CRUD-Based Employee Management System Using Java Servlet and JDBC

Abstract

This document outlines the development of a robust web-based Employee Management Application.
The system harnesses Java Servlets for request handling, JSP for view rendering, and JDBC for connecting to a MySQL database.
Adopting the Model-View-Controller design, the application cleanly decouples business logic, presentation, and data layers.
Core CRUD operations enable administrators to add, view, update, and delete employee records efficiently.
This project serves as a practical example of constructing maintainable, scalable enterprise applications using core Java technologies.

System Architecture: Model-View-Controller (MVC)

Model

- Encapsulates the core data structures and business rules.
    - Employee.java defines a POJO with properties (id, name, position, salary) and corresponding getters/setters.
    - EmployeeData.java implements a DAO using JDBC to perform parameterized CRUD queries, managing connections and handling SQL exceptions.
- Acts as the single source of truth for the application's data layer, decoupling database logic from presentation code.

View

- Consists of JSP pages that render HTML dynamically based on model data.
    - emplist.jsp displays all employee records in a Bootstrap-styled table, iterating with JSTL <c:forEach>.
    - empform.jsp provides an add/edit form, using <c:if> to toggle between insert and update modes.

- Ensures a clear separation of UI concerns by minimizing Java code in JSPs and leveraging JSTL for conditional rendering and value output.

Controller

- Centralized request handler implemented in EmployeeServlet.
    - Mapped at @WebServlet("/"), it intercepts all CRUD-related paths (/new, /insert, /edit, /update, /delete) and dispatches actions.
    - Uses doGet and doPost to parse parameters, invoke DAO methods, set request attributes, and forward to the appropriate JSP.
- Encapsulates navigation logic and orchestrates interactions between the model and the view, following a command-style switch on request.getServletPath().

1. Database Setup (MySQL)

1.1.Database Creation

```
CREATE DATABASE employee_mgmt;
USE employee_mgmt;
```

1.2.Creating the employees Table

```
CREATE TABLE employees (
    id         INT AUTO_INCREMENT PRIMARY KEY,
    name       VARCHAR(100) NOT NULL,
    position   VARCHAR(100) NOT NULL,
    department VARCHAR(100) NOT NULL,
    salary     DECIMAL(10,2) NOT NULL
);
```

1.3.Inserting Sample Records

```
INSERT INTO employees (name, position, department, salary) VALUES
('Amit Sharma',    'Software Engineer', 'IT',       55000.00);
```

2. Source Code

2.1 Employee.java

```
package com.example.employeemanage.model;

public class Employee {
    private int id;
    private String name;
```

```
    private String position;
    private String department;
    private double salary;

    public Employee() {}

    public Employee(String name, String position, String department, double salary) {
        this.name       = name;
        this.position   = position;
        this.department = department;
        this.salary     = salary;
    }

    public Employee(int id, String name, String position, String department, double salary) {
        this.id         = id;
        this.name       = name;
        this.position   = position;
        this.department = department;
        this.salary     = salary;
    }

    // Getters and setters
    public int getId() { return id; }
    public String getName() { return name; }
    public String getPosition() { return position; }
    public String getDepartment() { return department; }
    public double getSalary() { return salary; }

    public void setId(int id) { this.id = id; }
    public void setName(String name) { this.name = name; }
    public void setPosition(String position) { this.position = position; }
    public void setDepartment(String department) { this.department = department; }
    public void setSalary(double salary) { this.salary = salary; }
}
```

2.2 EmployeeData.java (DAO)

```
  package com.example.employeemanage.dao;

import com.example.employeemanage.model.Employee;
import java.sql.*;
```

```
import java.util.ArrayList;
import java.util.List;

public class EmployeeData {
   private String jdbcURL      = "jdbc:mysql://localhost:3306/employee_mgmt?useSSL=false";
   private String jdbcUser     = "root";      // update as needed
   private String jdbcPassword = "password";

   private static final String INSERT_SQL =
    "INSERT INTO employees (name, position, department, salary) VALUES (?, ?, ?, ?);";
   private static final String SELECT_BY_ID =
    "SELECT * FROM employees WHERE id = ?;";
   private static final String SELECT_ALL =
    "SELECT * FROM employees;";
   private static final String DELETE_SQL =
    "DELETE FROM employees WHERE id = ?;";
   private static final String UPDATE_SQL =
    "UPDATE employees SET name = ?, position = ?, department = ?, salary = ? WHERE id = ?;";

   protected Connection getConnection() throws SQLException {
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
      } catch (ClassNotFoundException ex) {
         ex.printStackTrace();
      }
      return DriverManager.getConnection(jdbcURL, jdbcUser, jdbcPassword);
   }

   public void insertEmployee(Employee emp) throws SQLException {
      try (Connection conn = getConnection();
           PreparedStatement ps = conn.prepareStatement(INSERT_SQL)) {
         ps.setString(1, emp.getName());
         ps.setString(2, emp.getPosition());
         ps.setString(3, emp.getDepartment());
         ps.setDouble(4, emp.getSalary());
         ps.executeUpdate();
      }
   }

   public Employee selectEmployee(int id) {
      Employee emp = null;
```

```
    try (Connection conn = getConnection();
         PreparedStatement ps = conn.prepareStatement(SELECT_BY_ID)) {
        ps.setInt(1, id);
        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            emp = new Employee(
              id,
              rs.getString("name"),
              rs.getString("position"),
              rs.getString("department"),
              rs.getDouble("salary")
            );
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    return emp;
}

public List<Employee> selectAllEmployees() {
    List<Employee> list = new ArrayList<>();
    try (Connection conn = getConnection();
         PreparedStatement ps = conn.prepareStatement(SELECT_ALL);
         ResultSet rs = ps.executeQuery()) {
        while (rs.next()) {
            list.add(new Employee(
              rs.getInt("id"),
              rs.getString("name"),
              rs.getString("position"),
              rs.getString("department"),
              rs.getDouble("salary")
            ));
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    }
    return list;
}

public boolean updateEmployee(Employee emp) throws SQLException {
    try (Connection conn = getConnection();
```

```
            PreparedStatement ps = conn.prepareStatement(UPDATE_SQL)) {
            ps.setString(1, emp.getName());
            ps.setString(2, emp.getPosition());
            ps.setString(3, emp.getDepartment());
            ps.setDouble(4, emp.getSalary());
            ps.setInt(5, emp.getId());
            return ps.executeUpdate() > 0;
        }
    }

    public boolean deleteEmployee(int id) throws SQLException {
        try (Connection conn = getConnection();
            PreparedStatement ps = conn.prepareStatement(DELETE_SQL)) {
            ps.setInt(1, id);
            return ps.executeUpdate() > 0;
        }
    }
}
```

2.3 EmployeeServlet.java

```
package com.example.employeemanage.web;


import com.example.employeemanage.dao.EmployeeData;
import com.example.employeemanage.model.Employee;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;


@WebServlet("/")
public class EmployeeServlet extends HttpServlet {
    private EmployeeData dao;
```

```
public void init() {
    dao = new EmployeeData();
}

protected void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    doGet(req, res);
}

protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    String action = req.getServletPath();
    try {
        switch (action) {
            case "/new":    showForm(req, res);     break;
            case "/insert": insert(req, res);       break;
            case "/delete": delete(req, res);       break;
            case "/edit":   edit(req, res);         break;
            case "/update": update(req, res);       break;
            default:        list(req, res);         break;
        }
    } catch (SQLException ex) {
        throw new ServletException(ex);
    }
}
```

```
private void list(HttpServletRequest req, HttpServletResponse res)
    throws SQLException, ServletException, IOException {
    List<Employee> list = dao.selectAllEmployees();
    req.setAttribute("listEmployee", list);
    RequestDispatcher dispatcher = req.getRequestDispatcher("emplist.jsp");
    dispatcher.forward(req, res);
}

private void showForm(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    RequestDispatcher dispatcher = req.getRequestDispatcher("empform.jsp");
    dispatcher.forward(req, res);
}

private void insert(HttpServletRequest req, HttpServletResponse res)
    throws SQLException, IOException {
    String name       = req.getParameter("name");
    String position   = req.getParameter("position");
    String department = req.getParameter("department");
    double salary     = Double.parseDouble(req.getParameter("salary"));
    Employee emp      = new Employee(name, position, department, salary);
    dao.insertEmployee(emp);
    res.sendRedirect("list");
}

private void edit(HttpServletRequest req, HttpServletResponse res)
    throws SQLException, ServletException, IOException {
```

```
        int id          = Integer.parseInt(req.getParameter("id"));
        Employee emp    = dao.selectEmployee(id);
        req.setAttribute("employee", emp);
        RequestDispatcher dispatcher = req.getRequestDispatcher("empform.jsp");
        dispatcher.forward(req, res);
    }

    private void update(HttpServletRequest req, HttpServletResponse res)
        throws SQLException, IOException {
        int id          = Integer.parseInt(req.getParameter("id"));
        String name     = req.getParameter("name");
        String position = req.getParameter("position");
        String department = req.getParameter("department");
        double salary   = Double.parseDouble(req.getParameter("salary"));
        Employee emp    = new Employee(id, name, position, department, salary);
        dao.updateEmployee(emp);
        res.sendRedirect("list");
    }

    private void delete(HttpServletRequest req, HttpServletResponse res)
        throws SQLException, IOException {
        int id = Integer.parseInt(req.getParameter("id"));
        dao.deleteEmployee(id);
        res.sendRedirect("list");
    }
}
```

3. JSP Pages

3.1 emplist.jsp

```
    <%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
 <title>Employee Management</title>
 <link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
 <div class="container mt-4">
  <h2 class="text-center">List of Employees</h2>
  <a href="new" class="btn btn-success mb-2">Add New Employee</a>
  <table class="table table-bordered">
   <thead>
    <tr>
     <th>ID</th><th>Name</th>
     <th>Position</th><th>Department</th>
     <th>Salary</th><th>Actions</th>
    </tr>
   </thead>
   <tbody>
    <c:forEach var="emp" items="${listEmployee}">
     <tr>
      <td>${emp.id}</td>
      <td>${emp.name}</td>
      <td>${emp.position}</td>
```

```
          <td>${emp.department}</td>
          <td>${String.format('%.2f', emp.salary)}</td>
          <td>
            <a href="edit?id=${emp.id}"
               class="btn btn-primary btn-sm">Edit</a>
            <a href="delete?id=${emp.id}"
               class="btn btn-danger btn-sm">Delete</a>
          </td>
        </tr>
      </c:forEach>
    </tbody>
  </table>
 </div>
</body>
</html>
```

3.2 empform.jsp

```
  <%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
 <title>Employee Form</title>
 <link rel="stylesheet"
     href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
 <div class="container mt-5 col-md-6">
```

```
<h3 class="text-center">
  <c:choose>
    <c:when test="${employee != null}">Edit Employee</c:when>
    <c:otherwise>Add Employee</c:otherwise>
  </c:choose>
</h3>
<form action="${employee != null ? 'update' : 'insert'}" method="post">
  <c:if test="${employee != null}">
    <input type="hidden" name="id" value="${employee.id}" />
  </c:if>
  <div class="form-group">
    <label>Name</label>
    <input type="text" name="name" value="${employee.name}"
          class="form-control" required/>
  </div>
  <div class="form-group">
    <label>Position</label>
    <input type="text" name="position" value="${employee.position}"
          class="form-control" required/>
  </div>
  <div class="form-group">
    <label>Department</label>
    <input type="text" name="department" value="${employee.department}"
          class="form-control" required/>
  </div>
  <div class="form-group">
    <label>Salary</label>
```

```
      <input type="number" step="0.01" name="salary"
          value="${employee.salary}" class="form-control" required/>
    </div>
    <button type="submit" class="btn btn-success">Save</button>
    <a href="list" class="btn btn-secondary">Cancel</a>
  </form>
 </div>
</body>
</html>
```

4. Deployment Descriptor (web.xml)

```
    <?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
       version="4.0">
 <display-name>EmployeeManagementSystem</display-name>
 <welcome-file-list>
  <welcome-file>list</welcome-file>
 </welcome-file-list>
</web-app>
```

Output and Functionality

Once deployed on a servlet container like Apache Tomcat, users can manage employee records through a web interface.

● View All Employees: Navigating to the application's root URL displays all employee entries in a paginated table.

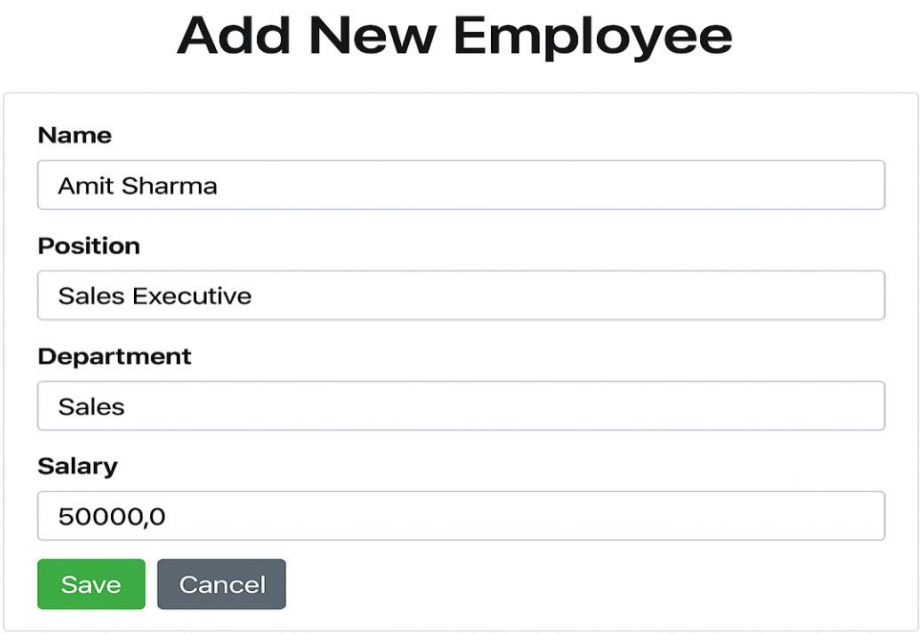
● Add New Employee: The "Add New Employee" button opens a form to input details, submitting a POST request to /insert and refreshing the list.

● Edit Employee: The "Edit" action loads existing employee data into the form, sending updates via a POST to /update and returning to the updated view.

● Delete Employee: The "Delete" action triggers a GET request to /delete, removes the record from the database, and updates the list accordingly.

Outputs:

Inserting record

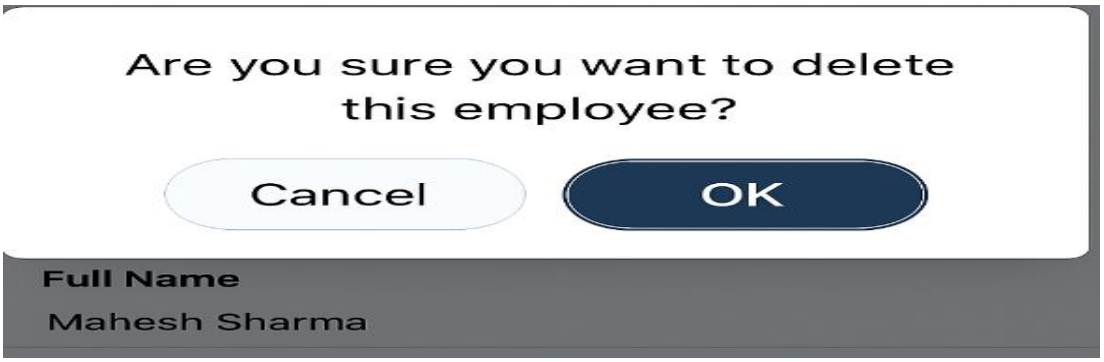


Updating record

3)Displaying Employees

ID	Name	Position	Department	Salary
3	Mahesh Sharma	Developer	IT	55000
4	Rohit Gupta	Developer	IT	57000
7	Vikram Patel	SysAdmin	IT	60000

4) Deleting record



Conclusion

The Employee Management System was successfully developed, meeting the project objectives with a cohesive web interface for full CRUD operations.

By leveraging Java Servlets, JSP, and JDBC within the MVC framework, the application achieves modularity, maintainability, and scalability.

The use of JDBC with prepared statements ensures secure and efficient database interactions, while JSP and JSTL deliver a clean and responsive user experience.

Overall, this project exemplifies a robust, enterprise-ready approach to Java web development, aligning with both academic goals and industry best practices.