PW 03 : Understanding the Importance of the Model-View-Controller (MVC) in Spring Boot

Objective:

We will create a Spring Boot application using the MVC architecture. The application will demonstrate how data is passed from the **Controller** to the **View** using a **Model**.

Step-by-Step Guide

Step 1: Set Up Your Spring Boot Project

- Follow the steps from the previous practical work to create a new Spring Boot project using **Spring Initializr**.
- Rename your project as : mySecondProject
- Add the following dependencies:
  - **Spring Web**: For building the web application.
  - **Thymeleaf**: As the template engine for the view.
- Open your folder from VsCode

Step 2: Create a Model Class

1. In the **src/main/java/MySecondProject** directory, create a new package called **model**.
2. Inside the models package, create a class called **student.java**.
3. Create private **student: Name, e-mail**, **Age**.
4. Generate a **Constructor for all fields and also generate Getters and Setters**.

Step 3: Create the Controller

Students' information (name, email, and age) will be hardcoded in the controller and passed to the view.

1. In the **src/main/java/MySecondProject** directory, create a new file Controllers.
2. Create class called StudentController.java:
3. Use @Controller. To define the class as an mvc controller

4. Use @GetMapping("/students") to maps the / URL
5. Add a function inside the class, **getStudents(Model model):** This is used to pass data from the controller to the view.
6. Create a liste of some students.
7. **model.addAttribute("students", students);** This adds the student object to the model so it can be accessed in the view.
8. The method returns the name of the view, which corresponds to the HTML file we will create in the next step.

Step 4: Create the View (Thymeleaf Template)

1. In the **src/main/resources/templates** directory, create a new file called **students.html:**
2. In the html tag, put: **xmlns:th=http://www.thymeleaf.org**
3. The title will be : **List of Students**
4. Create a new table, containing information of a student which are (student name, student email, student age)
5. th:text="${student.name}": This is a Thymeleaf expression that dynamically displays the name attribute of the student object..

Step 5. Run the application

Step 6: Test the Application

- Open your browser and go to http://localhost:8080/students.
- You should see the students’ information displayed on the page.