
54 </td>
55 <td>
56 <c:out value="${employee.name}" />
57 </td>
58 <td>
59 <c:out value="${employee.email}" />
60 </td>
61 <td>
62 <c:out value="${employee.position}" />
63 </td>
64 <td><a href="edit?id=<c:out value='${employee.id}' />">Edit</a> &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
65 <a href="delete?id=<c:out value='${employee.id}' />">Delete</a></td>
66 </tr>
67 </c:forEach>
68 </tbody>
69 </table>
70 </div>
71 </div>
72 </body>
73 </html>
74

```

#### 4. File error.jsp

```

1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" isErrorPage="true" %>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
3 <html>
4 <head>
5 <title>Error page</title>
6 </head>
7 <body>
8 <center>
9 <h1>Error</h1>
10 <h2><%=exception.getMessage() %><br/> </h2>
11 </center>
12 </body>
13 </html>

```

#### 5. File index.jsp

Contents of the Index.jsp:

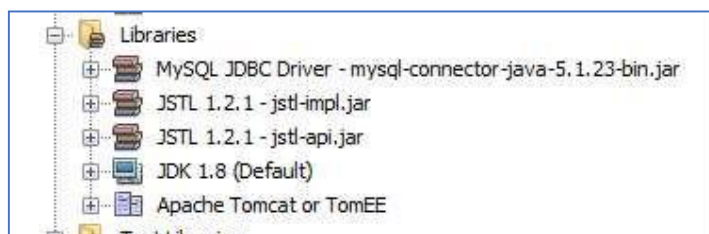


```

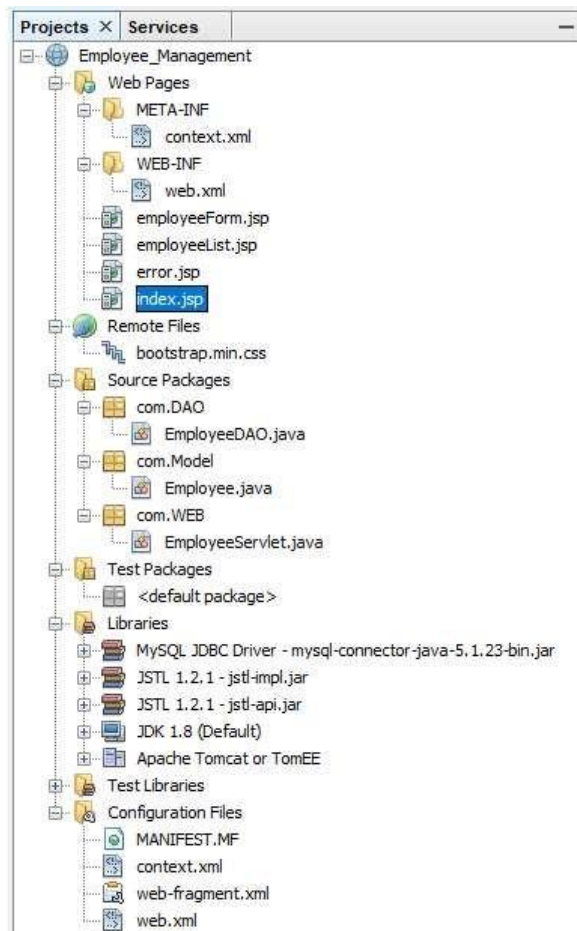
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <title>User Management Application</title>
6      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
7          integrity="sha384-ggOyR0iXCbMQV3Iipm34Iipm34Iipm34Iipm34Iipm34Iipm34Iipm34Iipm34Iipm34Iipm34"
8          crossorigin="anonymous">
9  </head>
10 <body>
11     <h1>Application MVC system for Employee Management</h1><br>
12
13     <ul>
14         <li> <a href="http://localhost:8080/Employee Management/list"> All Employee List </a></li>
15         <li> <a href="http://localhost:8080/Employee Management/new"> Add a New Employee </a></li>
16         <li> <a href="http://localhost:8080/Employee Management/list"> Edit Employee </a></li>
17     </ul>
18
19 </body>
20 </html>
21

```

**Step 5** - Add these libraries:



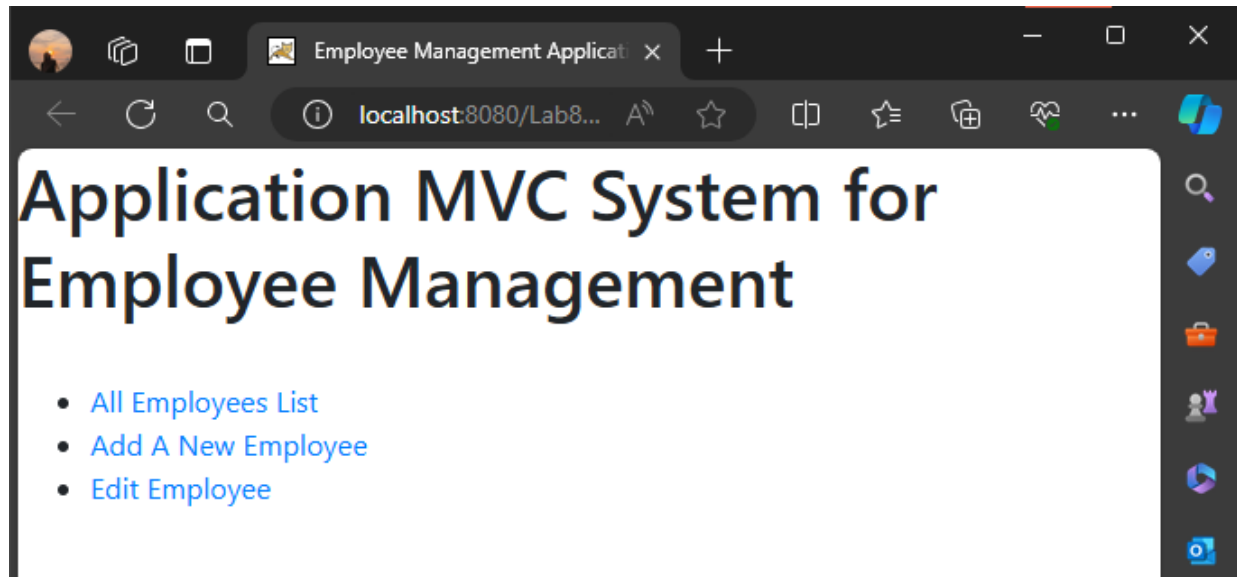
**Finally, the Project schema should be like this:**



## **Step 6** - Running the program and try CRUD process:

1. Run index.jsp page.
2. Click List All User button to show all records.
3. Click Add User button to create new record.
4. Click hyperlink Update to do edit/update an existing record.
5. Click hyperlink Delete to do delete an existing record.

## The Output :



A screenshot of the "Add New Employee" form within the "Employee Management App". The form is displayed in a web browser window with the URL `localhost:8080/Lab8...`. The form has a white background and is titled "Add New Employee". It contains three input fields: "Name" with the value "Arifah Hanafiah", "E-Mail" with the value "S66428@ocean.umt.edu.my", and "Position" with a dropdown menu showing "Manager". A green "Save" button is located at the bottom of the form.

Employee Management App

Employee

### Add New Employee

Name

Arifah Hanafiah

E-Mail

S66428@ocean.umt.edu.my

Position

Manager

Save

## Exercise

Using this database schema, please create MVC Application [CRUD] for Car Shop, using JSP, Servlet, and MySQL.

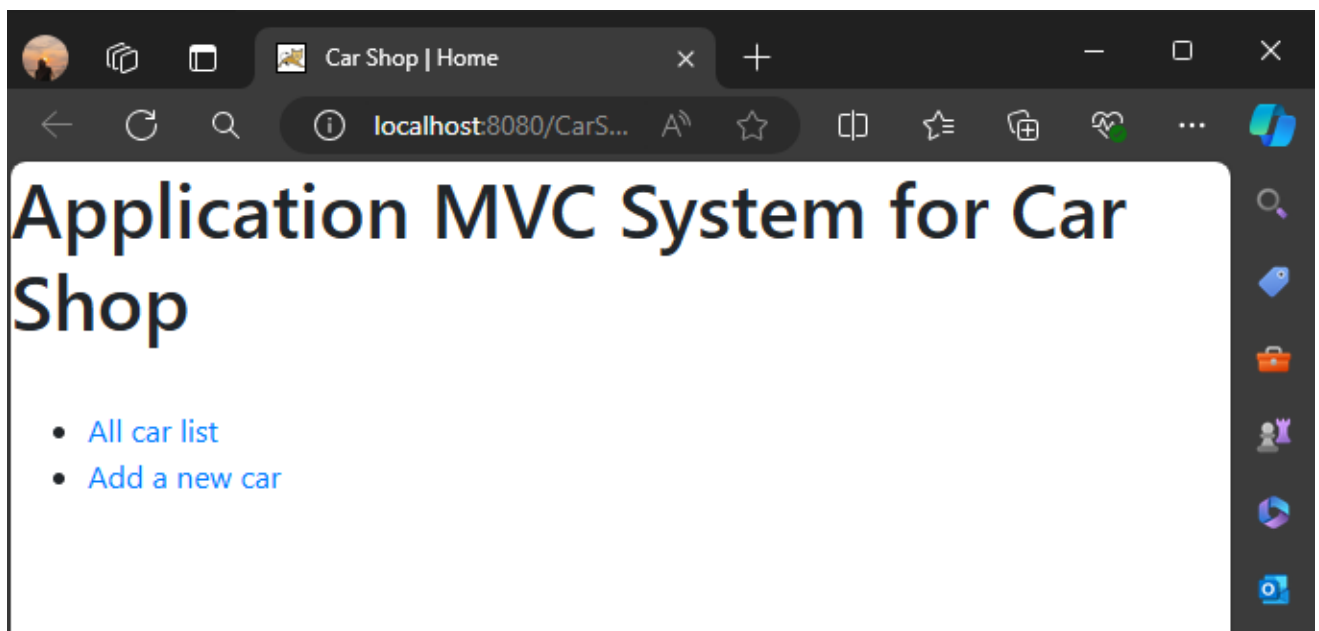
```
CREATE DATABASE if not EXISTS carshop;
USE carshop;

CREATE TABLE if not EXISTS CarPricelist(
    Car_id INT NOT NULL AUTO_INCREMENT,
    Brand VARCHAR(15),
    Model VARCHAR(30),
    Cylinder INT,
    Price DOUBLE,
    PRIMARY KEY (Car_id)
);
```

The application should be able to handle these activities:

1. View all data
2. Add new data
3. Edit/Update current data
4. Delete the specific data

## The Output :





Car Shop | Form

localhost:8080/CarS...

Car Shop

## Add New Car

Brand

Mercedes Benz

Model

CLA300

Cylinder

3

Price

450000.00

Clear

Add

Car Shop | List

Employee Manager

localhost8080/CarS...

Car Shop

Search

Home

Shopping Cart

Profile

Logout

Admin

Settings

Help

Feedback

Privacy Policy

Terms of Service

FAQ

Contact Us

Partners

Press

Investors

Jobs

Press Kit

Legal

Disclaimer

Refund Policy

Shipping Policy

Return Policy

Warranty

FAQ

Contact Us

Partners

Press

Investors

Jobs

Press Kit

Legal

Disclaimer

Refund Policy

Shipping Policy

Return Policy

Warranty

# List of Cars

Add new car

ID	Brand	Model	Cylinder	Price	Action
----	-------	-------	----------	-------	--------