HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO

THỰC TẬP CƠ SỞ

Giảng viên: Kim Ngọc Bách

Tuần 6: 19/4/2025

Sinh viên: Nguyễn Khắc Trường B22DCCN884

MỤC LỤC

A. Kiến thức học được ở tuần này

Giới thiệu

Dự án mà em định triển khai có phần Backend sử dụng framework Spring Boot vậy nên tuần này em đã tìm hiểu về framework này.

Phần 1: Khái niệm về Spring Boot

1. Khái niệm

Spring Boot là một framework Java được sử dụng để xây dựng các ứng dụng và dịch vụ web dễ dàng và nhanh chóng. Nền tảng cung cấp các cấu hình mặc định cho một số thư viện và bộ công cụ hỗ trợ xây dựng, triển khai, quản lý ứng dụng Spring-based. Cách Spring Boot hoạt động nhằm tối ưu hóa quy trình phát triển ứng dụng Java. Điều này sẽ giúp nhà phát triển tập trung vào việc xây dựng tính năng chính của ứng dụng mà không cần phải lo lắng về cấu hình phức tạp.

2. Ưu và nhược điểm

Spring Boot được thiết kế để giúp các lập trình viên tăng tốc quá trình phát triển của họ. Nó giúp loại bỏ việc cài đặt và thiết lập ban đầu tốn thời gian cho môi trường triển khai.

- Ưu điểm chính của Spring Boot:
    - Phát triển ứng dụng nhanh chóng và dễ dàng bằng Spring.
    - Tự động định cấu hình tất cả các thành phần cho ứng dụng Spring cấp sản xuất.
    - Máy chủ nhúng sẵn dùng (Tomcat, Jetty và Undertow) để triển khai ứng dụng nhanh hơn, hiệu quả hơn.
    - Điểm cuối HTTP cho phép bạn nhập các chức năng ứng dụng nội bộ như số liệu, trạng thái sức khỏe và các chức năng khác.
    - Không có cấu hình XML.

- Một lượng lớn các plugin giúp nhà phát triển làm việc dễ dàng hơn với cơ sở dữ liệu nhúng và trong bộ nhớ.
- Dễ dàng truy cập vào cơ sở dữ liệu và các dịch vụ xếp hàng như MySQL, Oracle, MongoDB, Redis, ActiveMQ, v.v.
- Tích hợp liền mạch với hệ sinh thái Spring.
- Cộng đồng lớn và nhiều chương trình đào tạo giúp giai đoạn nhập môn trở nên dễ dàng hơn.

- Nhược điểm của Spring Boot:
  - Thiếu kiểm soát. Spring Boot tạo ra rất nhiều dependency không được sử dụng, dẫn đến file triển khai có dung lượng lớn.
  - Quá trình phức tạp và tốn thời gian để chuyển đổi một dự án Spring cũ hoặc hiện có thành các ứng dụng Spring Boot.
  - Không phù hợp cho các dự án quy mô lớn. Theo nhiều nhà phát triển, mặc dù không có vấn đề gì khi làm việc với microservice nhưng Spring Boot không phù hợp để tạo các ứng dụng nguyên khối.

Phần 2: Cấu trúc dự án của Spring Boot

Trong một ứng dụng Spring Boot, cấu trúc thư mục thường được tổ chức một cách rõ ràng và hợp lý theo từng chức năng riêng biệt, nhằm giúp việc phát triển, mở rộng và bảo trì hệ thống trở nên dễ dàng hơn. Mỗi thư mục đóng vai trò cụ thể trong kiến trúc phân tầng của ứng dụng.

- **config**: Thư mục này chứa các tệp và lớp cấu hình dùng để tùy chỉnh hành vi của ứng dụng. Ví dụ, các file application.properties hoặc application.yml được đặt ở đây để cấu hình các thông số như cổng của server, thông tin kết nối đến cơ sở dữ liệu, cấu hình logging, hoặc các bean cấu hình của Spring.
- **controllers**: Thư mục controllers bao gồm các lớp điều khiển (controller) chịu trách nhiệm xử lý các yêu cầu HTTP từ phía client. Các controller định nghĩa các endpoint RESTful, nhận request, gọi các service để xử lý logic nghiệp vụ, và trả về response phù hợp cho người dùng.
- **enums**: Thư mục này chứa các lớp enum – đại diện cho một tập hợp giá trị hằng số không thay đổi, ví dụ như trạng thái đơn hàng (PENDING,

COMPLETED), vai trò người dùng (ADMIN, USER), hoặc các loại sản phẩm. Việc sử dụng enum giúp code dễ đọc và tránh được lỗi sai khi làm việc với các giá trị cố định.

- **models (entity)**: Đây là nơi lưu trữ các lớp mô hình dữ liệu (hay còn gọi là entity) – phản ánh cấu trúc của các bảng trong cơ sở dữ liệu. Mỗi lớp model thể hiện một đối tượng trong hệ thống, có thể chứa các thuộc tính, mối quan hệ với các đối tượng khác, và thường được ánh xạ với cơ sở dữ liệu thông qua JPA.
- **pojo/dtos**: POJO (Plain Old Java Object) và DTO (Data Transfer Object) là các lớp đơn giản dùng để chứa dữ liệu, không chứa logic nghiệp vụ. Chúng thường được sử dụng để truyền dữ liệu giữa các tầng trong ứng dụng, ví dụ như giữa controller và service, hoặc từ backend đến frontend. DTO giúp tách biệt giữa mô hình dữ liệu nội bộ và dữ liệu trả về cho client, đảm bảo tính bảo mật và rõ ràng.
- **repositories**: Thư mục này bao gồm các interface hoặc lớp có nhiệm vụ tương tác trực tiếp với cơ sở dữ liệu. Thông qua các repository, ứng dụng có thể thực hiện các thao tác như tìm kiếm, lưu, cập nhật, hoặc xóa dữ liệu. Spring Data JPA giúp tự động sinh ra các phương thức truy vấn phổ biến dựa trên tên phương thức.
- **security**: Đây là nơi xử lý tất cả các vấn đề liên quan đến bảo mật ứng dụng, bao gồm cấu hình xác thực người dùng (authentication), phân quyền truy cập (authorization), quản lý phiên đăng nhập, mã hóa mật khẩu, hoặc tích hợp với JWT/OAuth2. Việc tách biệt cấu hình bảo mật giúp hệ thống linh hoạt và dễ kiểm soát hơn.
- **services**: Các lớp trong thư mục services chứa các interface mô tả logic nghiệp vụ cốt lõi của ứng dụng. Service là trung gian giữa controller và repository, đảm nhận việc xử lý dữ liệu và thực thi các quy tắc nghiệp vụ trước khi lưu vào cơ sở dữ liệu hoặc trả về cho client.
- **service implementors**: Đây là nơi triển khai (implement) các interface trong thư mục services. Các lớp này thực hiện chi tiết các thao tác nghiệp vụ bằng cách gọi các repository và xử lý logic phù hợp. Việc tách riêng phần khai báo và phần triển khai giúp ứng dụng tuân theo nguyên tắc SOLID và dễ kiểm thử hơn.

- **utils**: Thư mục utils chứa các lớp tiện ích hoặc các hàm hỗ trợ dùng chung trong toàn bộ ứng dụng. Ví dụ: xử lý định dạng ngày giờ, chuyển đổi kiểu dữ liệu, mã hóa/giải mã chuỗi, đọc ghi tệp, hoặc gửi email. Việc gom các hàm tiện ích vào một nơi giúp tránh lặp lại code và tăng khả năng tái sử dụng.

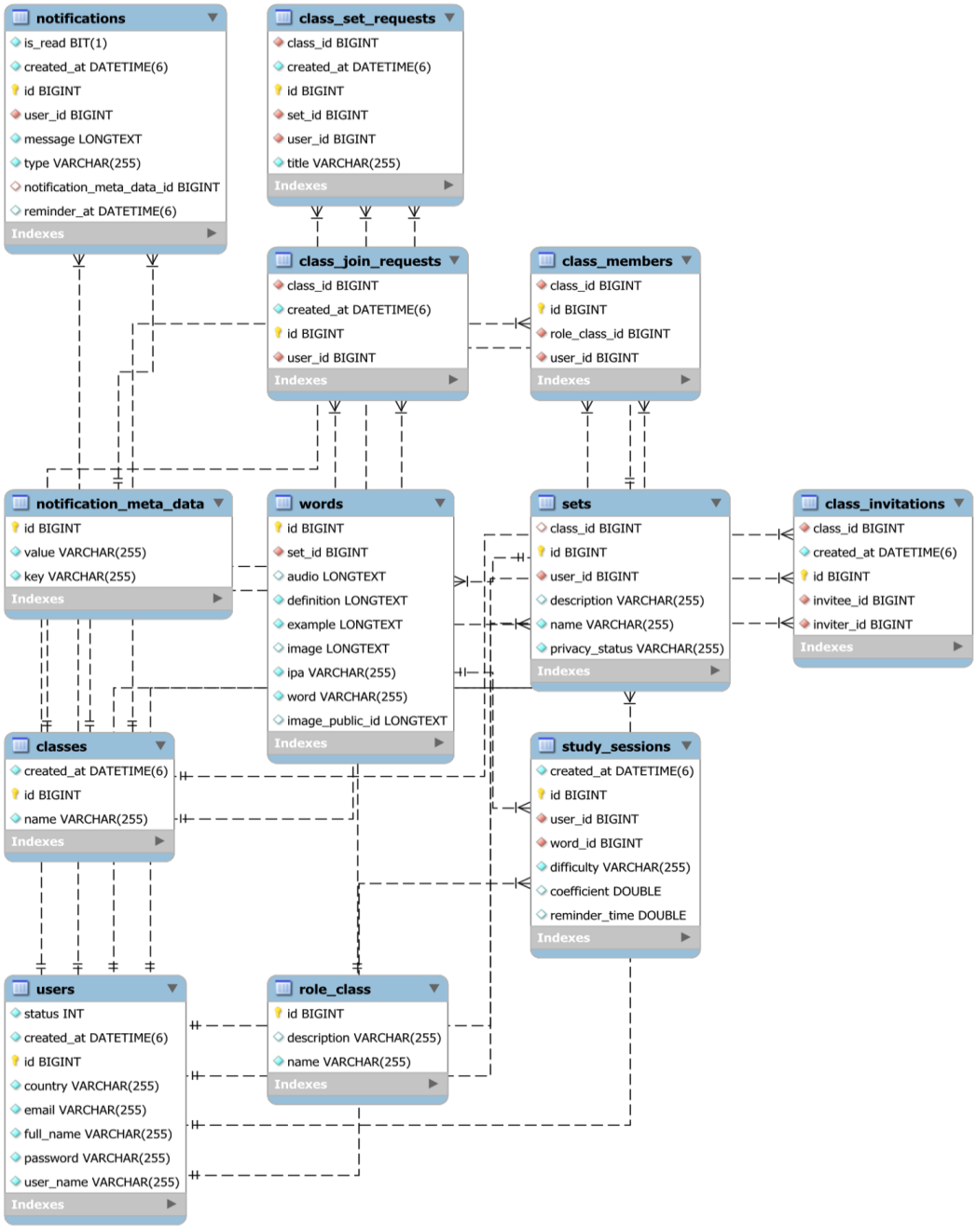
Phần 3: Thực thể (Entity) trong Spring Boot

Trong Spring Boot, Entity là các lớp Java đại diện cho các bảng trong cơ sở dữ liệu quan hệ. Mỗi đối tượng entity tương ứng với một bản ghi (row) trong bảng dữ liệu và được ánh xạ bằng cách sử dụng JPA (Java Persistence API) hoặc framework ORM như Hibernate. Các lớp entity thường được đặt trong thư mục models hoặc entities và được chú thích bằng các annotation như @Entity, @Table, @Id, @GeneratedValue, @Column, @ManyToOne, @OneToMany, v.v.

- Entity đóng vai trò quan trọng trong tầng dữ liệu của kiến trúc ứng dụng. Mỗi entity chứa các thuộc tính tương ứng với các cột trong bảng dữ liệu và có thể mô tả mối quan hệ giữa các bảng, ví dụ như quan hệ một-nhiều hoặc nhiều-nhiều. Điều này giúp nhà phát triển mô hình hóa dữ liệu trực tiếp trong mã nguồn mà không cần viết SQL thủ công, nhờ đó tăng tính trực quan và giảm lỗi phát sinh khi thay đổi cấu trúc dữ liệu.
- Trong thực tế, các entity thường kết hợp với các lớp Repository để truy vấn và thao tác dữ liệu. Tuy nhiên, để đảm bảo nguyên tắc tách biệt trách nhiệm, entity không chứa logic nghiệp vụ mà chỉ đơn thuần là lớp dữ liệu. Ngoài ra, khi làm việc với các API RESTful, các entity cũng có thể kết hợp với các annotation như @JsonIgnore, @JsonManagedReference, @JsonBackReference nhằm kiểm soát cách dữ liệu được serialize/deserialze, tránh lỗi tuần hoàn khi trả về JSON.
- Việc tổ chức entity rõ ràng, hợp lý không chỉ giúp dễ dàng mở rộng và bảo trì hệ thống mà còn tăng tính tái sử dụng, chuẩn hóa kiến trúc và đảm bảo hiệu năng trong các ứng dụng Spring Boot quy mô vừa và lớn.

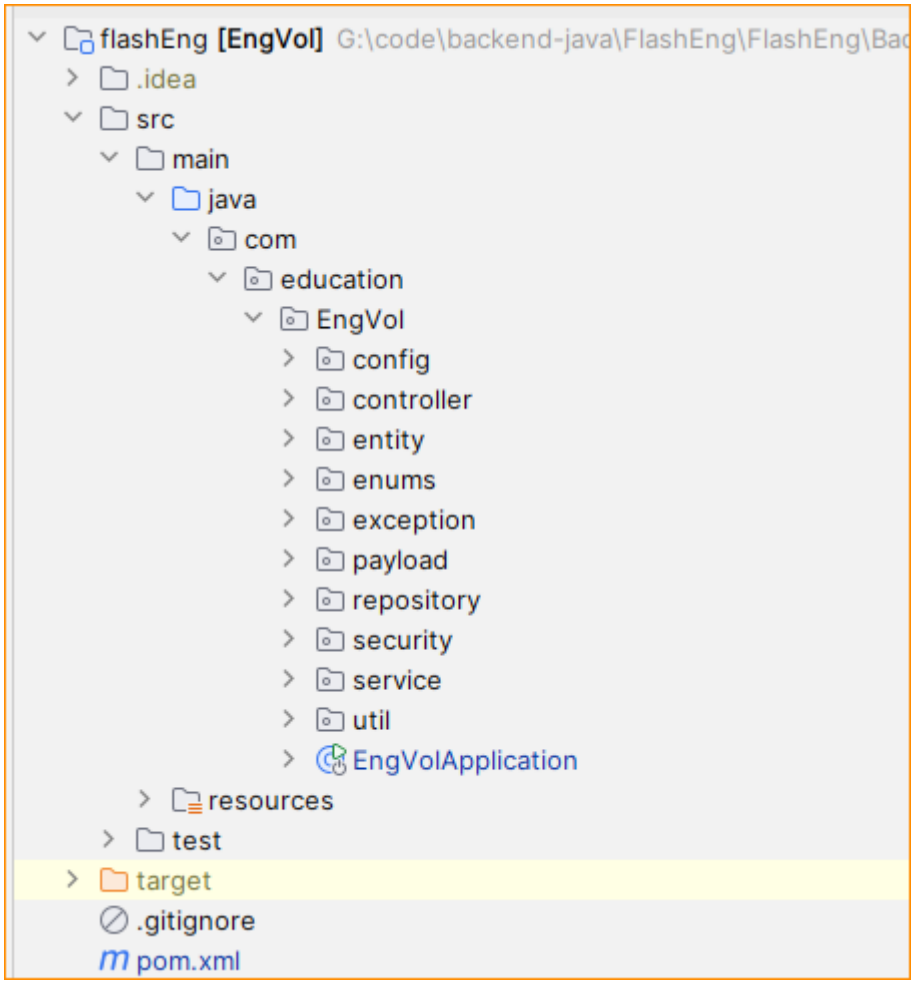
B. Tiến độ dự án

1. Hoàn thiện tạo database trong MySQL



EER Diagram tạo bởi database trong MySQL

2. Tạo được khung cấu trúc Backend



3. Tạo các Entity tương ứng với database