



---

**CSM3023 WEB BASED APPLICATION DEVELOPMENT (K1)**

---

**BACHELOR OF COMPUTER SCIENCE (MOBILE COMPUTING) WITH  
HONORS**

**SEMESTER 2 2023/2024**

**LAB 8 – An MVC Example with Servlets and JSP**

**Prepared by:**

**MUHAMMAD HAZIQ AIMAN BIN MUSTAFA (S67978)**

## Coding:

### Employee.java

```
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     this.id = id;
25     this.name = name;
26     this.email = email;
27     this.position = position;
28 }
29
30 public int getId() {
31     return id;
32 }
33
34 public void setId(int id) {
35     this.id = id;
36 }
37
38 public String getName() {
39     return name;
40 }
41
42 public void setName(String name) {
43     this.name = name;
44 }
45
46 public String getEmail() {
47     return email;
48 }
49
50 public void setEmail(String email) {
51     this.email = email;
52 }
53
54 public String getPosition() {
55     return position;
56 }
57
58 public void setPosition(String position) {
59     this.position = position;
60 }
```

### EmployeeDAO.java

```
1 public class EmployeeDAO {
2     Connection connection = null;
3     private String jdbcURL = "jdbc:mysql://localhost:3306/company?";
4     private String jdbcUsername = "root";
5     private String jdbcPassword = "root";
6
7     private static final String INSERT_EMPLOYEE_SQL = "INSERT INTO employees (name, email, position) VALUES (?, ?, ?)";
8     private static final String SELECT_EMPLOYEE_BY_ID_SQL = "SELECT id, name, email, position FROM employees WHERE id=?";
9     private static final String SELECT_ALL_EMPLOYEES_SQL = "SELECT * FROM employees";
10    private static final String SELECT_EMPLOYEE_BY_EMAIL_SQL = "SELECT * FROM employees WHERE email=?";
11    private static final String UPDATE_EMPLOYEE_SQL = "UPDATE employees SET name=?, email=?, position=? WHERE id=?";
12
13    public EmployeeDAO() {}
14
15    private Connection getConnection() {
16        Connection connection = null;
17        try {
18            Class.forName("com.mysql.cj.jdbc.Driver");
19            connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
20            System.out.println("Database connection established");
21        } catch (ClassNotFoundException e) {
22            e.printStackTrace();
23        } catch (SQLException e) {
24            e.printStackTrace();
25        }
26        return connection;
27    }
28
29    public void insertEmployee(Employee employee) throws SQLException {
30        System.out.println("Inserting Employee");
31        try {
32            Connection connection = getConnection();
33            PreparedStatement preparedStatement = connection.prepareStatement(INSERT_EMPLOYEE_SQL);
34            preparedStatement.setString(1, employee.getName());
35            preparedStatement.setString(2, employee.getEmail());
36            preparedStatement.setString(3, employee.getPosition());
37            System.out.println("Prepared statement");
38            preparedStatement.executeUpdate();
39        } catch (SQLException e) {
40            e.printStackTrace();
41        }
42    }
43
44    public Employee selectEmployee(int id) {
45        Employee employee = null;
46        try {
47            Connection connection = getConnection();
48            PreparedStatement preparedStatement = connection.prepareStatement(SELECT_EMPLOYEE_BY_ID_SQL);
49            preparedStatement.setInt(1, id);
50            System.out.println("Prepared statement");
51            ResultSet rs = preparedStatement.executeQuery();
52
53            while(rs.next()) {
54                String name = rs.getString("name");
55                String email = rs.getString("email");
56                String position = rs.getString("position");
57                employee = new Employee(id, name, email, position);
58            }
59        } catch (SQLException e) {
60            e.printStackTrace();
61        }
62        return employee;
63    }
64
65    public List < Employee > selectAllEmployees() {
66        List < Employee > employees = new ArrayList < > ();
67        try {
68            Connection connection = getConnection();
69            PreparedStatement preparedStatement = connection.prepareStatement(SELECT_ALL_EMPLOYEES_SQL);
70            System.out.println("Prepared statement");
71            ResultSet rs = preparedStatement.executeQuery();
72
73            while(rs.next()) {
74                int id = rs.getInt("id");
75                String name = rs.getString("name");
76                String email = rs.getString("email");
77                String position = rs.getString("position");
78                employee = new Employee(id, name, email, position);
79                employees.add(employee);
80            }
81        } catch (SQLException e) {
82            e.printStackTrace();
83        }
84        return employees;
85    }
86
87    public boolean deleteEmployee(int id) throws SQLException {
88        boolean rowDeleted;
89        try {
90            Connection connection = getConnection();
91            PreparedStatement statement = connection.prepareStatement(DELETE_EMPLOYEE_SQL);
92            statement.setInt(1, id);
93            rowDeleted = statement.executeUpdate() > 0;
94        } catch (SQLException e) {
95            e.printStackTrace();
96        }
97        return rowDeleted;
98    }
99
100    public boolean updateEmployee(Employee employee) throws SQLException {
101        boolean rowUpdated;
102        try {
103            Connection connection = getConnection();
104            PreparedStatement statement = connection.prepareStatement(UPDATE_EMPLOYEE_SQL);
105            statement.setString(1, employee.getName());
106            statement.setString(2, employee.getEmail());
107            statement.setString(3, employee.getPosition());
108            statement.setInt(4, employee.getId());
109            rowUpdated = statement.executeUpdate() > 0;
110        } catch (SQLException e) {
111            e.printStackTrace();
112        }
113        return rowUpdated;
114    }
115
116    private void printSQLException(SQLException ex) {
117        for (Throwable e: ex) {
118            if (e instanceof SQLException) {
119                System.out.println(System.err);
120                System.out.println("SQLState: " + ((SQLException) e).getSQLState());
121                System.out.println("Error Code: " + ((SQLException) e).getErrorCode());
122                System.out.println("Message: " + e.getMessage());
123                Throwable t = ex.getCause();
124                while (t != null) {
125                    System.out.println("Cause: " + t);
126                    t = t.getCause();
127                }
128            }
129        }
130    }
131 }
```

### EmployeeServlet.java

## EmployeeForm.jsp

[illegible]

## EmployeeList.jsp

```

1 <!--Employee contentTypes="text/html" contentType="text/html; charset=UTF-8"
2 <$english.wiz="http://www.ibm.com/csp/491/csp" prefill="c" >
3 <DOCTYPE html>
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 <title>Employee Management System</title>
8 <link rel="stylesheet" href="http://www.ibm.com/csp/491/csp" type="text/css">
9 <script type="text/javascript" src="http://www.ibm.com/csp/491/csp" type="text/javascript">
10 </script>
11 </head>
12 <body>
13 <div class="header" style="background-color: tomato;">
14 <div>
15 <a href="#" class="header-brand"> Employee Management App </a>
16 </div>
17 <div class="header-nav">
18 <div>
19 <a href="#">Request GETContentPath{/k/list} Class="new-link">Employees</a>{/k/list}
20 </div>
21 </div>
22 </header>
23 <div class="new">
24 <div class="container">
25 <div class="container">List of Employees</div>
26 <div class="container">test</div>
27 <div class="container">test</div>
28 </div>
29 <table class="table table-bordered">
30 <thead>
31 <tr>
32 <th>ID</th>
33 <th>Name</th>
34 <th>Email</th>
35 <th>Position</th>
36 <th>Actions</th>
37 </tr>
38 </thead>
39 <tbody>
40 <tr>
41 <td>{out value=$employee.id}</td>
42 <td>{out value=$employee.name}</td>
43 <td>{out value=$employee.email}</td>
44 <td>{out value=$employee.position}</td>
45 <td>
46 <a href="#">Edit</a>{out value=$employee.id} </td>
47 <a href="#">Delete</a>{out value=$employee.id} </td>
48 </td>
49 </tr>
50 </tbody>
51 </table>
52 </div>
53 </body>
54 </html>

```

## Index.jsp

```

Document : index
Created on : 6 Jun 2024, 12:52:38 am
Author : Lenevo

-->

<!page contentType="text/html" pageEncoding="UTF-8">
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>User Management Application</title>
<link rel="stylesheet" href="http://localhost:8080/userManagement/css/bootstrap/4.3.1/css/bootstrap.min.css">
<script src="http://localhost:8080/userManagement/js/bootstrap/4.3.1/js/bootstrap.min.js" crossorigin="anonymous"></script>
</head>
<body>
<h1>Application MVC system for Employee Management</h1>

<div>
<div>
<div><a href="http://localhost:8080/EmployeeManagement/list">All Employee</a> </div>
<div><a href="http://localhost:8080/EmployeeManagement/new">Add a New Employee</a> </div>
<div><a href="http://localhost:8080/EmployeeManagement/list">Edit Employee</a> </div>
</div>
</div>
</body>
</html>

```

Error.jsp

```
1 <%--
2 Document : error
3 Created on : 6 Jun 2024, 12:47:29 am
4 Author : Lenovo
5 --%>
6
7 <%@page contentType="text/html" pageEncoding="UTF-8" isErrorPage="true" %>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12 <title>Error page</title>
13 </head>
14 <body>
15 <center>
16 <h1>Error</h1>
17 <h2><%=exception.getMessage() %><br/></h2>
18 </center>
19 </body>
20 </html>
21
```

Output:

Application MVC system for Employee Management

- [All Employee List](#)
- [Add a New Employee](#)
- [Edit Employee](#)

Employee Management App Employees

List of Employees

Add New Employee

ID	Name	Email	Position	Actions
1	abe	s67978@ocean.unt.edu.my	Manager	<a href="#">Edit</a> <a href="#">Delete</a>
2	Ryoki Tenkai	haziqaiman34@gmail.com	Head of Dept	<a href="#">Edit</a> <a href="#">Delete</a>
3	Ziqman Al-Attas	ziziq34@icloud.com	Head of Dept	<a href="#">Edit</a> <a href="#">Delete</a>

Add New Employee

Employee Name

Hazim

Employee Email

jimjim77@gmail.com

Employee Position

Supervisor

Supervisor

Save

## Exercise

### Coding:

#### Car.java

```
17 public class Car {
18     protected int car_id;
19     protected String brand;
20     protected String model;
21     protected int cylinder;
22     protected double price;
23
24     public Car() {}
25
26     public Car(String brand, String model, int cylinder, double price) {
27         super();
28         this.brand = brand;
29         this.model = model;
30         this.cylinder = cylinder;
31         this.price = price;
32     }
33
34     public Car(int car_id, String brand, String model, int cylinder, double price) {
35         this.car_id = car_id;
36         this.brand = brand;
37         this.model = model;
38         this.cylinder = cylinder;
39         this.price = price;
40     }
41
42     public int getCar_id() {
43         return car_id;
44     }
45
46     public void setCar_id(int car_id) {
47         this.car_id = car_id;
48     }
49
50     public String getBrand() {
51         return brand;
52     }
53
54     public void setBrand(String brand) {
55         this.brand = brand;
56     }
57
58     public String getModel() {
59         return model;
60     }
61
62     public void setModel(String model) {
63         this.model = model;
64     }
65
66     public void setCylinder(int cylinder) {
67         this.cylinder = cylinder;
68     }
69
70     public double getPrice() {
71         return price;
72     }
73
74     public void setPrice(double price) {
75         this.price = price;
76     }
77 }
```

#### CarDAO.java

```

17 import java.sql.*;
18 import java.sql.SQLException;
19 import java.util.*;
20 import com.model.Car;
21
22 public class CarDAO {
23     Connection connection = null;
24     private String dbName = "jdbc:mysql://localhost:3306/carshop?";
25     private String dbNameUser = "root?";
26     private String dbNamePasswd = "admin?";
27
28     private static final String INSERT_CAR_SQL = "INSERT INTO carshoplist (brand, model, cylinder, price) VALUES (?, ?, ?, ?)";
29     private static final String SELECT_CAR_BY_ID = "select car_id, brand, model, cylinder, price from carshoplist where car_id=?";
30     private static final String SELECT_ALL_CAR = "select * from carshoplist";
31     private static final String DELETE_CAR_SQL = "delete from carshoplist where car_id = ?";
32     private static final String UPDATE_CAR_SQL = "update carshoplist set brand = ?,model = ?,cylinder = ?, price = ? where car_id = ?";
33
34     public CarDAO() {}
35
36     protected Connection getConnection() {
37         Connection connection = null;
38         try {
39             Class.forName("com.mysql.jdbc.Driver");
40             connection = DriverManager.getConnection(dbName, dbNameUser, dbNamePasswd);
41             System.out.println("Database connected");
42             return connection;
43         } catch (SQLException e) {
44             e.printStackTrace();
45             return null;
46         }
47     }
48
49     public void insertCar(Car car) throws SQLException {
50         System.out.println(INSERT_CAR_SQL);
51         try (Connection connection = getConnection(); PreparedStatement preparedStatement =
52             connection.prepareStatement(INSERT_CAR_SQL)) {
53             preparedStatement.setString(1, car.getBrand());
54             preparedStatement.setString(2, car.getModel());
55             preparedStatement.setInt(3, car.getCylinder());
56             preparedStatement.setDouble(4, car.getPrice());
57             System.out.println(preparedStatement);
58             preparedStatement.executeUpdate();
59         } catch (SQLException e) {
60             e.printStackTrace();
61         }
62     }
63
64     public Car selectCar(int id) {
65         Car car = null;
66         // Step 1: Establishing a Connection
67         try (Connection connection = getConnection();
68             // Step 2: Create a statement using connection
69             PreparedStatement preparedStatement = connection.prepareStatement(SELECT_CAR_BY_ID);) {
70             preparedStatement.setInt(1, id);
71             System.out.println(preparedStatement);
72             ResultSet rs = preparedStatement.executeQuery();
73
74             while(rs.next()) {
75                 String brand = rs.getString("brand");
76                 String model = rs.getString("model");
77                 int cylinder = rs.getInt("cylinder");
78                 double price = rs.getDouble("price");
79                 car = new Car(id, brand, model, cylinder, price);
80             }
81         } catch (SQLException e) {
82             e.printStackTrace();
83         }
84         return car;
85     }
86
87     public List < Car > selectAllCars() {
88         List < Car > cars = new ArrayList <>();
89         try (Connection connection = getConnection();
90             PreparedStatement preparedStatement = connection.prepareStatement(SELECT_ALL_CAR);) {
91             System.out.println(preparedStatement);
92             ResultSet rs = preparedStatement.executeQuery();
93
94             while(rs.next()) {
95                 int id = rs.getInt("car_id");
96                 String brand = rs.getString("brand");
97                 String model = rs.getString("model");
98                 int cylinder = rs.getInt("cylinder");
99                 double price = rs.getDouble("price");
100                 cars.add(new Car(id, brand, model, cylinder, price));
101             }
102         } catch (SQLException e) {
103             e.printStackTrace();
104         }
105         return cars;
106     }
107
108     public boolean deleteCar(int id) throws SQLException {
109         boolean rowDeleted;
110         try (Connection connection = getConnection(); PreparedStatement statement =
111             connection.prepareStatement(DELETE_CAR_SQL);) {
112             statement.setInt(1, id);
113             rowDeleted = statement.executeUpdate() > 0;
114         }
115         return rowDeleted;
116     }
117
118     public boolean updateCar(Car car) throws SQLException {
119         boolean rowUpdated;
120         try (Connection connection = getConnection(); PreparedStatement statement =
121             connection.prepareStatement(UPDATE_CAR_SQL);) {
122             statement.setString(1, car.getBrand());
123             statement.setString(2, car.getModel());
124             statement.setInt(3, car.getCylinder());
125             statement.setDouble(4, car.getPrice());
126             statement.setInt(5, car.getCar_id());
127
128             rowUpdated = statement.executeUpdate() > 0;
129         }
130         return rowUpdated;
131     }
132
133     private void printSQLException(SQLException ex) {
134         for (Throwable e: ex) {
135             if (e instanceof SQLException) {
136                 e.printStackTrace(System.err);
137                 System.err.println("SQLState: " + ((SQLException) e).getSQLState());
138                 System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
139                 System.err.println("Message: " + e.getMessage());
140                 Throwable t = ex.getCause();
141                 while (t != null) {
142                     System.out.println("Cause: " + t);
143                     t = t.getCause();
144                 }
145             }
146         }
147     }
148 }

```

CarServlet.java

```

7 import com.DAO.CarDAO;
8 import com.model.Car;
9 import jakarta.servlet.RequestDispatcher;
10 import java.io.IOException;
11 import java.io.PrintWriter;
12 import jakarta.servlet.ServletException;
13 import jakarta.servlet.annotation.WebServlet;
14 import jakarta.servlet.http.HttpServlet;
15 import jakarta.servlet.http.HttpServletRequest;
16 import jakarta.servlet.http.HttpServletResponse;
17 import java.sql.SQLException;
18 import java.util.List;
19
20 /**
21  *
22  * @author Ienovo
23  */
24 @WebServlet("/")
25 public class CarServlet extends HttpServlet {
26
27     /**
28      * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
29      * methods.
30      *
31      * @param request servlet request
32      * @param response servlet response
33      * @throws ServletException if a servlet-specific error occurs
34      * @throws IOException if an I/O error occurs
35      */
36     private CarDAO carDAO;
37
38     @Override
39     public void init() {
40         carDAO = new CarDAO();
41     }
42
43     @Override
44     protected void doGet(HttpServletRequest request, HttpServletResponse response)
45         throws ServletException, IOException {
46         String action = request.getServletPath();
47
48         try {
49             switch (action) {
50                 case "/new":
51                     showNewForm(request, response);
52                     break;
53                 case "/insert":
54                     insertCar(request, response);
55                     break;
56                 case "/delete":
57                     deleteCar(request, response);
58                     break;
59                 case "/edit":
60                     showEditForm(request, response);
61                     break;
62                 case "/update":
63                     updateCar(request, response);
64                     break;
65                 default:
66                     listCar(request, response);
67                     break;
68             }
69         } catch (SQLException ex) {
70             throw new ServletException(ex);
71         }
72     }
73
74     private void listCar(HttpServletRequest request, HttpServletResponse response)
75         throws ServletException, IOException, ServletException {
76         List < Car > listCar = carDAO.selectAllCars();
77         request.setAttribute("listCar", listCar);
78         RequestDispatcher dispatcher = request.getRequestDispatcher("CarList.jsp");
79         dispatcher.forward(request, response);
80     }
81
82     private void showNewForm(HttpServletRequest request, HttpServletResponse response)
83         throws ServletException, IOException {
84         RequestDispatcher dispatcher = request.getRequestDispatcher("CarForm.jsp");
85         dispatcher.forward(request, response);
86     }
87
88     private void updateCar(HttpServletRequest request, HttpServletResponse response)
89         throws ServletException, IOException {
90         int id = Integer.parseInt(request.getParameter("car_id"));
91         String brand = request.getParameter("brand");
92         String model = request.getParameter("model");
93         int cylinder = Integer.parseInt(request.getParameter("cylinder"));
94         double price = Double.parseDouble(request.getParameter("price"));
95         Car car = new Car(brand, model, cylinder, price);
96         carDAO.updateCar(car);
97         response.sendRedirect("listcar");
98     }
99
100     private void deleteCar(HttpServletRequest request, HttpServletResponse response)
101         throws ServletException, IOException {
102         int id = Integer.parseInt(request.getParameter("car_id"));
103         carDAO.deleteCar(id);
104         response.sendRedirect("listcar");
105     }
106
107     /**
108      * Handles the HTTP <code>POST</code> method.
109      *
110      * @param request servlet request
111      * @param response servlet response
112      * @throws ServletException if a servlet-specific error occurs
113      * @throws IOException if an I/O error occurs
114      */
115     @Override
116     protected void doPost(HttpServletRequest request, HttpServletResponse response)
117         throws ServletException, IOException {
118         doGet(request, response);
119     }
120
121     /**
122      * Returns a short description of the servlet.
123      *
124      * @return a String containing servlet description
125      */
126     @Override
127     public String getServletInfo() {
128         return "Short description";
129     }
130 }

```

CarForm.jsp

[illegible][illegible]

## CarList.jsp

[illegible]



## Index.jsp

```
<%%page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Car Shop 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/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
</head>
<body>
<h1>Car Shop Management System - Encik Ayah Used Car</h1><br>
<ul>
<li><a href="http://localhost:8080/CarShop_Management/listcar">All Car List</a></li>
<li><a href="http://localhost:8080/CarShop_Management/new">Add a New Car</a></li>
<li><a href="http://localhost:8080/CarShop_Management/listcar">Edit Car</a></li>
</ul>
</body>
</html>
```

## Error.jsp

```
<%%page contentType="text/html" pageEncoding="UTF-8" isErrorPage="true" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Error page</title>
</head>
<body>
<center>
<h1>Error</h1>
<h2><%=exception.getMessage() %><br/></h2>
</center>
</body>
</html>
```

Output:

## Car Shop Management System - Encik Ayah Used Car

- [All Car List](#)
- [Add a New Car](#)
- [Edit Car](#)

### List of Cars

[Add New Car](#)

ID	Brand	Model	Engine Cylinder	Price	Actions
4	Toyota	X70	2	55000.0	<a href="#">Edit</a> <a href="#">Delete</a>
5	Toyota	X70	2	73000.0	<a href="#">Edit</a> <a href="#">Delete</a>

### Add New Car

Car Brand

Toyota

Toyota

Car Model

Yaris GR

Engine Cylinder

3

Car Price

180000.00

[Save](#)