## DAY-9\_SPRING MVC

## Case Study Title: Online Course Enrollment System

**Scenario:** An educational startup wants to build a basic web application for students to view available courses and enroll online. The company has a small IT team familiar with Java and wants to use Spring MVC to ensure the application follows a clean, maintainable structure based on MVC architecture.

## Objectives:

1. Display a list of available courses.
2. Allow students to register by filling out an enrollment form.
3. Confirm enrollment and store student details.
   * Beans.xml or Java Config: Defines Spring beans, view resolvers, and component scanning setup

## Example Use Cases:

1. **CourseController**
   * /courses → Displays list of courses
   * /enroll → Shows enrollment form
   * /submitEnrollment → Processes submitted data

## Views (JSP)

* + courses.jsp → Displays all courses
  + enroll.jsp → Input form for registration
  + success.jsp → Confirmation message

**SOLUTION :**

### **#Dependencies**

#### <dependencies>

<**dependency**>

<**groupId**>org.springframework</**groupId**>

<**artifactId**>spring-webmvc</**artifactId**>

<**version**>5.3.29</**version**>

</**dependency**>

#### <dependency>

<**groupId**>javax.servlet</**groupId**>

<**artifactId**>javax.servlet-api</**artifactId**>

<**version**>4.0.1</**version**>

<**scope**>provided</**scope**>

</**dependency**>

</**dependencies**>

### #**Create Model Classes**

**public class** Course { **private** String id; **private** String name;

**private** String description;

*// Getters and Setters*

}

**public class** Student { **private** String name; **private** String email; **private** String courseId;

*// Getters and Setters*

}

**#Create Controller**

@Controller

**public class** CourseController {

List<Course> courses = Arrays.asList(

**new** Course("101", "Java Basics", "Intro to Java"),

**new** Course("102", "Spring MVC", "Build MVC apps")

);

@RequestMapping("/courses")

**public** String showCourses(Model model) { model.addAttribute("courses", courses); **return** "courses";

}

@RequestMapping("/enroll")

**public** String enrollForm(@RequestParam("courseId") String courseId, Model model) { model.addAttribute("courseId", courseId);

**return** "enroll";

}

@RequestMapping(value = "/submitEnrollment", method = RequestMethod.POST)

**public** String submitEnrollment(@ModelAttribute Student student, Model model) { model.addAttribute("student", student);

**return** "success";

}

}

### **#Create Views (JSP files)**

``

<%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c" %>

<html><body>

<h2>Available Courses</h2>

<ul>

**<c:forEach** var="course" items="${courses}"**>**

<li>${course.name} - <a href="enroll?courseId=${course.id}">Enroll</a></li>

#### </c:forEach>

</ul>

</body></html>

``

<html><body>

<h2>Enroll in Course: ${courseId}</h2>

<form action="submitEnrollment" method="post"> Name: <input type="text" name="name" required/><br/>

Email: <input type="email" name="email" required/><br/>

<input type="hidden" name="courseId" value="${courseId}" />

<input type="submit" value="Enroll" />

</form>

</body></html>

``

<html><body>

<h2>Enrollment Successful!</h2>

<p>Name: ${student.name}</p>

<p>Email: ${student.email}</p>

<p>Enrolled Course ID: ${student.courseId}</p>

</body></html>

### **#Configure web.xml and Spring Config**

``

<**web-app**>

<**servlet**>

<**servlet-name**>dispatcher</**servlet-name**>

<**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>

#### <load-on-startup>1</load-on-startup>

</**servlet**>

#### <servlet-mapping>

<**servlet-name**>dispatcher</**servlet-name**>

<**url-pattern**>/</**url-pattern**>

#### </servlet-mapping>

</**web-app**>

``

<**context:component-scan** base-package="com.example.controller" />

<**bean** class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<**property** name="prefix" value="/WEB-INF/views/" />

<**property** name="suffix" value=".jsp" />

</**bean**>

## Case Study Title: Online Shopping Portal – Order Processing Monitoring

**Scenario Description** An online shopping portal provides a service class OrderService that has three key methods:

1. addToCart(String product)
2. placeOrder(String orderId)
3. cancelOrder(String orderId)

As a developer, you want to add cross-cutting concerns like:

* + Logging when methods start (@Before)
  + Logging after successful method execution (@AfterReturning)
  + Logging errors when a method fails (@AfterThrowing)
  + Performing cleanup or logging after any method execution, success or failure (@After)

SOLUTION :

### # **Add dependency in pom.xml**

#### <dependency>

<**groupId**>org.springframework</**groupId**>

<**artifactId**>spring-aspects</**artifactId**>

<**version**>5.3.29</**version**>

</**dependency**>

### #**Create Service Class**

@Component

**public class** OrderService {

**public** void addToCart(String product) { System.out.println("Product added to cart: " + product);

}

**public** void placeOrder(String orderId) {

**if** (orderId.equals("INVALID\_ID")) {

**throw new** RuntimeException("OrderNotFoundException");

}

System.out.println("Order placed: " + orderId);

}

**public** void cancelOrder(String orderId) { System.out.println("Order cancelled: " + orderId);

}

}

### **#Create Aspect Class**

@Aspect @Component

**public class** OrderLoggingAspect

@Before("execution(\* com.example.service.OrderService.\*(..))")

**public** void logBefore(JoinPoint joinPoint) { System.out.println("Starting method: " + joinPoint.getSignature());

}

@AfterReturning("execution(\* com.example.service.OrderService.\*(..))")

**public** void logAfterSuccess(JoinPoint joinPoint) {

System.out.println("Method executed successfully: " + joinPoint.getSignature());

}

@AfterThrowing(pointcut = "execution(\* com.example.service.OrderService.\*(..))", throwing = "ex")

**public** void logException(JoinPoint joinPoint, Throwable ex) { System.out.println("Exception in method: " + joinPoint.getSignature() + ", Message: " +

ex.getMessage());

}

@After("execution(\* com.example.service.OrderService.\*(..))")

**public** void logAfter(JoinPoint joinPoint) {

System.out.println("Method execution finished: " + joinPoint.getSignature());

}

}

***# Java Config Class***

@Configuration @EnableAspectJAutoProxy @ComponentScan("com.example")

**public class** AppConfig {

}

### #**Main Application**

**public class** MainApp {

**public** static void main(String[] args) { AnnotationConfigApplicationContext context = **new**

AnnotationConfigApplicationContext(AppConfig.class);

OrderService orderService = context.getBean(OrderService.class);

orderService.addToCart("Laptop");

#### try {

orderService.placeOrder("ORD123"); orderService.placeOrder("INVALID\_ID");

} **catch** (Exception e) {

*// Exception will be logged by aspect*

}

orderService.cancelOrder("ORD123"); context.close();

}

}