EXPERIMENT NO. 04

Title: Application to handle form data in spring MVC.

Outcome: Students will be able to develop application to handle form data in spring MVC

Theory:

SOFTWARE & HARDWARE REQUIREMENTS:

- Eclipse IDE (with Spring Tools)
- JDK 17 or later
- Spring Boot or Spring MVC libraries
- Maven Build Tool
- Embedded Tomcat Server
- Web Browser (for testing)

THEORY:

Spring MVC (Model–View–Controller) is a web framework that separates the application logic into three layers:

- **Model** – Represents application data or form backing objects.
- **View** – Defines the user interface (HTML, JSP, or Thymeleaf).
- **Controller** – Handles incoming requests, processes data, and returns the response.

When a user submits a form:

1. The **View** (HTML/JSP) sends the data to a controller via HTTP POST.
2. The **Controller** binds this form data to a **Model** object using @ModelAttribute or reads values using @RequestParam.
3. The controller returns a logical view name, and the **View Resolver** displays the processed result.

PROCEDURE / STEPS:

1. Create a Spring Boot Project

- In Eclipse:
  File → New → Spring Starter Project
  Add dependencies:
  - **Spring Web**
  - **Thymeleaf** (or JSP if using non-Boot Spring MVC)

---

2. Project Structure

src/main/java/com/example/formapp
├─ FormAppApplication.java

```
  ├─ controller/
  │    └─ StudentController.java
  ├─ model/
  │    └─ Student.java
src/main/resources/templates/
  ├─ form.html
  └─ success.html
```

---

3. Model Class

Student.java

```
package com.example.formapp.model;

public class Student {
    private String name;
    private String email;
    private String department;

    // Getters and setters
    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 getDepartment() { return department; }
    public void setDepartment(String department) { this.department = department; }
}
```

---

4. Controller Class

StudentController.java

```
package com.example.formapp.controller;

import com.example.formapp.model.Student;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class StudentController {

    @GetMapping("/form")
```

```
    public String showForm(Model model) {
        model.addAttribute("student", new Student());
        return "form";  // form.html
    }

    @PostMapping("/submit")
    public String processForm(@ModelAttribute("student") Student student, Model
model) {
        model.addAttribute("name", student.getName());
        model.addAttribute("email", student.getEmail());
        model.addAttribute("department", student.getDepartment());
        return "success"; // success.html
    }
}
```

---

5. View Templates

form.html (in src/main/resources/templates/)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Student Form</title>
</head>
<body>
<h2>Enter Student Details</h2>
<form th:action="@{/submit}" th:object="${student}" method="post">
    <label>Name:</label>
    <input type="text" th:field="*{name}" /><br><br>

    <label>Email:</label>
    <input type="email" th:field="*{email}" /><br><br>

    <label>Department:</label>
    <input type="text" th:field="*{department}" /><br><br>

    <button type="submit">Submit</button>
</form>
</body>
</html>
```

---

success.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
   <meta charset="UTF-8">
   <title>Submission Successful</title>
</head>
<body>
<h2>Student Information Submitted Successfully!</h2>
<p><b>Name:</b> <span th:text="${name}"></span></p>
<p><b>Email:</b> <span th:text="${email}"></span></p>
<p><b>Department:</b> <span th:text="${department}"></span></p>

<a href="/form">Go Back</a>
</body>
</html>
```

---

6. Run the Application

- Right-click → **Run As → Spring Boot App**
- Open browser → http://localhost:8080/form
- Fill in details and submit the form.
- The response page displays the submitted data.

OUTPUT:

- A user input form is displayed in the browser.
- On submission, the application displays the entered details on a new page.
- Data binding between the form and the model class works successfully.

RESULT:

A Spring MVC application was successfully developed to handle user form data using @ModelAttribute in a controller and display the response using Thymeleaf views.

Questions:

1. What is the role of the @ModelAttribute annotation in Spring MVC?
2. Differentiate between @RequestParam and @ModelAttribute.

Conclusion : Thus, I have developed application to handle form data in spring MVC