



---

**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 HARITH BIN ZULKIFLI (S67335)**

## Coding:

### Employee.java

```
1 public Employee() {}
2
3 public Employee(String name, String email, String position) {
4     super();
5     this.name = name;
6     this.email = email;
7     this.position = position;
8 }
9
10 public Employee(int id, String name, String email, String position) {
11     this.id = id;
12     this.name = name;
13     this.email = email;
14     this.position = position;
15 }
16
17 public int getId() {
18     return id;
19 }
20
21 public void setId(int id) {
22     this.id = id;
23 }
24
25 public String getName() {
26     return name;
27 }
28
29 public void setName(String name) {
30     this.name = name;
31 }
32
33 public String getEmail() {
34     return email;
35 }
36
37 public void setEmail(String email) {
38     this.email = email;
39 }
40
41 public String getPosition() {
42     return position;
43 }
44
45 public void setPosition(String position) {
46     this.position = position;
47 }
```

### 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_ALL_EMPLOYEES_SQL = "SELECT * FROM employees WHERE id=?";
9     private static final String SELECT_BY_ID_EMPLOYEE_SQL = "SELECT * FROM employees WHERE id=?";
10    private static final String UPDATE_EMPLOYEE_SQL = "UPDATE employees SET name=?, email=?, position=? WHERE id=?";
11
12    public EmployeeDAO() {}
13
14    protected Connection getConnection() {
15        Connection connection = null;
16        try {
17            Class.forName("com.mysql.jdbc.Driver");
18            connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
19            System.out.println("Database connected");
20        } catch (Exception e) {
21            e.printStackTrace();
22        }
23        return connection;
24    }
25
26    public void insertEmployee(Employee employee) throws SQLException {
27        System.out.println("Inserting Employee");
28        try (Connection connection = getConnection(); PreparedStatement preparedStatement =
29            connection.prepareStatement(INSERT_EMPLOYEE_SQL)) {
30            preparedStatement.setString(1, employee.getName());
31            preparedStatement.setString(2, employee.getEmail());
32            preparedStatement.setString(3, employee.getPosition());
33            System.out.println("PreparedStatement");
34            preparedStatement.executeUpdate();
35        } catch (SQLException e) {
36            e.printStackTrace();
37        }
38    }
39
40    public Employee selectEmployee(int id) {
41        Employee employee = null;
42        try (Connection connection = getConnection();
43            Statement statement = connection.createStatement()) {
44            PreparedStatement preparedStatement = connection.prepareStatement(SELECT_BY_ID_EMPLOYEE_SQL);
45            preparedStatement.setInt(1, id);
46            System.out.println("PreparedStatement");
47            ResultSet rs = preparedStatement.executeQuery();
48
49            while (rs.next()) {
50                String name = rs.getString("name");
51                String email = rs.getString("email");
52                String position = rs.getString("position");
53                employee = new Employee(id, name, email, position);
54            }
55        } catch (SQLException e) {
56            e.printStackTrace();
57        }
58        return employee;
59    }
60
61    public List<Employee> selectAllEmployees() {
62        List<Employee> employees = new ArrayList<>();
63        try (Connection connection = getConnection();
64            PreparedStatement preparedStatement = connection.prepareStatement(SELECT_ALL_EMPLOYEES_SQL)) {
65            System.out.println("PreparedStatement");
66            ResultSet rs = preparedStatement.executeQuery();
67
68            while (rs.next()) {
69                int id = rs.getInt("id");
70                String name = rs.getString("name");
71                String email = rs.getString("email");
72                String position = rs.getString("position");
73                employees.add(new Employee(id, name, email, position));
74            }
75        } catch (SQLException e) {
76            e.printStackTrace();
77        }
78        return employees;
79    }
80
81    public boolean deleteEmployee(int id) throws SQLException {
82        boolean rowDeleted;
83        try (Connection connection = getConnection(); PreparedStatement statement =
84            connection.prepareStatement(DELETE_BY_ID_EMPLOYEE_SQL)) {
85            statement.setInt(1, id);
86            rowDeleted = statement.executeUpdate() > 0;
87        }
88        return rowDeleted;
89    }
90
91    public boolean updateEmployee(Employee employee) throws SQLException {
92        boolean rowUpdated;
93        try (Connection connection = getConnection(); PreparedStatement statement =
94            connection.prepareStatement(UPDATE_EMPLOYEE_SQL)) {
95            statement.setString(1, employee.getName());
96            statement.setString(2, employee.getEmail());
97            statement.setString(3, employee.getPosition());
98            statement.setInt(4, employee.getId());
99            rowUpdated = statement.executeUpdate() > 0;
100        }
101    }
102
103    private void printSQLException(SQLException ex) {
104        for (Throwable t : ex) {
105            if (t instanceof SQLException) {
106                System.out.println("Cause: " + t);
107                System.out.println("SQLState: " + ((SQLException) t).getSQLState());
108                System.out.println("Error Code: " + ((SQLException) t).getErrorCode());
109                System.out.println("Message: " + t.getMessage());
110                Throwable t2 = t.getCause();
111                while (t2 != null) {
112                    System.out.println("Cause: " + t2);
113                    t2 = t2.getCause();
114                }
115            }
116        }
117    }
118 }
```

### EmployeeServlet.java

## EmployeeForm.jsp

```

<!-->
<script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
</script>
<!-->
<script>
    $(document).ready(function() {
        // Fetch data from the database
        $.ajax({
            url: 'http://localhost:8080/employees',
            type: 'GET',
            success: function(data) {
                // Display data in the table
                $('#employees').empty();
                $.each(data, function(index, employee) {
                    $('#employees').append(
                        '<tr>
                            <td>' + employee.id + '</td>
                            <td>' + employee.name + '</td>
                            <td>' + employee.position + '</td>
                            <td>' + employee.salary + '</td>
                        </tr>'
                    );
                });
            },
            error: function(xhr) {
                console.log('Error: ' + xhr.responseText);
            }
        });

        // Add new employee
        $('#addEmployee').click(function() {
            var name = $('#name').val();
            var position = $('#position').val();
            var salary = $('#salary').val();

            $.ajax({
                url: 'http://localhost:8080/employees',
                type: 'POST',
                data: {
                    'name': name,
                    'position': position,
                    'salary': salary
                },
                success: function(data) {
                    // Refresh the table
                    $.ajax({
                        url: 'http://localhost:8080/employees',
                        type: 'GET',
                        success: function(data) {
                            $('#employees').empty();
                            $.each(data, function(index, employee) {
                                $('#employees').append(
                                    '<tr>
                                        <td>' + employee.id + '</td>
                                        <td>' + employee.name + '</td>
                                        <td>' + employee.position + '</td>
                                        <td>' + employee.salary + '</td>
                                    </tr>'
                                );
                            });
                        },
                        error: function(xhr) {
                            console.log('Error: ' + xhr.responseText);
                        }
                    });
                },
                error: function(xhr) {
                    console.log('Error: ' + xhr.responseText);
                }
            });
        });

        // Edit employee
        $('#editEmployee').click(function() {
            var id = $('#id').val();
            var name = $('#name').val();
            var position = $('#position').val();
            var salary = $('#salary').val();

            $.ajax({
                url: 'http://localhost:8080/employees/' + id,
                type: 'PUT',
                data: {
                    'name': name,
                    'position': position,
                    'salary': salary
                },
                success: function(data) {
                    // Refresh the table
                    $.ajax({
                        url: 'http://localhost:8080/employees',
                        type: 'GET',
                        success: function(data) {
                            $('#employees').empty();
                            $.each(data, function(index, employee) {
                                $('#employees').append(
                                    '<tr>
                                        <td>' + employee.id + '</td>
                                        <td>' + employee.name + '</td>
                                        <td>' + employee.position + '</td>
                                        <td>' + employee.salary + '</td>
                                    </tr>'
                                );
                            });
                        },
                        error: function(xhr) {
                            console.log('Error: ' + xhr.responseText);
                        }
                    });
                },
                error: function(xhr) {
                    console.log('Error: ' + xhr.responseText);
                }
            });
        });

        // Delete employee
        $('#deleteEmployee').click(function() {
            var id = $('#id').val();

            $.ajax({
                url: 'http://localhost:8080/employees/' + id,
                type: 'DELETE',
                success: function(data) {
                    // Refresh the table
                    $.ajax({
                        url: 'http://localhost:8080/employees',
                        type: 'GET',
                        success: function(data) {
                            $('#employees').empty();
                            $.each(data, function(index, employee) {
                                $('#employees').append(
                                    '<tr>
                                        <td>' + employee.id + '</td>
                                        <td>' + employee.name + '</td>
                                        <td>' + employee.position + '</td>
                                        <td>' + employee.salary + '</td>
                                    </tr>'
                                );
                            });
                        },
                        error: function(xhr) {
                            console.log('Error: ' + xhr.responseText);
                        }
                    });
                },
                error: function(xhr) {
                    console.log('Error: ' + xhr.responseText);
                }
            });
        });
    });
</script>
</body>
</html>

```

## EmployeeList.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>User 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/iJTQUOhcWr7x9JvoRxT2M2w1T" crossorigin="anonymous">
</head>
<body>
<h1>Application MVC system for Employee Management</h1><br>
<ul>
<li><a href="http://localhost:8080/Employee Management/list">All Employee List</a></li>
<li><a href="http://localhost:8080/Employee Management/new">Add a New Employee</a></li>
<li><a href="http://localhost:8080/Employee Management/list">Edit Employee</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:

## Application MVC system for Employee Management

- All Employee List
- Add a New Employee
- Edit Employee

Employee Management App

### Add New Employee

Employee Name

Lutfil Haziq

Employee Email

S67911@ocean.umat.edu.my

Employee Position

Manager

Manager

Save

Employee Management App

### List of Employees

Add New Employee

ID	Name	Email	Position	Actions
1	Lutfil Haziq	S67911@ocean.umat.edu.my	Manager	<a href="#">Edit</a> <a href="#">Delete</a>
2	Ahmad Salam	salam@gmail.com	Head of Dept	<a href="#">Edit</a> <a href="#">Delete</a>

## 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

```

11 //
12 import java.sql.*;
13 import java.sql.SQLException;
14 import java.util.*;
15 import com.model.Car;
16
17 public class CarServlet {
18     Connection connection = null;
19     private String jdbcURL = "jdbc:mysql://localhost:3306/carshop";
20     private String jdbcUsername = "root";
21     private String jdbcPassword = "admin";
22
23     private static final String INSERT_CAR_SQL = "INSERT INTO carshoplist (brand, model, cylinder, price) VALUES (?, ?, ?, ?)";
24     private static final String SELECT_CAR_BY_ID_SQL = "select car_id, brand, model, cylinder, price from carshoplist where car_id=?";
25     private static final String SELECT_ALL_CAR_SQL = "select * from carshoplist";
26     private static final String DELETE_CAR_SQL = "delete from carshoplist where car_id = ?";
27     private static final String UPDATE_CAR_SQL = "update carshoplist set brand = ?, model = ?, cylinder = ?, price = ? where car_id = ?";
28
29     public CarServlet() {
30
31     }
32
33     protected Connection getConnection() {
34         Connection connection = null;
35         try {
36             Class.forName("com.mysql.jdbc.Driver");
37             connection = DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
38             System.out.println("Database connected");
39         } catch (ClassNotFoundException e) {
40             e.printStackTrace();
41         } catch (SQLException e) {
42             e.printStackTrace();
43         }
44         return connection;
45     }
46
47     public void insertCar(Car car) throws SQLException {
48         System.out.println(INSERT_CAR_SQL);
49         try (Connection connection = getConnection(); PreparedStatement preparedStatement =
50             connection.prepareStatement(INSERT_CAR_SQL)) {
51             preparedStatement.setString(1, car.getBrand());
52             preparedStatement.setString(2, car.getModel());
53             preparedStatement.setInt(3, car.getCylinder());
54             preparedStatement.setDouble(4, car.getPrice());
55             System.out.println(preparedStatement);
56             preparedStatement.executeUpdate();
57         } catch (SQLException e) {
58             printSQLException(e);
59         }
60     }
61
62     public Car selectCar(int id) {
63         Car car = null;
64         // Step 1: Establishing a Connection
65         try (Connection connection = getConnection();
66             // Step 2: create a statement using connection
67             PreparedStatement preparedStatement = connection.prepareStatement(SELECT_CAR_BY_ID_SQL);) {
68             preparedStatement.setInt(1, id);
69             System.out.println(preparedStatement);
70             ResultSet rs = preparedStatement.executeQuery();
71
72             while (rs.next()) {
73                 String brand = rs.getString("brand");
74                 String model = rs.getString("model");
75                 int cylinder = rs.getInt("cylinder");
76                 double price = rs.getDouble("price");
77                 car = new Car(id, brand, model, cylinder, price);
78             }
79         } catch (SQLException e) {
80             printSQLException(e);
81         }
82         return car;
83     }
84
85     public List<Car> selectAllCars() {
86         List<Car> cars = new ArrayList<>();
87         try (Connection connection = getConnection();
88             PreparedStatement preparedStatement = connection.prepareStatement(SELECT_ALL_CAR_SQL);) {
89             System.out.println(preparedStatement);
90             ResultSet rs = preparedStatement.executeQuery();
91
92             while (rs.next()) {
93                 int id = rs.getInt("car_id");
94                 String brand = rs.getString("brand");
95                 String model = rs.getString("model");
96                 int cylinder = rs.getInt("cylinder");
97                 double price = rs.getDouble("price");
98                 cars.add(new Car(id, brand, model, cylinder, price));
99             }
100         } catch (SQLException e) {
101             printSQLException(e);
102         }
103         return cars;
104     }
105
106     public boolean deleteCar(int id) throws SQLException {
107         boolean rowDeleted;
108         try (Connection connection = getConnection(); PreparedStatement statement =
109             connection.prepareStatement(DELETE_CAR_SQL);) {
110             statement.setInt(1, id);
111             rowDeleted = statement.executeUpdate() > 0;
112         }
113         return rowDeleted;
114     }
115
116     public boolean updateCar(Car car) throws SQLException {
117         boolean rowUpdated;
118         try (Connection connection = getConnection(); PreparedStatement statement =
119             connection.prepareStatement(UPDATE_CAR_SQL);) {
120             statement.setString(1, car.getBrand());
121             statement.setString(2, car.getModel());
122             statement.setInt(3, car.getCylinder());
123             statement.setDouble(4, car.getPrice());
124             statement.setInt(5, car.getCar_id());
125
126             rowUpdated = statement.executeUpdate() > 0;
127         }
128         return rowUpdated;
129     }
130
131     private void printSQLException(SQLException ex) {
132         for (Throwable e: ex) {
133             if (e instanceof SQLException) {
134                 e.printStackTrace(System.err);
135                 System.err.println("SQLState: " + ((SQLException) e).getSQLState());
136                 System.err.println("Error Code: " + ((SQLException) e).getErrorCode());
137                 System.err.println("Message: " + e.getMessage());
138                 Throwable t = ex.getCause();
139                 while (t != null) {
140                     System.out.println("Cause: " + t);
141                     t = t.getCause();
142                 }
143             }
144         }
145     }
146 }

```

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 Lenovo
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-ggOyR0iXCbMQV3Iipma34ND+dH/1fQ784/36cY/1jTQUOhcWz7x9JwoRxT2MZw1T" 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 -Harith

### Used Car

- All Car List
- Add a New Car
- Edit Car

Car Shop Management App					
List of Cars					
<a href="#">Add New Car</a>					
ID	Brand	Model	Engine Cylinder	Price	Actions
1	Ford	Ford Fiesta	3	80000.0	<a href="#">Edit</a> <a href="#">Delete</a>
2	Perodua	Myvi	2	40000.0	<a href="#">Edit</a> <a href="#">Delete</a>
3	Honda	Civic	4	200000.0	<a href="#">Edit</a> <a href="#">Delete</a>

Car Management Application	
<h3>Add New Car</h3>	
Car Brand	<div><div>Honda</div><div>Honda</div></div>
Car Model	<div><div>Civic</div></div>
Engine Cylinder	<div><div>4</div></div>
Car Price	<div><div>200000.00</div></div>
<div>Save</div>	