



Web based Application Development (CSM3023)

Lab Report 8

An MVC Example with Servlets and JSP

Programme

Computer Science (Software Engineering)

Prepared by

NurSyaza Amira Binti Selamat

Matric No

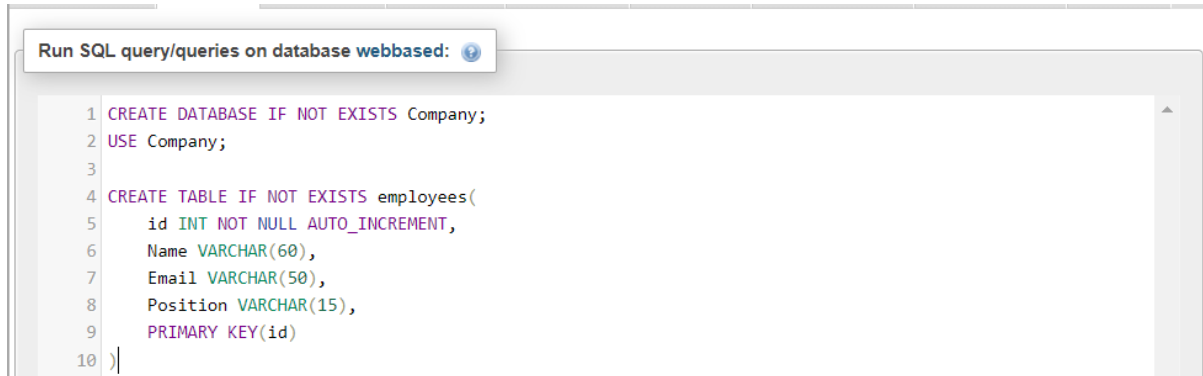
S62318

Prepared for

Sir Arizal

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

Step 1: Create table employees in COMPANY database schema

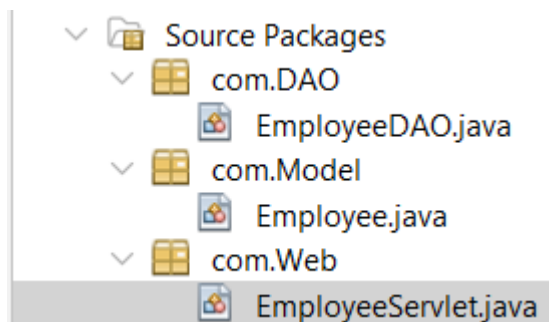


```
1 CREATE DATABASE IF NOT EXISTS Company;
2 USE Company;
3
4 CREATE TABLE IF NOT EXISTS employees(
5     id INT NOT NULL AUTO_INCREMENT,
6     Name VARCHAR(60),
7     Email VARCHAR(50),
8     Position VARCHAR(15),
9     PRIMARY KEY(id)
10 )
```

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

```
package com.DAO;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
```

```

import java.util.List;

import com.Model.Employee;
import java.sql.Statement;

public class EmployeeDAO {

    Connection connection = null;
    private String jdbcurl = "jdbc:mysql://localhost:3306/company";
    private String jdbcUsername = "root";
    private String jdbcPassword = "admin01";

    private static final String INSERT_EMPLOYEES_SQL = "INSERT INTO
employees (name, email, position) VALUES "
        + "(?,?,?);";
    private static final String SELECT_EMPLOYEE_BY_ID = "select id, name,
email, position from employees where id =?";
    private static final String SELECT_ALL_EMPLOYEES = "select * from
employees";
    private static final String DELETE_EMPLOYEES_SQL = "delete from
employees where id =?";
    private static final String UPDATE_EMPLOYEES_SQL = "update employees
set name =?, email =?, position =? where id =?";

    public EmployeeDAO() {
    }

    protected Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(jdbcurl, jdbcUsername,
jdbcPassword);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return connection;
    }
}

```

```

public void insertEmployee(Employee employee) throws SQLException {
    System.out.println(INSERT_EMPLOYEES_SQL);
    try (Connection connection = getConnection(); PreparedStatement
preparedStatement
        = connection.prepareStatement(INSERT_EMPLOYEES_SQL)) {
        preparedStatement.setString(1, employee.getName());
        preparedStatement.setString(2, employee.getEmail());
        preparedStatement.setString(3, employee.getPosition());
        System.out.println(preparedStatement);
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        printSQLException(e);
    }
}

```

```

public Employee selectEmployee(int id) {
    Employee employee = null;
    //step 1 : Establishing a connection
    try (Connection connection = getConnection(); //step 2: Create a statement
using connection object
        PreparedStatement preparedStatement =
connection.prepareStatement(SELECT_EMPLOYEE_BY_ID);) {
        preparedStatement.setInt(1, id);
        System.out.println(preparedStatement);
        //step 3: Execute the query or update query
        ResultSet rs = preparedStatement.executeQuery();

        //Step 4: Process the ResultSet object
        while (rs.next()) {
            String name = rs.getString("name");
            String email = rs.getString("email");
            String position = rs.getString("position");
            employee = new Employee(id, name, email, position);
        }
    } catch (SQLException e) {
        printSQLException(e);
    }
    return employee;
}

```

```

public List< Employee> selectAllEmployees() {

    //using try with response to avoid closing resources to avoid resources
    (boiler plate code)
    List< Employee> employees = new ArrayList<>();
    //Step 1: Establishing a connection
    try (Connection connection = getConnection(); //Step 2: Create a statement
using connection object
        PreparedStatement preparedStatement =
connection.prepareStatement(SELECT_ALL_EMPLOYEES);) {

        System.out.println(preparedStatement);
        //Step 3: Execute the query or update query
        ResultSet rs = preparedStatement.executeQuery();

        //Step 4: Process the ResultSet onject
        while (rs.next()) {
            int id = rs.getInt("id");
            String name = rs.getString("name");
            String email = rs.getString("email");
            String position = rs.getString("position");
            employees.add(new Employee(id, name, email, position));
        }
    } catch (SQLException e) {
        printSQLException(e);
    }
    return employees;
}

public boolean deleteEmployee(int id) throws SQLException {
    boolean rowDeleted;
    try (Connection connection = getConnection(); PreparedStatement
statement
        = connection.prepareStatement(DELETE_EMPLOYEES_SQL);) {
        statement.setInt(1, id);
        rowDeleted = statement.executeUpdate() > 0;
    }
    return rowDeleted;
}

```

```

public boolean updateEmployee(Employee employee) throws SQLException
{
    boolean rowUpdated;
    try (Connection connection = getConnection(); PreparedStatement
statement
        = connection.prepareStatement(UPDATE_EMPLOYEES_SQL);) {
        statement.setString(1, employee.getName());
        statement.setString(2, employee.getEmail());
        statement.setString(3, employee.getPosition());
        statement.setInt(4, employee.getId());

        rowUpdated = statement.executeUpdate() > 0;
    }
    return rowUpdated;
}

```

```

private void printSQLException(SQLException ex) {
    for (Throwable e : ex) {
        if (e instanceof SQLException) {
            e.printStackTrace(System.err);
            System.err.println("SQLState: " + ((SQLException) e).getSQLState());
            System.err.println("Error Code: " + ((SQLException)
e).getErrorCode());
            System.err.println("Message: " + e.getMessage());
            Throwable t = ex.getCause();
            while (t != null) {
                System.out.println("Cause: " + t);
                t = t.getCause();
            }
        }
    }
}

```

```

public static void main(String[] args) {
    EmployeeDAO dao = new EmployeeDAO();
    try (Connection connection = dao.getConnection()) {
        if (connection != null) {
            System.out.println("Connected to the database!");
            // Perform a simple query

```

```

try (Statement stmt = connection.createStatement()) {
    ResultSet rs = stmt.executeQuery("SELECT * FROM employees");
    while (rs.next()) {
        System.out.println("ID: " + rs.getInt("id"));
        System.out.println("Name: " + rs.getString("Name"));
        System.out.println("Email: " + rs.getString("Email"));
        System.out.println("Position: " + rs.getString("Position"));
    }
}
} else {
    System.out.println("Failed to make connection!");
}
} catch (SQLException e) {
    e.printStackTrace();
}
}
}

```

Employee.java

```
package com.Model;
```

```
public class Employee {
```

```

    private int id;
    private String name;
    private String email;
    private String position;

```

```

    public Employee(int id, String name, String email, String position) {
        super();
        this.id = id;
        this.name = name;
        this.email = email;
        this.position = position;
    }

```

```

    public Employee(String name, String email, String position) {
        super();
        this.name = name;
    }

```

```
        this.email = email;
        this.position = position;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }
}
```

EmployeeServlet.java

```
package com.Web;
```



```
import java.io.IOException;
import java.sql.SQLException;
import java.util.List;
```

```
import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
```

```
import com.DAO.EmployeeDAO;
import com.Model.Employee;
```

```
@WebServlet("/")
public class EmployeeServlet extends HttpServlet {
```

```
    private EmployeeDAO employeeDAO;
```

```
    public void init() {
        employeeDAO = new EmployeeDAO();
    }
```

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

```
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
        String action = request.getServletPath();
        try {
            switch (action) {
                case "/new":
                    showNewForm(request, response);
```

```

        break;
    case "/insert":
        insertEmployee(request, response);
        break;
    case "/delete":
        deleteEmployee(request, response);
        break;
    case "/edit":
        showEditForm(request, response);
        break;
    case "/update":
        updateEmployee(request, response);
        break;
    default:
        listEmployee(request, response);
        break;
    }
} catch (SQLException ex) {
    throw new ServletException(ex);
}
}

```

```

private void listEmployee(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException, ServletException {
    List<Employee> listEmployee = employeeDAO.selectAllEmployees();
    request.setAttribute("listEmployee", listEmployee);
    RequestDispatcher dispatcher =
request.getRequestDispatcher("EmployeeList.jsp");
    dispatcher.forward(request, response);
}

```

```

private void showNewForm(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException, ServletException {
    RequestDispatcher dispatcher =
request.getRequestDispatcher("EmployeeForm.jsp");
    dispatcher.forward(request, response);
}

```

```

private void showEditForm(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException, ServletException {
    int id = Integer.parseInt(request.getParameter("id"));
    Employee existingEmployee = employeeDAO.selectEmployee(id);
    RequestDispatcher dispatcher =
request.getRequestDispatcher("EmployeeForm.jsp");
    request.setAttribute("employee", existingEmployee);
    dispatcher.forward(request, response);
}

```

```

private void insertEmployee(HttpServletRequest request,
HttpServletResponse response)
    throws SQLException, IOException, ServletException {
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String position = request.getParameter("position");
    Employee newEmployee = new Employee(name, email, position);
    employeeDAO.insertEmployee(newEmployee);
    response.sendRedirect("list");
}

```

```

private void updateEmployee(HttpServletRequest request,
HttpServletResponse response)
    throws SQLException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String position = request.getParameter("position");
    Employee employee = new Employee(id, name, email, position);
    employeeDAO.updateEmployee(employee);
    response.sendRedirect("list");
}

```

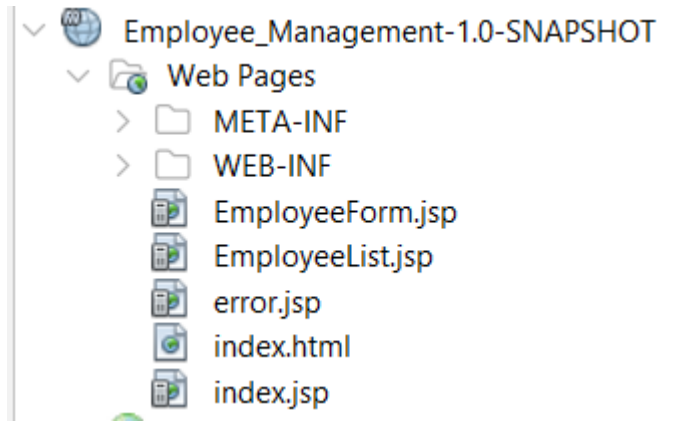
```

private void deleteEmployee(HttpServletRequest request,
HttpServletResponse response)
    throws SQLException, IOException, ServletException {
    int id = Integer.parseInt(request.getParameter("id"));
    employeeDAO.deleteEmployee(id);
    response.sendRedirect("list");
}

```

```
}  
}
```

Step 4: Create EmployeeForm.jsp, EmployeeList.jsp, error.jsp, and index.jsp



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

```
<% @page contentType="text/html" pageEncoding="UTF-8"%>  
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  
<!DOCTYPE html>  
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <title>Employee Management Application</title>  
    <link rel="stylesheet"  
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"  
    integrity="sha384-  
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2  
MZw1T"  
    crossorigin="anonymous">  
  </head>  
  <body>  
    <header>  
      <nav class="navbar navbar-expand-md navbar-dark" style="background-  
color:  
        tomato">  
        <div>  
          <a href="" class="navbar-brand">Employee Management App</a>  
        </div>
```



```

</fieldset>

<fieldset class="form-group">
  <label>Employee Email</label>
  <input type="text" value="<c:out

      value="\${employee.email}"/>"
      class="form-control" name="email">
</fieldset>

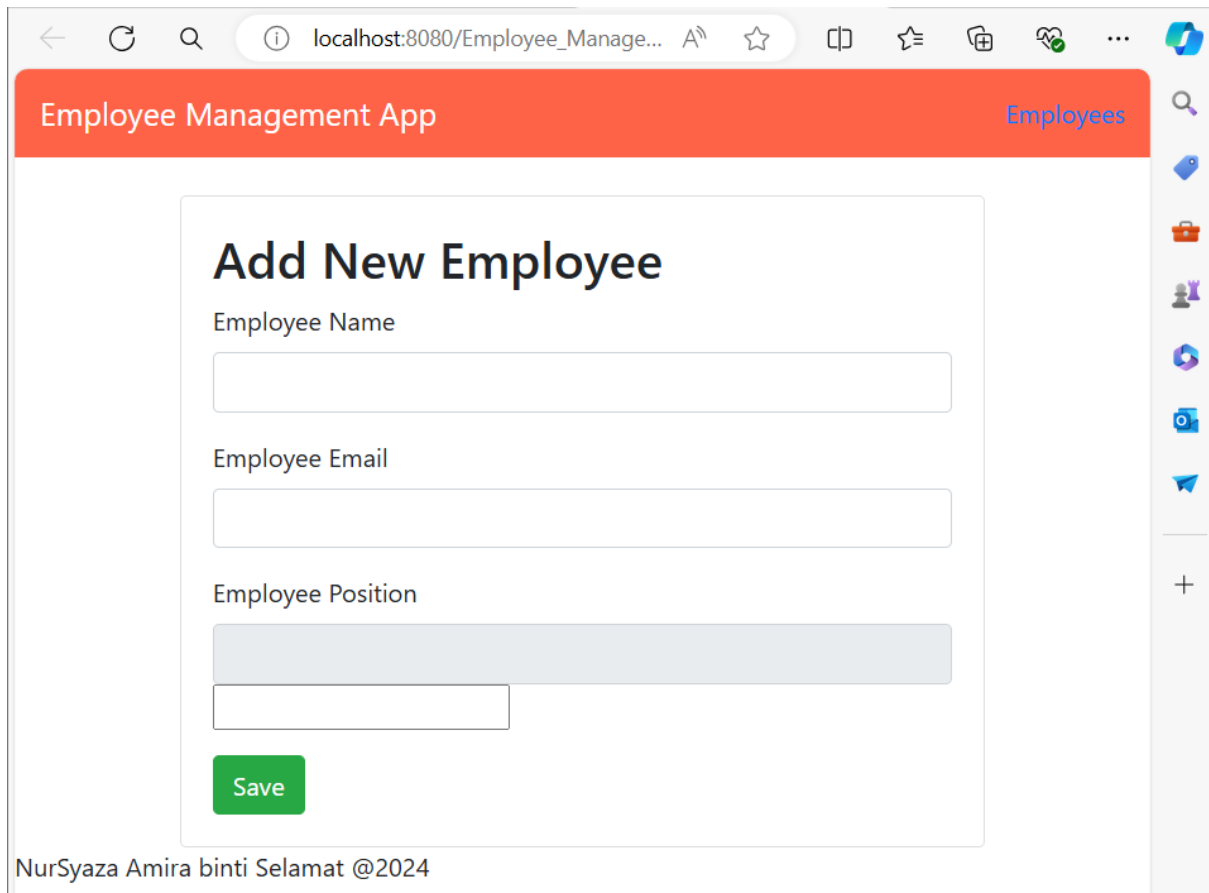
<fieldset class="form-group">
  <label>Employee Position</label>
  <input type="text" value="<c:out
      value="\${employee.position}"/>"      class="form-
control" readonly>

  <input list="positionList" id="position" class="form-
control" name="position">

  <datalist id="positionList">
    <option value="Manager"></option>
    <option value="Head of Dept"></option>
    <option value="Supervisor"></option>
    <option value="Director"></option>

  </datalist>
</fieldset>
<button type="submit" class="btn btn-success">Save</button>
</form>
</div>
</div>
</div>
</body>
<footer>NurSyaza Amira binti Selamat @2024</footer>
</html>

```



EmployeeList.jsp (used for displaying all employee records)

```
<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Employee Management Application</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
      integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2
MZw1T"
      crossorigin="anonymous">
  </head>
  <body>
    <header>
```

```

    <nav class="navbar navbar-expand-md navbar-dark" style="background-
color:
    tomato">
    <div>
        <a href="" class="navbar-brand">Employee Management App</a>
    </div>
    <ul class="navbar-nav">
        <li><a
href="<%=request.getContextPath()%>/list">Employees</a></li>
    </ul>
    </nav>
</header>
<br>
<div class="row">
    <div class="container">
        <h3 class="text-center">List of Employees</h3>
        <hr>
        <div class="container text-left">

            <a href="<%=request.getContextPath()%>/new" class="btn btn-
            success">Add New Employee</a>

        </div>
        <br>
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Email</th>
                    <th>Position</th>
                    <th>Actions</th>

                </tr>
            </thead>
            <tbody>
                <c:forEach var="employee" items="${listEmployee}">
                    <tr>
                        <td>
                            <c:out value="${employee.id}" />

```


[illegible]

List of Employees

Add New Employee

ID	Name	Email	Position	Actions
2	NurSyaza Amira	nursyazaamira77@gmail.com	Manager	Edit Delete
3	Ahmad Ali	ahmadali@gmail.com	Head of Dept	Edit Delete
4	Farrah Waheeda	frhwhaeda@gmail.com	Supervisor	Edit Delete
5	Nur Qaseh	nrqsseh@gmail.com	Supervisor	Edit Delete
6	Anis Amirah Zawawi	a.amirazawawi@yahoo.com	Head of Dept	Edit Delete

NurSyaza Amira binti Selamat @2024

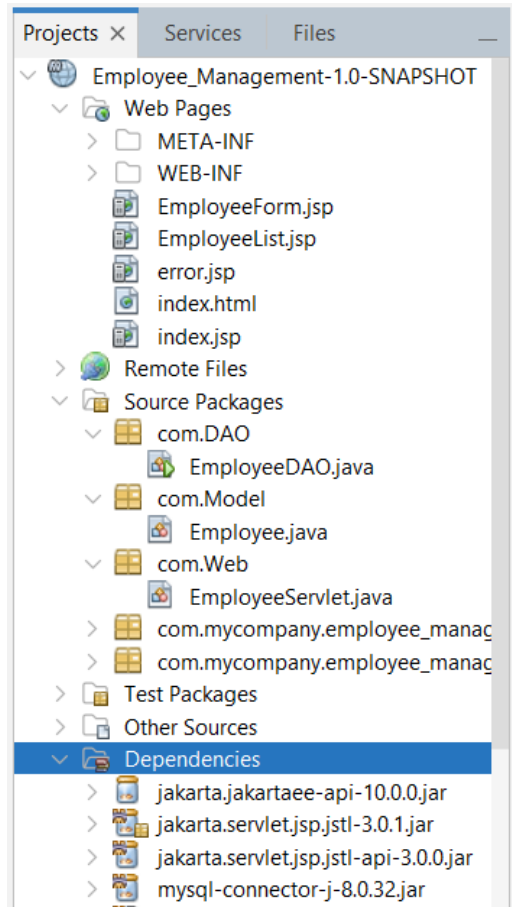
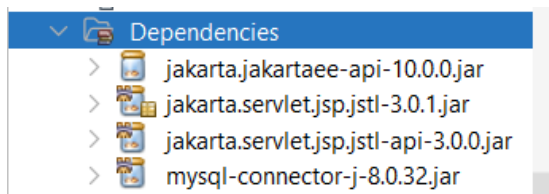
File error.jsp

```
Start Page x EmployeeDAO.java x Employee.java x EmployeeServlet.java x EmployeeForm.jsp x EmployeeList.jsp x error.jsp x
4 <html>
5 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"
6     isErrorPage="true"%>
7 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
8     "http://www.w3.org/TR/html4/loose.dtd">
9 <html>
10 <head>
11 <title>Error Page</title>
12 </head>
13 <body>
14 <h1>Error</h1>
15 <h2><%=exception.getMessage() %><br>Exception</h2>
16 </body>
17 <footer>NurSyaza Amira binti Selamat @2024</footer>
18 </html>
```

Index.jsp

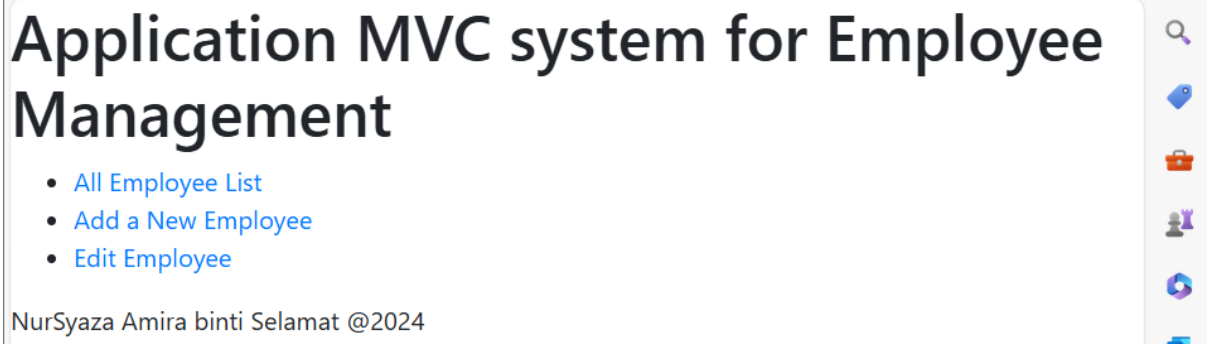
```
Start Page x EmployeeDAO.java x Employee.java x EmployeeServlet.java x EmployeeForm.jsp x EmployeeList.jsp x error.jsp x index.jsp x
4 <html>
5 <!--page contentType="text/html" pageEncoding="UTF-8"-->
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <title>User Management Application</title>
10 <link rel="stylesheet"
11 href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
12 integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
13 crossorigin="anonymous">
14 </head>
15 <body>
16 <h1>Application MVC system for Employee Management</h1>
17 <ul>
18 <li><a href="http://localhost:8080/Employee Management/list">All Employee
19 List</a></li>
20 <li><a href="http://localhost:8080/Employee Management/new">Add a New
21 Employee </a></li>
22 <li><a href="http://localhost:8080/Employee Management/list">Edit Employee
23 </a></li>
24 </ul>
25 </body>
26 <footer>NurSyaza Amira binti Selamat 82024</footer>
27 </html>
```

Step 5: Add these libraries

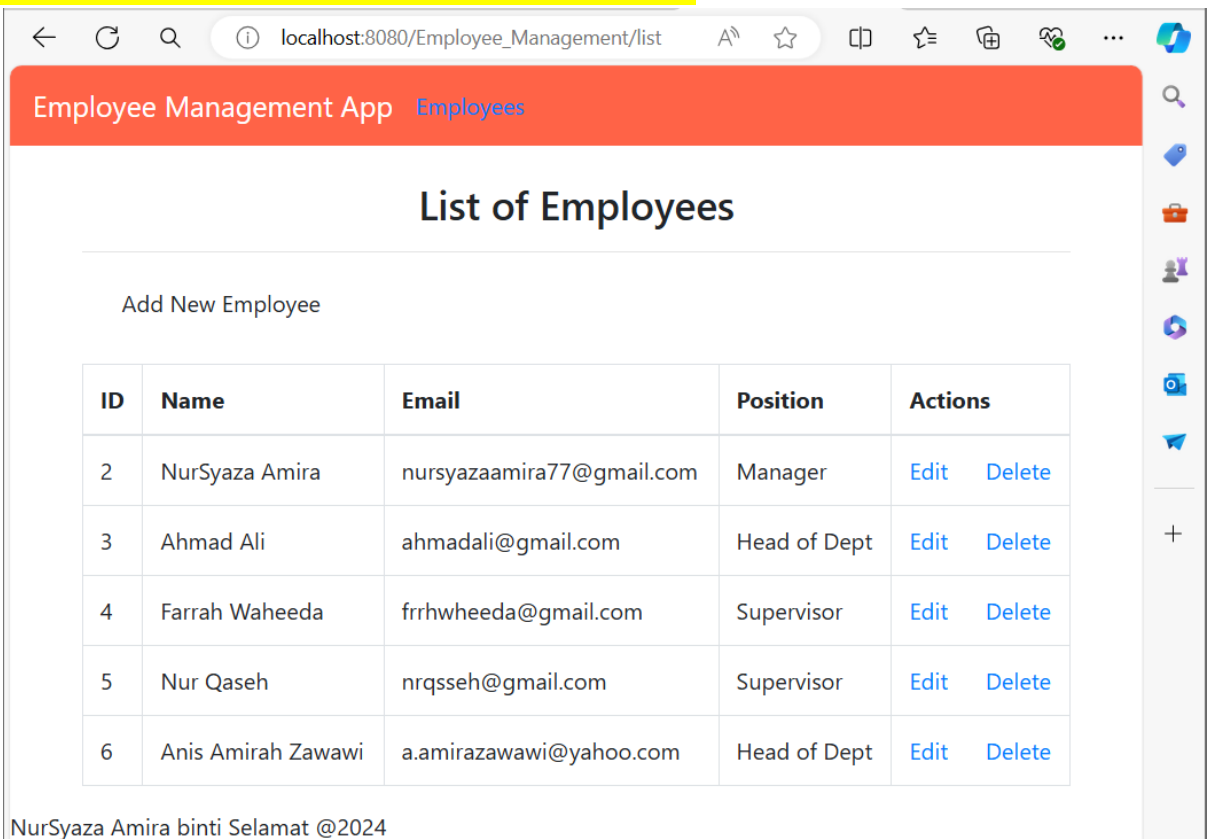


Step 6: Running the program and try CRUD process.

1. Run index.jsp page.



2. Click List All User button to show all records.



3. Click Add User button to create new record.

Employee Management App

Employees

Edit Employee

Employee Name

Fakrul Fahmi

Employee Email

f.fahmi77@gmail.com

Employee Position

Director

Save

NurSyaza Amira binti Selamat @2024

4. Click hyperlink Update to do edit/update an existing record.

Employee Management App

Employees

Edit Employee

Employee Name

Fakrul Fahmi

Employee Email

f.fahmi70@gmail.com

Employee Position

Director

Manager

Save

NurSyaza Amira binti Selamat @2024

7	Fakrul Fahmi	f.fahmi70@gmail.com	Manager	Edit	Delete
---	--------------	---------------------	---------	----------------------	------------------------

NurSyaza Amira binti Selamat @2024

5. Click hyperlink Delete to do delete an existing record.

Employee Management App Employees

List of Employees

Add New Employee

ID	Name	Email	Position	Actions
2	NurSyaza Amira	nursyazaamira77@gmail.com	Manager	Edit Delete
3	Ahmad Ali	ahmadali@gmail.com	Head of Dept	Edit Delete
4	Farrah Waheeda	frrhwheeda@gmail.com	Supervisor	Edit Delete
5	Nur Qaseh	nrqsseh@gmail.com	Supervisor	Edit Delete
6	Anis Amirah Zawawi	a.amirazawawi@yahoo.com	Head of Dept	Edit Delete

NurSyaza Amira binti Selamat @2024









		id	Name	Email	Position
<input type="checkbox"/>	Edit	Copy	Delete	2 NurSyaza Amira	nursyazaamira77@gmail.com Manager
<input type="checkbox"/>	Edit	Copy	Delete	3 Ahmad Ali	ahmadali@gmail.com Head of Dept
<input type="checkbox"/>	Edit	Copy	Delete	4 Farrah Waheeda	frrhwheeda@gmail.com Supervisor
<input type="checkbox"/>	Edit	Copy	Delete	5 Nur Qaseh	nrqsseh@gmail.com Supervisor
<input type="checkbox"/>	Edit	Copy	Delete	6 Anis Amirah Zawawi	a.amirazawawi@yahoo.com Head of Dept

☐ Check all With selected: Edit Copy Delete Export

☐ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Exercie

```
CREATE TABLE cars (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    brand VARCHAR(255) NOT NULL,  
    model VARCHAR(255) NOT NULL,  
    cylinder INT NOT NULL,  
    price DOUBLE NOT NULL  
);
```

	id	brand	model	cylinder	price
<input type="checkbox"/>  Edit  Copy  Delete	1	Produa	axia	1	60000
 <input type="checkbox"/> Check all <i>With selected:</i>  Edit  Copy  Delete  Export					

CarDAO.java

```
package com.DAO;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.List;  
  
import com.Model.Car;  
import java.sql.Statement;  
  
public class CarDAO {  
    private String jdbcurl = "jdbc:mysql://localhost:3306/carshop";  
    private String jdbcUsername = "root";  
    private String jdbcPassword = "admin01";  
  
    private static final String INSERT_CARS_SQL = "INSERT INTO cars (brand,  
model, cylinder, price) VALUES (?, ?, ?, ?)";  
    private static final String SELECT_CAR_BY_ID = "SELECT id, brand,  
model, cylinder, price FROM cars WHERE id = ?";  
    private static final String SELECT_ALL_CARS = "SELECT * FROM cars";
```

```
private static final String DELETE_CARS_SQL = "DELETE FROM cars  
WHERE id = ?";
```

```
private static final String UPDATE_CARS_SQL = "UPDATE cars SET brand  
= ?, model = ?, cylinder = ?, price = ? WHERE id = ?";
```

```
public CarDAO() {}
```

```
protected Connection getConnection() {  
    Connection connection = null;  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
        connection = DriverManager.getConnection(jdbcurl, jdbcUsername,  
jdbcPassword);  
    } catch (SQLException | ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
    return connection;  
}
```

```
public void insertCar(Car car) throws SQLException {  
    try (Connection connection = getConnection();  
        PreparedStatement preparedStatement =  
connection.prepareStatement(INSERT_CARS_SQL)) {  
        preparedStatement.setString(1, car.getBrand());  
        preparedStatement.setString(2, car.getModel());  
        preparedStatement.setInt(3, car.getCylinder());  
        preparedStatement.setDouble(4, car.getPrice());  
        preparedStatement.executeUpdate();  
    } catch (SQLException e) {  
        printSQLException(e);  
    }  
}
```

```
public Car selectCar(int id) {  
    Car car = null;  
    try (Connection connection = getConnection();  
        PreparedStatement preparedStatement =  
connection.prepareStatement(SELECT_CAR_BY_ID)) {  
        preparedStatement.setInt(1, id);  
        ResultSet rs = preparedStatement.executeQuery();
```



```

        if (rs.next()) {
            String brand = rs.getString("brand");
            String model = rs.getString("model");
            int cylinder = rs.getInt("cylinder");
            double price = rs.getDouble("price");
            car = new Car(id, brand, model, cylinder, price);
        }
    } catch (SQLException e) {
        printSQLException(e);
    }
    return car;
}

```

```

public List<Car> selectAllCars() {
    List<Car> cars = new ArrayList<>();
    try (Connection connection = getConnection();
        PreparedStatement preparedStatement =
connection.prepareStatement(SELECT_ALL_CARS)) {
        ResultSet rs = preparedStatement.executeQuery();
        while (rs.next()) {
            int id = rs.getInt("id");
            String brand = rs.getString("brand");
            String model = rs.getString("model");
            int cylinder = rs.getInt("cylinder");
            double price = rs.getDouble("price");
            cars.add(new Car(id, brand, model, cylinder, price));
        }
    } catch (SQLException e) {
        printSQLException(e);
    }
    return cars;
}

```

```

public boolean deleteCar(int id) throws SQLException {
    boolean rowDeleted;
    try (Connection connection = getConnection();
        PreparedStatement statement =
connection.prepareStatement(DELETE_CARS_SQL)) {
        statement.setInt(1, id);
        rowDeleted = statement.executeUpdate() > 0;
    }
}

```

```

    }
    return rowDeleted;
}

```

```

public boolean updateCar(Car car) throws SQLException {
    boolean rowUpdated;
    try (Connection connection = getConnection();
        PreparedStatement statement =
connection.prepareStatement(UPDATE_CARS_SQL)) {
        statement.setString(1, car.getBrand());
        statement.setString(2, car.getModel());
        statement.setInt(3, car.getCylinder());
        statement.setDouble(4, car.getPrice());
        statement.setInt(5, car.getCarId());
        rowUpdated = statement.executeUpdate() > 0;
    }
    return rowUpdated;
}

```

```

private void printSQLException(SQLException ex) {
    for (Throwable e : ex) {
        if (e instanceof SQLException) {
            e.printStackTrace(System.err);
            System.err.println("SQLState: " + ((SQLException) e).getSQLState());
            System.err.println("Error Code: " + ((SQLException)
e).getErrorCode());
            System.err.println("Message: " + e.getMessage());
            Throwable t = ex.getCause();
            while (t != null) {
                System.out.println("Cause: " + t);
                t = t.getCause();
            }
        }
    }
}

```

```

public static void main(String[] args) {
    CarDAO dao = new CarDAO();
    try (Connection connection = dao.getConnection()) {
        if (connection != null) {

```

```

        System.out.println("Connected to the database!");
    try (Statement stmt = connection.createStatement()) {
        ResultSet rs = stmt.executeQuery("SELECT * FROM cars");
        while (rs.next()) {
            System.out.println("ID: " + rs.getInt("id"));
            System.out.println("Brand: " + rs.getString("brand"));
            System.out.println("Model: " + rs.getString("model"));
            System.out.println("Cylinder: " + rs.getInt("cylinder"));
            System.out.println("Price: " + rs.getDouble("price"));
        }
    }
    } else {
        System.out.println("Failed to make connection!");
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}
}

```

Car.java

```
package com.Model;
```

```

public class Car {
    private int carId;
    private String brand;
    private String model;
    private int cylinder;
    private double price;

    public Car(int carId, String brand, String model, int cylinder, double price) {
        this.carId = carId;
        this.brand = brand;
        this.model = model;
        this.cylinder = cylinder;
        this.price = price;
    }

    public Car(String brand, String model, int cylinder, double price) {
        this.brand = brand;
    }
}

```

```
        this.model = model;
        this.cylinder = cylinder;
        this.price = price;
    }

    public int getCarId() {
        return carId;
    }

    public void setCarId(int carId) {
        this.carId = carId;
    }

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getCylinder() {
        return cylinder;
    }

    public void setCylinder(int cylinder) {
        this.cylinder = cylinder;
    }

    public double getPrice() {
        return price;
    }
}
```

```

    public void setPrice(double price) {
        this.price = price;
    }
}

```

CarServlet.java

```

package com.DAO;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.Model.Car;
import java.sql.Statement;

public class CarDAO {
    private String jdbcurl = "jdbc:mysql://localhost:3306/carshop";
    private String jdbcUsername = "root";
    private String jdbcPassword = "admin01";

    private static final String INSERT_CARS_SQL = "INSERT INTO cars (brand,
model, cylinder, price) VALUES (?, ?, ?, ?)";
    private static final String SELECT_CAR_BY_ID = "SELECT id, brand,
model, cylinder, price FROM cars WHERE id = ?";
    private static final String SELECT_ALL_CARS = "SELECT * FROM cars";
    private static final String DELETE_CARS_SQL = "DELETE FROM cars
WHERE id = ?";
    private static final String UPDATE_CARS_SQL = "UPDATE cars SET brand
= ?, model = ?, cylinder = ?, price = ? WHERE id = ?";

    public CarDAO() {}

    protected Connection getConnection() {
        Connection connection = null;
        try {

```

```

        Class.forName("com.mysql.jdbc.Driver");
        connection = DriverManager.getConnection(jdbcurl, jdbcUsername,
jdbcPassword);
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    }
    return connection;
}

```

```

public void insertCar(Car car) throws SQLException {
    try (Connection connection = getConnection();
        PreparedStatement preparedStatement =
connection.prepareStatement(INSERT_CARS_SQL)) {
        preparedStatement.setString(1, car.getBrand());
        preparedStatement.setString(2, car.getModel());
        preparedStatement.setInt(3, car.getCylinder());
        preparedStatement.setDouble(4, car.getPrice());
        preparedStatement.executeUpdate();
    } catch (SQLException e) {
        printSQLException(e);
    }
}

```

```

public Car selectCar(int id) {
    Car car = null;
    try (Connection connection = getConnection();
        PreparedStatement preparedStatement =
connection.prepareStatement(SELECT_CAR_BY_ID)) {
        preparedStatement.setInt(1, id);
        ResultSet rs = preparedStatement.executeQuery();
        if (rs.next()) {
            String brand = rs.getString("brand");
            String model = rs.getString("model");
            int cylinder = rs.getInt("cylinder");
            double price = rs.getDouble("price");
            car = new Car(id, brand, model, cylinder, price);
        }
    } catch (SQLException e) {
        printSQLException(e);
    }
}

```

```
    return car;
}
```

```
public List<Car> selectAllCars() {
    List<Car> cars = new ArrayList<>();
    try (Connection connection = getConnection();
        PreparedStatement preparedStatement =
connection.prepareStatement(SELECT_ALL_CARS)) {
        ResultSet rs = preparedStatement.executeQuery();
        while (rs.next()) {
            int id = rs.getInt("id");
            String brand = rs.getString("brand");
            String model = rs.getString("model");
            int cylinder = rs.getInt("cylinder");
            double price = rs.getDouble("price");
            cars.add(new Car(id, brand, model, cylinder, price));
        }
    } catch (SQLException e) {
        printSQLException(e);
    }
    return cars;
}
```

```
public boolean deleteCar(int id) throws SQLException {
    boolean rowDeleted;
    try (Connection connection = getConnection();
        PreparedStatement statement =
connection.prepareStatement(DELETE_CARS_SQL)) {
        statement.setInt(1, id);
        rowDeleted = statement.executeUpdate() > 0;
    }
    return rowDeleted;
}
```

```
public boolean updateCar(Car car) throws SQLException {
    boolean rowUpdated;
    try (Connection connection = getConnection();
        PreparedStatement statement =
connection.prepareStatement(UPDATE_CARS_SQL)) {
        statement.setString(1, car.getBrand());
    }
}
```

```

        statement.setString(2, car.getModel());
        statement.setInt(3, car.getCylinder());
        statement.setDouble(4, car.getPrice());
        statement.setInt(5, car.getCarId());
        rowUpdated = statement.executeUpdate() > 0;
    }
    return rowUpdated;
}

```

```

private void printSQLException(SQLException ex) {
    for (Throwable e : ex) {
        if (e instanceof SQLException) {
            e.printStackTrace(System.err);
            System.err.println("SQLState: " + ((SQLException) e).getSQLState());
            System.err.println("Error Code: " + ((SQLException)
e).getErrorCode());
            System.err.println("Message: " + e.getMessage());
            Throwable t = ex.getCause();
            while (t != null) {
                System.out.println("Cause: " + t);
                t = t.getCause();
            }
        }
    }
}

```

```

public static void main(String[] args) {
    CarDAO dao = new CarDAO();
    try (Connection connection = dao.getConnection()) {
        if (connection != null) {
            System.out.println("Connected to the database!");
            try (Statement stmt = connection.createStatement()) {
                ResultSet rs = stmt.executeQuery("SELECT * FROM cars");
                while (rs.next()) {
                    System.out.println("ID: " + rs.getInt("id"));
                    System.out.println("Brand: " + rs.getString("brand"));
                    System.out.println("Model: " + rs.getString("model"));
                    System.out.println("Cylinder: " + rs.getInt("cylinder"));
                    System.out.println("Price: " + rs.getDouble("price"));
                }
            }
        }
    }
}

```



```

    }
    } else {
        System.out.println("Failed to make connection!");
    }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    }
}

```

AddCar.jsp

```

<% @page contentType="text/html" pageEncoding="UTF-8"% >
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <head>
        <title>Add Car</title>
    </head>
    <body>
        <h1>Add Car</h1>
        <form action="insert" method="post">
            <label>Brand:</label>
            <input type="text" value="<c:out value='${car.brand}'/>" class="form-
control" name="brand" required="required">

            <label>Model:</label>
            <input type="text" value="<c:out value='${car.model}'/>" class="form-
control" name="model" required="required">

            <label>Cylinder:</label>
            <input type="number" value="<c:out value='${car.cylinder}'/>"
class="form-control" name="cylinder" required="required">

            <label>Price:</label>
            <input type="number" step="0.01" value="<c:out
value='${car.price}'/>" class="form-control" name="price"
required="required">

            <button type="submit" class="btn btn-success">Save</button>
        </form>
    </body>
</html>

```

←

↻

🔍

localhost:8080/CarShop/new

Add Car

Brand: Model: Cylinder: Price: Save

[Back to List](#)

NurSyaza Amira binti Selamat @2024

```
<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<% @page contentType="text/html" pageEncoding="UTF-8"%>
<html>
    <head>
        <title>Car List</title>
    </head>
    <body>
        <h1>Car List</h1>
        <table border="1">
            <tr>
                <th>Car ID</th>
                <th>Brand</th>
                <th>Model</th>
                <th>Cylinder</th>
                <th>Price</th>
                <th>Actions</th>
            </tr>
            <c:forEach var="car" items="${listCar}">
                <tr>
                    <td><c:out value="${ car.carId}" /></td>
                    <td><c:out value="${ car.brand}" /></td>
                    <td><c:out value="${ car.model}" /></td>
                    <td><c:out value="${ car.cylinder}" /></td>
                    <td><c:out value="${ car.price}" /></td>
                    <td>
                        <a href="edit?id=<c:out value='${ car.carId}' />">Edit</a>

```

```

        <a href="delete?id=<c:out value='${ car.carId }'/">Delete</a>
    </td>
</tr>
</c:forEach>
</table>
<a href="new">Add New Car</a>
</body>
<footer>NurSyaza Amira binti Selamat @2024</footer>
</html>

```

Add Car

Brand: Model: Cylinder: Price:

[Back to List](#)
NurSyaza Amira binti Selamat @2024

Car List

Car ID	Brand	Model	Cylinder	Price	Actions
1	Produa	axia	1	60000.0	Edit Delete
2	Produa	Myvi	2	65000.0	Edit Delete
3	Toyota	Vios	2	63000.0	Edit Delete
4	Proton	Persona	2	55000.0	Edit Delete
5	Honda	BRV	1	67000.0	Edit Delete
6	Honda	Civic	2	82000.0	Edit Delete
7	Proton	Saga	1	45000.0	Edit Delete

[Add New Car](#)
NurSyaza Amira binti Selamat @2024

Add Car

Brand: Model: Cylinder: Price:

[Back to List](#)
NurSyaza Amira binti Selamat @2024

7	Proton	Saga	1	45000.0	Edit	Delete
8	Proton	Saga-x	1	43000.0	Edit	Delete

[Add New Car](#)

NurSyaza Amira binti Selamat @2024

Car List

Car ID	Brand	Model	Cylinder	Price	Actions	
1	Produa	axia	1	60000.0	Edit	Delete
2	Produa	Myvi	2	65000.0	Edit	Delete
3	Toyota	Vios	2	63000.0	Edit	Delete
4	Proton	Persona	2	55000.0	Edit	Delete
5	Honda	BRV	1	67000.0	Edit	Delete
6	Honda	Civic	2	82000.0	Edit	Delete

[Add New Car](#)

NurSyaza Amira binti Selamat @2024