**Day1: 05/09/2023**

1.Theory

Spring Boot is a powerful FW that give everything we need to build an application.

A screenshot of a computer

Description automatically generated

Easy to connect with database.

API Layer(get,post,put,delete) -> Services Layer -> Data Access Layer -> DB

Tool to build software: Maven and Gradle

Maven: base on pure Java language and use XML for creating project structure.

Gradle: base on developing domain – specific language (DSL) for creating project structure.

* Choose Maven project: The process of project building is simplified and well organized.

Pom.xml -> modify the dependence of project

Application.property: Configuration all the property of application as well as environment specific properties

Static and template is use for web development.

Annotation:

@RestController: handle post and get, similar like response and request in Java Core

@GetMapping: handle Get request

@AutoWired: Auto Search for engine, use for Field, Constructor, Getter and Setter. When using this annotation, IOC will look for a portable bean and auto inject it.

**IOC(Inversion of Control) : is a develop rule use in software development for control component of system, the way they interact, the dependency between them.**

@Component: to identify that a class is component or been of Spring. When using this annotation, Spring will auto scan and create a bean for that class -> ready to inject.

@Service: When you annotate a class with @Service , Spring will automatically detect and register it as a bean in the application context. This allows you to easily inject and use the service component in other parts of your application.

Connect to a real database (PostgreSQL)

Application.properties:

Spring.datasource.url = jdbc:postgresql://localhost:5432/amigoscode

Spring.datasource.username=

Spring.datasource.password=

Spring.jpa.hibernate.ddl-auto=create-drop

Spring.jpa.show-sql=true

Spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect

Spring.jpa.properties.hibernate.format\_sql=true

Exam: Grant All privileges on Database "student" to postgres;

Grant All privileges on Database "student" to sa;

\c student : connect to database

\d student : maybe like dir?

JPA and @Entity

Use spring JPA to create a table inside db and CRUD again db

@Repository -> data access

@Configuration: mark that class for Spring Boot to identify new Bean

@Bean: mark the method that Spring Boot know this is Bean give to Context (CRUD)

@PostMapping: use when add more resources

A screen shot of a computer program

Description automatically generated

Post: create

Get: use to retrieve

Put: update

Delete: delete

A screenshot of a computer

Description automatically generated

@Transactional : annotation for update method that when the method is finished, ny changes made to JPA-managed entities within that method are automatically synchronized with the database when the transaction is committed.

Link github of demo project: [ThanhPCHE170611/DemoSpringBoot: This project is my Demo of SpringBoot FW (github.com)](https://github.com/ThanhPCHE170611/DemoSpringBoot)

**Day2: 06/09/2023**

Learning Target: 1. Complete User Login and Registration Backend

1.

Config for application.property or application.yml

Setup database: using psql for PostgreSQL command line

\l : list all database

Create a new Package to handle AppUser

In AppUser package, create AppUser Entity Class that implement UserDetails -> to do with security

Coding properties for AppUser

Then, finish other override methods with what you actually want it be (methods implement from UserDetails)

Create Constructors

Lombok: using @Getter and @Setter to define for class with no need to create new

Create Service Class

Create Repository

Create Controller

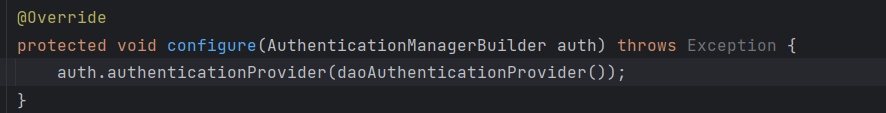
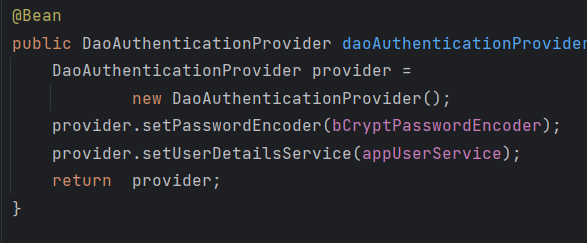
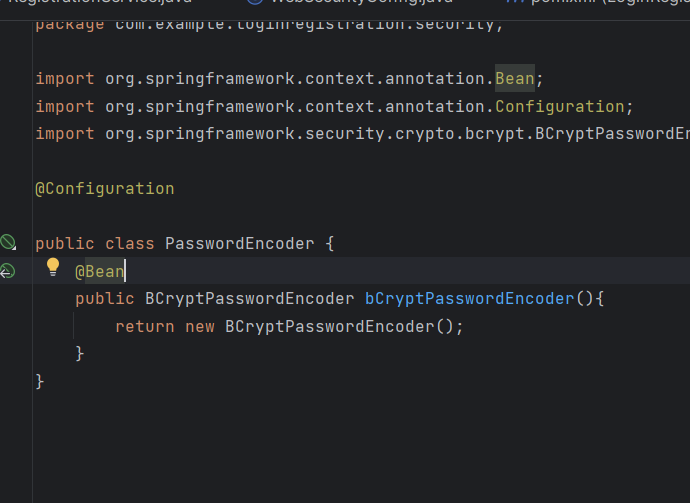
Create new package Registration to handle registration: registration controller, service, request

Create new package security: create a config package inside to config security

In Config package, create WebSecurityConfig to Config web security, the class will extends

WebSecurityConfigurerAdapter. I got problem in this step that can’t extend the right class. Fix by edit pom.xml -> change version to <version> 2.7.15 </version>

First, configuration for websecurity example:  


Then setup for the Bcrypt Passwordd:   
 

Connect to db and test the table creating using post environment (do not learn create html css yet)

* Summarize day2: learning about spring security, Encrypt Password by Bcrypt, using Lombok tool in coding.
* **Day3: 07/09/2023**

1. Learning about session
2. Using thymeleaf tool to write a login, register, maybe home page.

Similar like Session in java.

Im coding a simple website to understand more about thymeleaf and session both.

When finished. I figure out that thymeleaf isn’t similar to jsp. It have diff template to initiative and call parameter. When you use session in coding, you still can setAttribute() and getAttribute(), which method use session need to input HttpSession into. I use session in store user information when login and setTimeout for that session for 30 minutes. After that time, the session automatically invalidates and user have to login again if want to access the website. I didn’t try to using 2 threads (one in normal and one in private mode) to illustrate that if you login normal way and turn the brower off that you till access in private mode. But with my knowledge, I think this can’t be true. With the remain time, I will try to encode password then save to db and decode when user input to check.

First, adding spring security in pom.xml. This dependency is necessary to using Bcrypt in spring boot

Next, create new package name Security and give access permission for each situation and page.

In Security, create SecurityConfig class and extend to WebSecurityConfigurerAdapter:  
A computer screen shot of a computer code

Description automatically generated

Must have PasswordEncoder interface to use Bcrypt encode.

Change logic of register controller. Before save password to db, the system will call PasswordEncoder to encode password to Hash format and then save to db:

A screen shot of a computer code

Description automatically generated

When user input password to login, the system will compare the input of user with the encode password in db by method matches by PasswordEncoder interface to authorize.

A screen shot of a computer

Description automatically generated

* **Day4: 08/09/2023**

Learn about Object Mapping

Object Mapping in Spring Boot is the transfer process between Java Objects and data from other sources, eg: json, xml, etc… Spring provide tools and library for easier object mapping

Two main parts when work with object mapping in Spring Boot:

1. Object-to-Data: This process transfer data from Java Object to other data type like json or xml. This usually happened when you want to get data from system in HTTP form or store data to db.
2. Data-to-Object: This process transfer data from other resources eg: db,… to Java object so that you can work with it in application environment.

Some way to use object mapping:

Spring Data JPA: a part of Spring Framework that access transfer between java object and db using repository interface.

Jackson: strong library to transfer between Java and Json. It automatically transfer JSON to Java Object and either.

….

Learn about Entity annotation and relationship annotation:

Software requirement specification: teacher n – n class

Student n – 1 class

Student n – n subject

Teacher n – n subject

Teacher n – n student

Subject n – n class

User table -> for teacher and student, properties:

Id, name, username, password, status, role

Class table -> for class entity, properties:

Id, name

Subject table -> for Subject entity, properties:

Id, name