Lab 3: Xây Dựng Frontend (Server-Side Rendering)

Ở Lab 2 (REST API), chúng ta đã xây dựng API trả về JSON. Tuy nhiên, để người dùng cuối có thể tương tác dễ dàng, chúng ta cần một giao diện đồ họa (UI). Trong Lab 3 này, thay vì xây dựng một ứng dụng Frontend tách biệt gọi API (như React/Vue), chúng ta sẽ sử dụng kỹ thuật **Server-Side Rendering (SSR)** với **Thymeleaf**. Đây là cách tiếp cận truyền thống nhưng vẫn rất mạnh mẽ, giúp tạo ra các trang web động trực tiếp từ Spring Boot.

Mục Tiêu

- Hiểu mô hình **MVC** (Model-View-Controller) trong web truyền thống.
- Cấu hình và sử dụng **Thymeleaf Template Engine**.
- Sử dụng cú pháp Thymeleaf (`th:each`, `th:text`) để hiển thị dữ liệu từ Backend.

1. Kiến Trúc (MVC vs REST API)

Hãy so sánh sự khác biệt:

- **REST API (Lab 2)**: Controller trả về **Dữ liệu thô (JSON)**. Trình duyệt (hoặc Frontend App) tự lo việc vẽ giao diện.
- **SSR Web (Lab 3)**: Controller trả về **Giao diện HTML** (đã được điền sẵn dữ liệu).

```
Parse error on line 3:
...oller    subgraph "Spring Boot Server"
----------------------^
Expecting 'SEMI', 'NEWLINE', 'SPACE', 'EOF', 'GRAPH',
'DIR', 'TAGEND', 'TAGSTART', 'UP', 'DOWN', 'subgraph',
'end', 'SQE', 'PE', '-)', 'DIAMOND_STOP', 'MINUS', '--',
'ARROW_POINT', 'ARROW_CIRCLE', 'ARROW_CROSS', 'ARROW_OPEN',
'DOTTED_ARROW_POINT', 'DOTTED_ARROW_CIRCLE',
'DOTTED_ARROW_CROSS', 'DOTTED_ARROW_OPEN', '==',
'THICK_ARROW_POINT', 'THICK_ARROW_CIRCLE',
'THICK_ARROW_CROSS', 'THICK_ARROW_OPEN', 'PIPE', 'STYLE',
'LINKSTYLE', 'CLASSDEF', 'CLASS', 'CLICK', 'DEFAULT',
'NUM', 'PCT', 'COMMA', 'ALPHA', 'COLON', 'BRKT', 'DOT',
'PUNCTUATION', 'UNICODE_TEXT', 'PLUS', 'EQUALS', 'MULT',
got 'STR'
```

2. Cấu Hình Thymeleaf

Để sử dụng Thymeleaf, chúng ta cần thêm thư viện vào dự án.

Mở file `pom.xml`, thêm dependency sau vào trong thẻ `<dependencies>`:

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

Sau khi thêm, hãy chạy lệnh Reload Maven (hoặc click icon Reload trong IDE) để tải thư viện về.

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

3. Tạo Web Controller

Chúng ta tạo một Controller mới chuyên trách việc trả về giao diện HTML.

Tạo class `StudentWebController` trong package `controller`:

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller; // Lưu ý: dùng
@Controller, KHÔNG dùng @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.adsoftweng.lab.service.StudentService;
import vn.edu.hcmut.cse.adsoftweng.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. Lấy dữ liệu từ Service
        List<Student> students = service.getAll();

        // 2. Đóng gói dữ liệu vào "Model" để chuyển sang View
        // Key "dsSinhVien" sẽ được dùng bên file HTML
        model.addAttribute("dsSinhVien", students);

        // 3. Trả về tên của View (không cần đuôi .html)
        // Spring Boot sẽ tự tìm file tại:
src/main/resources/templates/students.html
        return "students";
    }
}
```

Điểm quan trọng:

- `@Controller`: Báo hiệu class này trả về View (HTML), khác với `@RestController` trả về Data (JSON).
- `Model`: Là cái túi chứa dữ liệu để mang từ Java code sang file HTML.

4. Tạo View (Thymeleaf Template)

Tạo file `students.html` trong thư mục `src/main/resources/templates/` (nếu chưa có thư mục templates thì tạo mới nó).

Lưu ý: Không tạo trong `static` nhé. `static` là cho file tĩnh nguyên bản, còn `templates` là cho file cần xử lý động.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"> <!-- Khai báo namespace để dùng th:* -->
<head>
    <meta charset="UTF-8">
    <title>Danh Sách Sinh Viên</title>
    <!-- CSS đơn giản -->
    <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 Sách Sinh Viên (SSR)</h1>

    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Họ và Tên</th>
                <th>Email</th>
                <th>Tuổi</th>
            </tr>
        </thead>
        <tbody>
            <!-- Cú pháp Thymeleaf: Duyệt qua danh sách "dsSinhVien" -->
            <!-- th:each="biến_tạm : ${key_trong_model}" -->
            <tr th:each="student : ${dsSinhVien}">
                <td th:text="${student.id}">ID Mẫu</td>
                <td th:text="${student.name}">Tên Mẫu</td>
                <td th:text="${student.email}">Email Mẫu</td>
                <td th:text="${student.age}">0</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```

Giải thích cú pháp Thymeleaf:

- `th:each="student : ${dsSinhVien}"`: Vòng lặp for-each. Với mỗi phần tử trong list `dsSinhVien`, gán vào biến `student` và lặp lại thẻ `<tr>`.
- `th:text="${student.name}"`: Gán nội dung text của thẻ `<td>` bằng giá trị `student.getName()`. Giá trị cũ "Tên Mẫu" sẽ bị ghi đè khi chạy thật.

5. 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.

6. Bài Tập Nâng Cao

6.1. 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="Nhập tên..." />
    <button type="submit">Tìm</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()) {
        // Cần viết thêm hàm searchByName trong Service/Repository
        students = service.searchByName(keyword);
    } else {
        students = service.getAll();
    }
    model.addAttribute("dsSinhVien", students);
    return "students";
}
```

6.2. Hiển Thị Có Điều Kiện

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

```
<!-- Ví dụ gợi ý -->
<tr th:each="student : ${dsSinhVien}" th:class="${student.age < 18} ? 'text-
danger' : ''">
    ...
</tr>
```