Lab Series: Xây Dựng Web App Căn Bản
Lab 3: Xây Dựng Frontend (Server-Side Rendering)

Ở bài thực hành Lab 2 (REST API), sinh viên đã xây dựng API trả về dữ liệu dưới định dạng JSON. Tuy nhiên, để người dùng cuối có thể tương tác trực quan, hệ thống cần cung cấp một giao diện đồ họa (User Interface - UI).

Bài thực hành Lab 3 hướng dẫn kỹ thuật **Server-Side Rendering (SSR)** sử dụng **Thymeleaf**. Đây là phương pháp tạo các trang web động trực tiếp từ Spring Boot, cho phép tích hợp dữ liệu từ Backend vào giao diện HTML trước khi trả về cho trình duyệt.

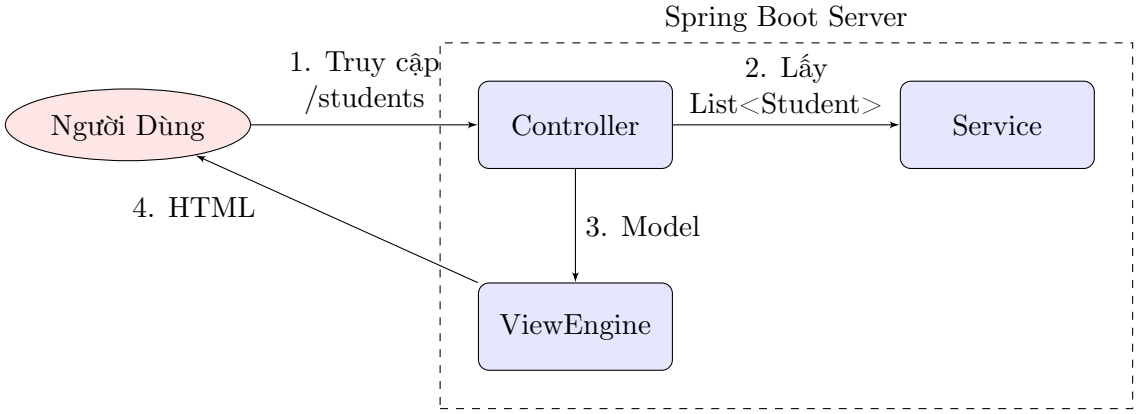
1 Mục Tiêu

- Hiểu mô hình kiến trúc **MVC** (Model-View-Controller) trong phát triển web truyền thống.
- Cấu hình và sử dụng thư viện **Thymeleaf Template Engine**.
- Vận dụng các cú pháp Thymeleaf (`th:each`, `th:text`) để hiển thị dữ liệu động từ Backend.

2 So Sánh Kiến Trúc (MVC vs REST API)

Sự khác biệt cơ bản giữa hai mô hình:

- **REST API (Lab 2)**: Controller trả về **Dữ liệu thô (JSON)**. Trình duyệt (hoặc Frontend App) chịu trách nhiệm xử lý hiển thị giao diện.
- **SSR Web (Lab 3)**: Controller trả về **Giao diện HTML** đã được điền sẵn dữ liệu (rendered).



3 Cấu Hình Thymeleaf

Để tích hợp Thymeleaf, thêm dependency sau vào tập tin cấu hình `pom.xml` (trong thẻ `<dependencies>`):

```
<!-- Thymeleaf: Template Engine tao giao dien HTML -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

Sau khi cập nhật, thực hiện lệnh Reload Maven để tải thư viện:

```
./mvnw dependency:resolve
```

4 Xây Dựng Web Controller

Tạo lớp `StudentWebController` trong package `controller` để xử lý và trả về giao diện HTML.

```java
package vn.edu.hcmut.cse.adse.lab.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller; // Luu y: su dung @Controller, KHONG dung
    ↪ @RestController
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import vn.edu.hcmut.cse.adse.lab.service.StudentService;
import vn.edu.hcmut.cse.adse.lab.entity.Student;

import java.util.List;

@Controller
@RequestMapping("/students")
public class StudentWebController {

    @Autowired
    private StudentService service;

    // Route: GET http://localhost:8080/students
    @GetMapping
    public String getAllStudents(Model model) {
        // 1. Lay du lieu tu Service
        List<Student> students = service.getAll();

        // 2. Dong goi du lieu vao "Model" de chuyen sang View
        // Key "dsSinhVien" se duoc su dung ben file HTML
        model.addAttribute("dsSinhVien", students);

        // 3. Tra ve ten cua View (khong can duoi .html)
        // Spring Boot se tu dong tim file tai: src/main/resources/templates/students.html
        return "students";
    }
}
```

Giải thích:

- `@Controller`: Đánh dấu lớp này trả về View (HTML), khác với `@RestController` trả về Data (JSON).
- `Model`: Đối tượng trung gian dùng để truyền tải dữ liệu từ Java code sang View Template (HTML).

5 Xây Dựng View Template

Tạo tập tin `students.html` trong thư mục `src/main/resources/templates/` (nếu chưa có thư mục templates, hãy tạo mới).

Lưu ý: Thư mục `static` dành cho tài nguyên tĩnh (CSS, JS, Images), trong khi `templates` dành cho các file cần xử lý động bởi Template Engine.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"> <!-- Khai bao namespace de dung th:* -->
<head>
    <meta charset="UTF-8">
    <title>Danh Sach Sinh Vien</title>
    <!-- CSS don gian -->
    <style>
        body { font-family: sans-serif; padding: 20px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 8px; text-align: left; }
        th { background-color: #f2f2f2; }
    </style>
</head>
<body>
    <h1>Danh Sach Sinh Vien (SSR)</h1>

    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Ho va Ten</th>
                <th>Email</th>
                <th>Tuoi</th>
            </tr>
        </thead>
        <tbody>
            <!-- Cu phap Thymeleaf: Duyet qua danh sach "dsSinhVien" -->
            <!-- th:each="bien_tam : ${key_trong_model}" -->
            <tr th:each="student : ${dsSinhVien}">
                <td th:text="${student.id}">ID Mau</td>
                <td th:text="${student.name}">Ten Mau</td>
                <td th:text="${student.email}">Email Mau</td>
                <td th:text="${student.age}">0</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Giải thích cú pháp Thymeleaf:

- `th:each="student : ${dsSinhVien}"`: Cấu trúc lặp (tương tự for-each). Với mỗi phần tử trong danh sách `dsSinhVien`, biến `student` sẽ được gán giá trị tương ứng để hiển thị trong thẻ `<tr>`.
- `th:text="${student.name}"`:
  Thay thế nội dung text của thẻ `<td>` bằng giá trị trả về từ phương thức `student.getName()`. Giá trị mẫu "Tên Mẫu" sẽ bị ghi đè khi ứng dụng thực thi.

6 Chạy & Kiểm Tra

1. Khởi động lại ứng dụng: `./mvnw spring-boot:run`
2. Mở trình duyệt truy cập: `http://localhost:8080/students`

Kết quả mong đợi: Bạn sẽ thấy bảng danh sách sinh viên được hiển thị. Nếu View Source (Ctrl+U) trang web, bạn sẽ thấy mã HTML đầy đủ (có dữ liệu) được trả về từ server, không phải là bảng trống như cách làm Fetch API.

7 Bài Tập Nâng Cao

7.1 Chức năng Tìm Kiếm

Hãy thêm một Form tìm kiếm đơn giản phía trên bảng.

Gợi ý HTML:

```
<form action="/students" method="GET">
    <input type="text" name="keyword" placeholder="Nhap ten..." />
    <button type="submit">Tim</button>
</form>
```

Gợi ý Controller: Sửa lại hàm `getAllStudents` để nhận tham số keyword:

```
@GetMapping
public String getAllStudents(@RequestParam(required = false) String keyword, Model model) {
    List<Student> students;
    if (keyword != null && !keyword.isEmpty()) {
        // Can viet them ham searchByName trong Service/Repository
        students = service.searchByName(keyword);
    } else {
        students = service.getAll();
    }
    model.addAttribute("dsSinhVien", students);
    return "students";
}
```

7.2 Hiển Thị Có Điều Kiện

Sử dụng `th:if` hoặc `th:classappend` để tô màu đỏ các sinh viên chưa đủ 18 tuổi.

```
<!-- Vi du goi y -->
<tr th:each="student : ${dsSinhVien}" th:class="${student.age < 18} ? 'text-danger' : ''">
    ...
</tr>
```