Lab Series: Xây Dựng Web App Căn Bản
Lab 1: Khởi Tạo & Kiến Trúc Hệ Thống

1 Mục Tiêu

- Hiểu bài toán và kiến trúc hệ thống (MVC / Layered Architecture).
- Khởi tạo project Spring Boot.
- Cấu hình kết nối cơ sở dữ liệu (SQLite).

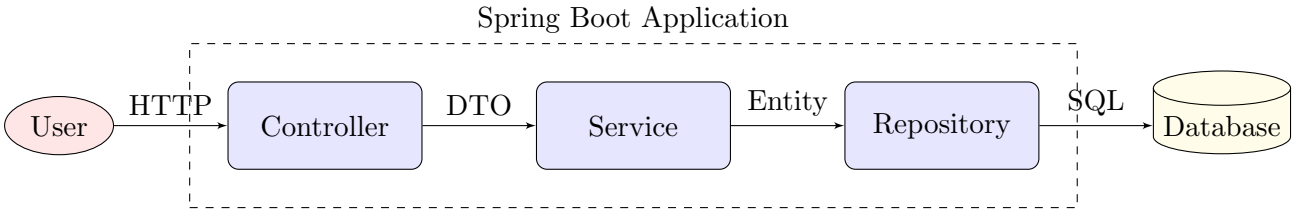
2 Giới Thiệu Chuyên Đề

Bài thực hành hướng dẫn xây dựng **Student Management System** - một ứng dụng quản lý sinh viên cơ bản. Sinh viên sẽ đóng vai trò Fullstack Developer để phát triển toàn diện từ Backend (API), Cơ sở dữ liệu (Database) đến Frontend.

2.1 Kiến Trúc Hệ Thống (Layered Architecture)

Mô hình kiến trúc phân lớp (Layered Architecture) được áp dụng để tổ chức mã nguồn. Đây là mô hình tiêu chuẩn trong phát triển ứng dụng Spring Boot, cụ thể hóa tư duy MVC cho Backend API.

Mô hình luồng dữ liệu:



Chi tiết các tầng:

- **Controller Layer (Presentation)**:
  - Đóng vai trò cổng giao tiếp (Gateway) của ứng dụng.
  - Tiếp nhận các HTTP Request (GET, POST, PUT, DELETE).
  - Thực hiện kiểm tra sơ bộ (Validate) dữ liệu đầu vào.
  - Chuyển yêu cầu xuống Service Layer để xử lý và trả về phản hồi (Response) dưới dạng JSON.
- **Service Layer (Business Logic)**:
  - Chứa toàn bộ logic nghiệp vụ cốt lõi (Core Business).
  - Ví dụ: Kiểm tra ràng buộc dữ liệu (tuổi > 0, email duy nhất).
  - Đóng vai trò cầu nối giữa Controller và Repository.
- **Repository Layer (Data Access)**:
  - Trực tiếp tương tác với Cơ sở dữ liệu.

  - Thực thi các truy vấn dữ liệu (SELECT, INSERT, UPDATE, DELETE).
  - Trong Spring Data JPA, tầng này được hiện thực thông qua các Interface.

- **Database**:
  - Nơi lưu trữ dữ liệu vật lý (File .db của SQLite hoặc PostgreSQL Server).

2.2 Giới Thiệu Công Nghệ

2.2.1 Java

Là ngôn ngữ lập trình hướng đối tượng, mạnh mẽ và độc lập nền tảng (Write Once, Run Anywhere). Trong bài lab này, chúng ta sử dụng Java để viết logic xử lý cho Backend.

2.2.2 Spring Boot

Là một framework được xây dựng trên nền tảng Spring, giúp đơn giản hóa việc phát triển ứng dụng Java.

- **Stand-alone**: Có thể chạy độc lập với Embedded Server (Tomcat) tích hợp sẵn.
- **Opinionated**: Cung cấp các cấu hình mặc định hợp lý (Convention over Configuration).
- **Production-ready**: Tích hợp sẵn các công cụ monitoring, health check.

2.2.3 Maven

Là công cụ quản lý dự án (Project Management) và tự động hóa quy trình build (Build Automation).

- **Dependency Management**: Quản lý các thư viện (như Spring Web, SQLite) thông qua file `pom.xml`. Tự động tải về thay vì phải copy thủ công.
- **Standard Directory Layout**: Định nghĩa cấu trúc thư mục chuẩn cho project Java.

2.2.4 Database Management System (DBMS)

SQLite: Là hệ quản trị cơ sở dữ liệu quan hệ nhỏ gọn, không cần cài đặt server riêng (serverless), dữ liệu được lưu trực tiếp vào một file duy nhất. Rất phù hợp cho quá trình phát triển (development) và các ứng dụng nhỏ. Trong bài lab này, chúng ta sử dụng SQLite để làm quen với cấu hình Database trong Spring Boot.

Các DBMS khác (PostgreSQL, MySQL, ...): Trong môi trường thực tế (production), các hệ thống lớn thường sử dụng PostgreSQL hoặc MySQL vì khả năng chịu tải cao, hỗ trợ đồng thời nhiều kết nối và các tính năng nâng cao khác. Spring Boot hỗ trợ chuyển đổi giữa các DBMS này rất dễ dàng, chỉ cần thay đổi cấu hình trong `application.properties` và dependency tương ứng.

3 Khởi Tạo Dự Án (Project Initialization)

Sử dụng công cụ **Spring Initializr** để khởi tạo cấu trúc dự án chuẩn.

3.1 Truy cập trang chủ Spring Initializr

Truy cập `https://start.spring.io/`

3.2 Cấu hình thông số dự án

Cấu hình các thông số sau:

- **Project**: Chọn `Maven`.
- **Language**: Chọn `Java`.
- **Spring Boot**: Chọn `4.0.2`.

3.3 Điền thông tin Project Metadata

- **Group**: `vn.edu.hcmut.cse.adse`.
- **Artifact**: `student-management`.
- **Name**: `StudentManagement`.
- **Package name**: `vn.edu.hcmut.cse.adse.lab`.
- **Packaging**: `Jar`.
- **Java**: Chọn `17` (hoặc 21 nếu máy bạn đã cài bản mới).

3.4 Thêm Dependencies

Nhấn nút **ADD DEPENDENCIES** và tìm chọn:

1. **Spring Web**: Để xây dựng RESTful API và MVC.
2. **Spring Data JPA**: Để làm việc với Database (ORM).

3.5 Generate & Open

- Nhấn **GENERATE** để tải file `.zip` về máy.
- Giải nén file và mở thư mục bằng IDE của bạn.

3.6 Cấu Trúc Thư Mục Mặc Định

Sau khi giải nén, bạn sẽ thấy cấu trúc project Maven chuẩn như sau:

- `pom.xml`: File quan trọng nhất, nơi khai báo các thư viện (dependencies) mà chúng ta đã chọn ở Bước 4 (Spring Web, Spring Data JPA,...).
- `mvnw` & `mvnw.cmd`: Maven Wrapper script. Giúp bạn chạy các lệnh Maven (như build, run) mà KHÔNG cần cài đặt Maven vào máy tính. Đây là công cụ được khuyên dùng.
- `src/main/java`: Nơi chứa mã nguồn Java của ứng dụng.
  - Trong đó sẽ có sẵn package `vn.edu.hcmut.cse.adse.lab` và file main `StudentManagementApplication.java`.
- `src/main/resources`: Nơi chứa các file cấu hình và tài nguyên tĩnh.
  - `application.properties`: File cấu hình chính của Spring Boot (kết nối DB, cổng server,...).
  - `static/` & `templates/`: Folders chứa file CSS, JS, HTML (cho Frontend).
- `src/test`: Nơi viết các Unit Test.

4 Cấu Trúc Package

Tổ chức mã nguồn theo kiến trúc Layered Architecture (đã giới thiệu ở Phần 2). Tạo các package rỗng bên trong `vn.edu.hcmut.cse.adse.lab` theo cấu trúc sau:

```
vn.edu.hcmut.cse.adse.lab
|-- controller # Chua API Controller
|-- service # Chua Business Logic
|-- repository # Chua Interface tuong tac DB
|-- entity # Chua Model/Table
\-- StudentManagementApplication.java
```

5 Cấu Hình Database

Lưu ý: *Trong các bài Lab 1, 2 và 3, chúng ta sử dụng* ***SQLite*** *để đơn giản hóa quá trình cài đặt và cấu hình môi trường. Đến* ***Lab 4****, yêu cầu sẽ được nâng cao và các bạn sẽ cần chuyển đổi sang một hệ quản trị cơ sở dữ liệu mạnh mẽ hơn (ví dụ:* ***PostgreSQL****) để phục vụ tính năng mở rộng.*

5.1 Thêm Dependency SQLite

Mở file `pom.xml` và thêm đoạn code sau vào trong thẻ `<dependencies>`:

```
<!-- SQLite JDBC Driver -->
<dependency>
    <groupId>org.xerial</groupId>
    <artifactId>sqlite-jdbc</artifactId>
    <version>3.41.2.1</version>
</dependency>

<!-- Hibernate Dialect cho SQLite -->
<dependency>
    <groupId>org.hibernate.orm</groupId>
    <artifactId>hibernate-community-dialects</artifactId>
    <version>6.2.4.Final</version>
</dependency>
```

Lưu ý: Sau khi thêm, hãy chạy lệnh sau tại thư mục gốc của project để tải thư viện về:

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

5.2 Cấu Hình Kết Nối

Mở file `src/main/resources/application.properties` và dán nội dung sau:

```
spring.application.name=student-management

# Cau hinh file Database SQLite (tu dong tao file student.db tai thu muc goc project)
spring.datasource.url=jdbc:sqlite:student.db
spring.datasource.driver-class-name=org.sqlite.JDBC

# Cau hinh JPA/Hibernate
spring.jpa.database-platform=org.hibernate.community.dialect.SQLiteDialect
spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
```

Giải thích: `ddl-auto=create` có nghĩa là mỗi khi chạy lại app, Hibernate sẽ xóa dữ liệu cũ và tạo lại bảng mới. Khi làm thật (production) ta sẽ đổi thành `update` hoặc `none`.

6 Chạy Thử Ứng Dụng

Để đảm bảo mọi thứ đã được cấu hình đúng, hãy chạy thử ứng dụng.

Dùng dòng lệnh (Terminal) Tại thư mục gốc của project, chạy lệnh:

```
./mvnw spring-boot:run
```

Kết quả mong đợi: Nếu thấy log hiện lên dòng chữ sau là thành công: `Started StudentManagementAppli in ... seconds (process running for ...)` Và file `student.db` sẽ tự động được tạo ra tại thư mục gốc của project.

7 Tạo Dữ Liệu Giả (Mock Data)

Vì chưa có API để thêm dữ liệu, chúng ta sẽ dùng công cụ **DB Browser for SQLite** để "nhồi" trước một ít dữ liệu mẫu, phục vụ cho việc test ở Lab sau.

7.1 Cài đặt & Mở DB

- Tải DB Browser for SQLite.
- Mở app, chọn **Open Database** -> trỏ tới file `student.db` trong project.

7.2 Tạo bảng & Thêm dữ liệu

Chuyển sang tab **Execute SQL** và thực thi đoạn script sau để tạo bảng `students` và thêm dữ liệu mẫu:

```
-- Tao bang sinh vien (giong cau truc Entity Lab 2)
CREATE TABLE IF NOT EXISTS students (
    id INTEGER PRIMARY KEY,
    name TEXT,
    email TEXT,
    age INTEGER
);

-- Them du lieu mau
INSERT INTO students (id, name, email, age) VALUES (1, 'Nguyen Van A', 'vana@example.com', 20);
INSERT INTO students (id, name, email, age) VALUES (2, 'Tran Thi B', 'thib@example.com', 21);
```

Nhấn nút **Run SQL** (biểu tượng ▷).

7.3 Kiểm tra kết quả

- Chuyển sang tab **Browse Data**.
- Chọn bảng `students`.
- Nếu thấy hiển thị 2 dòng dữ liệu của "Nguyen Van A" và "Tran Thi B" là thành công.

8 Bài tập

1. **Dữ liệu lớn**: Hãy thử thêm ít nhất **10 sinh viên** nữa.
2. **Ràng buộc Khóa Chính (Primary Key)**:
   - Cố tình Insert một sinh viên có `id` trùng với một người đã có sẵn.
   - Quan sát thông báo lỗi: `UNIQUE constraint failed`. Tại sao Database lại chặn thao tác này?
3. **Toàn vẹn dữ liệu (Constraints)**:
   - Thử Insert một sinh viên nhưng bỏ trống cột `name` (để NULL).
   - Database có báo lỗi không? Từ đó suy nghĩ xem sự thiếu chặt chẽ này ảnh hưởng gì khi code Java đọc dữ liệu lên?
4. **Cấu hình Hibernate**:
   - Tại sao mỗi lần tắt ứng dụng và chạy lại, dữ liệu cũ trong Database lại bị mất hết?

Lưu ý: Bạn có thể đóng DB Browser sau khi làm xong.