# SLOT 09_ PRJ301

## HOW TO APPLY MVC IN JAVA WEB APPLICATION

To apply the **MVC (Model-View-Controller)** architecture in a **Java web application**, you follow a structured approach where each component (Model, View, and Controller) is clearly defined and interacts with the others. Here's a step-by-step guide to implement MVC in a Java web application using **JSP**, **Servlets**, and **JDBC**:

### 1. Model (Business Logic and Data Access Layer)

The **Model** is responsible for interacting with the database and handling business logic. In a Java web application, this is typically implemented using **JavaBeans** (POJOs) and **Data Access Objects (DAO)** that connect to a database via **JDBC**.

**Example of a Product Model and ProductDAO (Data Access Object):**

```
// Product.java (Model)
public class Product {
    private int id;
    private String name;
    private double price;

    // Getters and Setters
}

// ProductDAO.java (Data Access Object)
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class ProductDAO {
    public List<Product> getAllProducts() {
        List<Product> products = new ArrayList<>();
        try (Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "user", "password");
             Statement stmt = conn.createStatement()) {
                 ResultSet rs = stmt.executeQuery("SELECT * FROM Product");
            while (rs.next()) {
                Product product = new Product();
                product.setId(rs.getInt("id"));
                product.setName(rs.getString("name"));
                product.setPrice(rs.getDouble("price"));
                products.add(product);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return products;
    }

    // Add, Update, Delete methods can be added here
}
```

**2. View (User Interface Layer)**

The **View** is responsible for displaying the data to the user.

In a Java web application, the **View** is typically implemented using **JSP (JavaServer Pages)**. JSP files are used to render dynamic content and can use **JSTL** and **Expression Language (EL)** to simplify access to data provided by the Controller.

**Example of a product-list.jsp View:**

```
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

```
<html>
<head>
    <title>Product List</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
    <h1>Product List</h1>
    <table class="table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Price</th>
            </tr>
        </thead>
        <tbody>
            <c:forEach var="product" items="${productList}">
                <tr>
                    <td>${product.id}</td>
                    <td>${product.name}</td>
                    <td>${product.price}</td>
                </tr>
            </c:forEach>
        </tbody>
    </table>
</body>
```

</html>

- This JSP file will display a list of products in a table format.
- **Expression Language (EL)** ${productList} is used to reference the list of products passed from the controller.

**3. Controller (Request Handling Layer)**

The **Controller** in a Java web application is typically implemented using **Servlets**.

The Controller receives user input (HTTP requests), interacts with the Model to retrieve or modify data, and forwards the data to the View (JSP) for display.

**Example of a ProductServlet Controller:**

```
// ProductServlet.java (Controller)

import java.io.IOException;

import javax.servlet.*;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.*;

import java.util.List;


@WebServlet("/product-list")

public class ProductServlet extends HttpServlet {

    private ProductDAO productDAO;


    public void init() {

        productDAO = new ProductDAO();

    }


    protected void doGet(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        List<Product> productList = productDAO.getAllProducts();
        request.setAttribute("productList", productList);
        RequestDispatcher dispatcher = request.getRequestDispatcher("product-list.jsp");
        dispatcher.forward(request, response);
    }
}
```

- The ProductServlet handles requests sent to /product-list.
- It calls the ProductDAO to retrieve the list of products and sets it as a request attribute (productList).
- The request is then forwarded to the product-list.jsp view for rendering.

**Steps to Set Up MVC in a Java Web Application:**

1. **Set Up Your Project Structure**:
    - **src/main/java**: For your Java classes (Model, DAO, and Servlet).
    - **src/main/webapp/WEB-INF**: For JSP files and configuration (like web.xml).
2. **Configure Web Deployment Descriptor (web.xml)**: This can be used to map URLs to Servlets.

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="3.0">
    <servlet>
        <servlet-name>ProductServlet</servlet-name>
        <servlet-class>com.example.ProductServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ProductServlet</servlet-name>
        <url-pattern>/product-list</url-pattern>
    </servlet-mapping>
</web-app>
```

3. **Deploy the Application**:

    - Deploy your Java web application to a **Servlet container** like **Apache Tomcat**.
    - Ensure that the JDBC connection is correctly set up to interact with your database (e.g., MySQL).
4. **Access the Application**: After deployment, you can access the product list by navigating to the appropriate URL (e.g., http://localhost:8080/yourapp/product-list).

**Benefits of Using MVC in Java Web Applications:**

- **Separation of Concerns**: Each layer (Model, View, Controller) has a distinct responsibility, making the application easier to maintain and scale.
- **Modularity**: You can modify the View without changing the business logic (Model) or the flow control (Controller).
- **Testability**: Each component can be tested independently.

**Exercise 01: Triển khai MVC trong Java Web bằng Apache NetBeans**

**1. Tạo dự án mới**

1. Mở Apache NetBeans.
2. Chọn **File** > **New Project**.
3. Chọn **Java Web** > **Web Application** và nhấn **Next**.
4. Đặt tên cho dự án **No_Fullname_BookManagement**, Next
5. Chọn máy chủ, nhấn **Finish**.

**2. Tạo Model**

1. Trong thư mục Source Packages, tạo một package mới tên là **model**.
2. Trong package **model**, tạo một lớp mới tên là **Book.java**.

```
package model;

public class Book {
    private int id;
    private String title;
    private String author;
    private double price;

    public Book(int id, String title, String author, double price) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.price = price;
    }

    // Getters and Setters
}
```

mục `Web Pages`, tạo một tệp JSP mới tên là `bookList.jsp`.

jsp <%@ page contentType="text/html;charset=UTF-8" language="java" %>

**Book List**

| **ID** | **Title** | **Author** | **Price** |
|---|---|---|---|
| ${book.id} | ${book.title} | ${book.author} | ${book.price} |

**Bước 3: Tạo View**

3.1 Trong thư mục Web Pages, tạo một tệp JSP mới tên là bookList.jsp.

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
   <title>Book List</title>
</head>
<body>
<h2>Book List</h2>
<table border="1">
   <tr>
      <th>ID</th>
      <th>Title</th>
      <th>Author</th>
      <th>Price</th>
   </tr>
   <c:forEach var="book" items="${bookList}">
      <tr>
         <td>${book.id}</td>
         <td>${book.title}</td>
```

```
            <td>${book.author}</td>
            <td>${book.price}</td>
        </tr>
    </c:forEach>
</table>
</body>
</html>
```

Giải thích:

- Dòng @page thiết lập loại nội dung và mã hóa ký tự.
- Dòng @taglib khai báo thư viện JSTL (JavaServer Pages Standard Tag Library) để sử dụng thẻ c:forEach.
- Thẻ c:forEach lặp qua danh sách sách (bookList) và hiển thị thông tin từng cuốn sách trong bảng.

**4. Tạo Controller**

4.1. Trong thư mục `Source Packages`, tạo một package mới tên là `controller`.

4.2. Trong package `controller`, tạo một servlet mới tên là `BookController.java`.

```
package controller;
import model.Book;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
@WebServlet("/books")
```

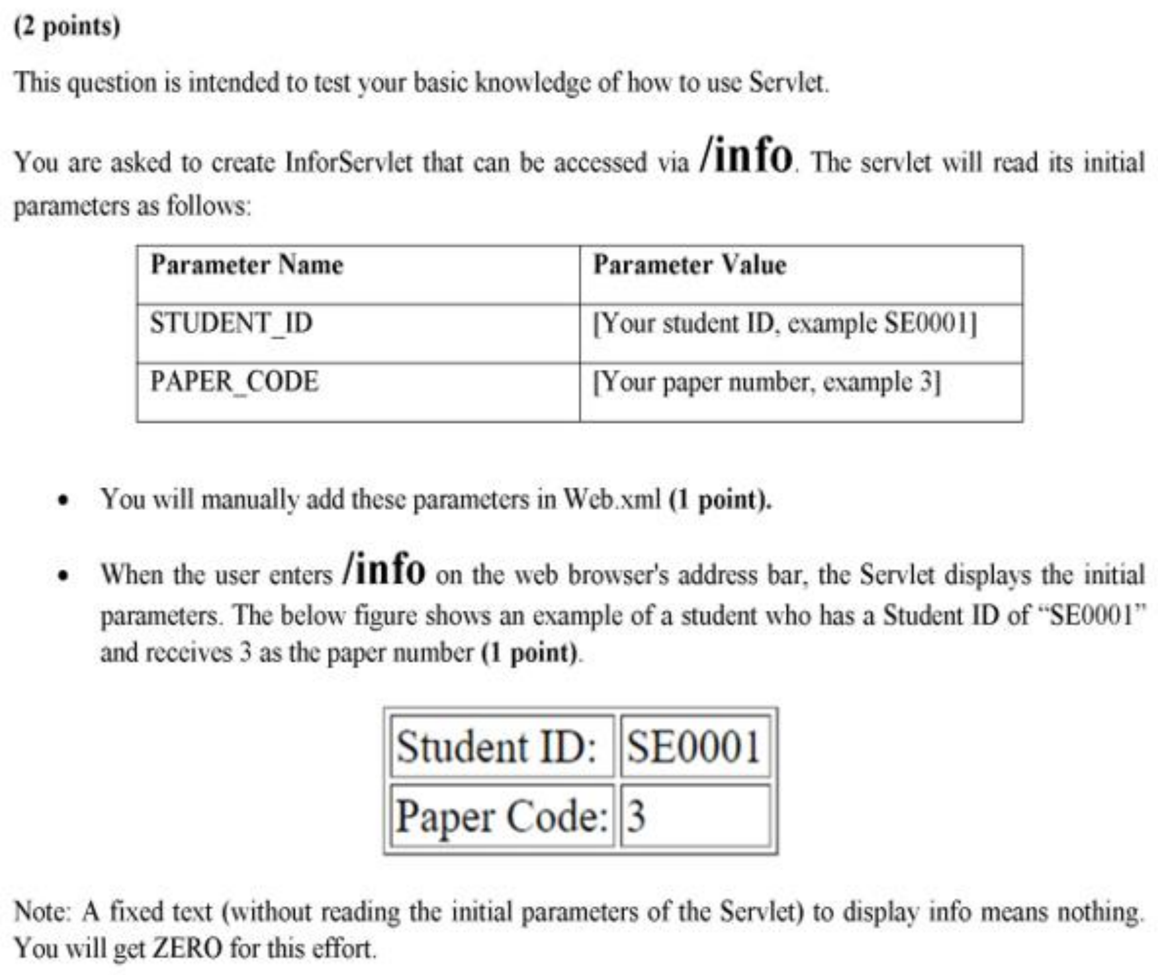
```
public class BookController extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        List<Book> bookList = new ArrayList<>();
        bookList.add(new Book(1, "Java Programming", "John Doe", 29.99));
        bookList.add(new Book(2, "Web Development", "Jane Smith", 39.99));
        bookList.add(new Book(3, "Database Systems", "Mike Johnson", 49.99));


        request.setAttribute("bookList", bookList);
        request.getRequestDispatcher("bookList.jsp").forward(request, response);
    }
}
```

**5. Cấu hình web.xml**

```
<web-app>
    <servlet>
        <servlet-name>BookController</servlet-name>
        <servlet-class>controller.BookController</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>BookController</servlet-name>
        <url-pattern>/books</url-pattern>
    </servlet-mapping>
</web-app>
```

Exercise slot 9_slot 10:

Exercise 02:



**(2 points)**

This question is intended to test your basic knowledge of how to use Servlet.

You are asked to create InforServlet that can be accessed via **/info**. The servlet will read its initial parameters as follows:

| **Parameter Name** | **Parameter Value** |
|---|---|
| STUDENT_ID | [Your student ID, example SE0001] |
| PAPER_CODE | [Your paper number, example 3] |

- You will manually add these parameters in Web.xml **(1 point).**
- When the user enters **/info** on the web browser's address bar, the Servlet displays the initial parameters. The below figure shows an example of a student who has a Student ID of "SE0001" and receives 3 as the paper number **(1 point)**.

| | |
|---|---|
| Student ID: | SE0001 |
| Paper Code: | 3 |

Note: A fixed text (without reading the initial parameters of the Servlet) to display info means nothing. You will get ZERO for this effort.

Exercise 03:

Tạo form cho phép nhập vào 1 số và trả ra kết quả thông báo số đó là số chẵn hay lẻ, có là số hoàn hảo hay không, có là số nguyên tố hay không?

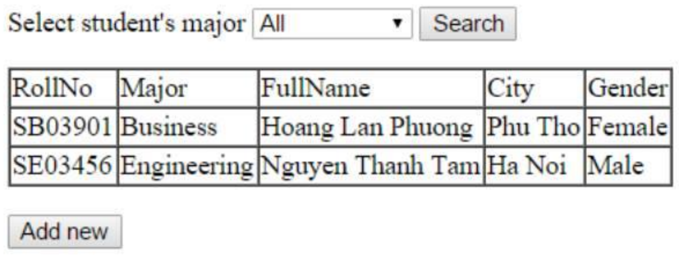
Exercise 04:

**Preparation:**

1. Create your MS SQL database named **TestFinal by running code in script SQLFile.sql.**

**Questions:**

Create an **index.jsp** page with GUI as below figures:

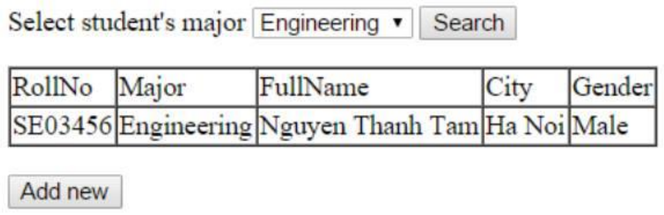


| RollNo | Major | FullName | City | Gender |
|---|---|---|---|---|
| SB03901 | Business | Hoang Lan Phuong | Phu Tho | Female |
| SE03456 | Engineering | Nguyen Thanh Tam | Ha Noi | Male |

**Requirements:**

- Values for select control are **All, Engineering, Business and Others**. By default the selected value of select control is **All (0.1 Point)**
- At the running time, display the information of all students as table (**2.0 Point**), order by Major (**0.2 Point**)

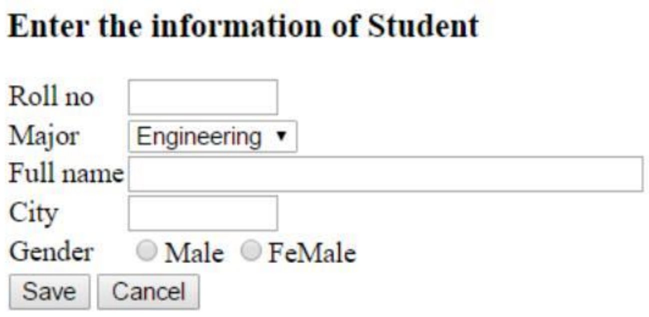
When the users select an item from select control and click Search, display the information of all Students who have major is equal to selected major from select control **(2.5 point).** Data must be displayed on the page **index.jsp**, and the selected value on select control must be set to the one that has been selected **(1.0 Point)**.



| RollNo | Major | FullName | City | Gender |
|---|---|---|---|---|
| SE03456 | Engineering | Nguyen Thanh Tam | Ha Noi | Male |

*The table must have the same columns with one in above figures and must on page index.jsp*

When the users click on **Add new** button, browse to page **add.jsp** which will display the input form as the below figures **(1.0 Point).**


Enter the information of Student

| | |
|---|---|
| Roll no | |
| Major | Engineering ▾ |
| Full name | |
| City | |
| Gender | ○ Male ○ FeMale |

**Requirements:**

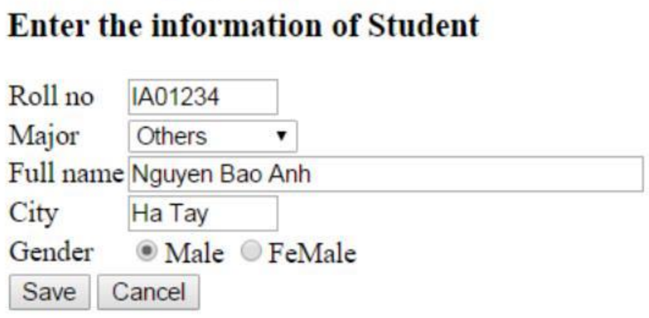
- Values for select control are **Engineering, Business and Others**. By default the selected value of select control is Engineering (**0.1 Point**).

When the users click on **Save** button, validate the data on the input form as rules:

- Require data for all text fields (**0.3 Point - 0.1 Point for each error message**).
- Roll no must start with SE if major is Engineering and must start with SB if the major is Bussiness (**0.2 Point**).
- Gender must be selected (**0.1 Point**).

All the validation must be DONE on **add.jsp** page.

When the validation is correct. Save information on input form to Student table (**2.0 Point**) and browse back to page **list.jsp**.


Enter the information of Student

| | |
|---|---|
| Roll no | IA01234 |
| Major | Others ▾ |
| Full name | Nguyen Bao Anh |
| City | Ha Tay |
| Gender | ● Male ○ FeMale |

When user clicks on the Cancel button, browse back to page **list.jsp (0.5 Point)**