ManyToMany -> phải tạo ra 1 table phụ để chứa cả idA và idB

OneToMany -> A chưa idb hoặc ngược lại

Controller gọi Service, Service gọi Repository và Mapper

Tạo Repository -> Tạo Service -> Tạo DTO (Req và Resp) -> Tạo Mapper -> Tạo Controller

Thứ tự:

Entity ->dto (request, response), repository -> mapper, (passwordEncoder) -> service -> controller

ANOTATION

1. @Value() => Dùng để đọc biến từ file yaml

TOKEN

Để tạo 1 token cần

+ 1 header

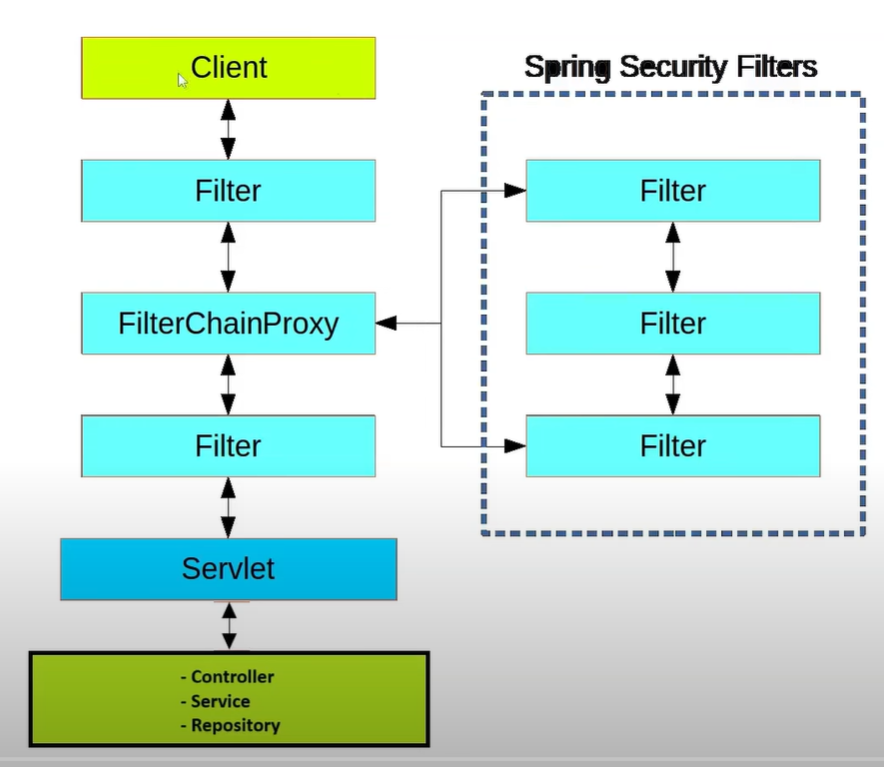
+ body => Nội dung gởi đi kèm trong token (username, issuer -> Định danh token do ai issuer, issuerTime)

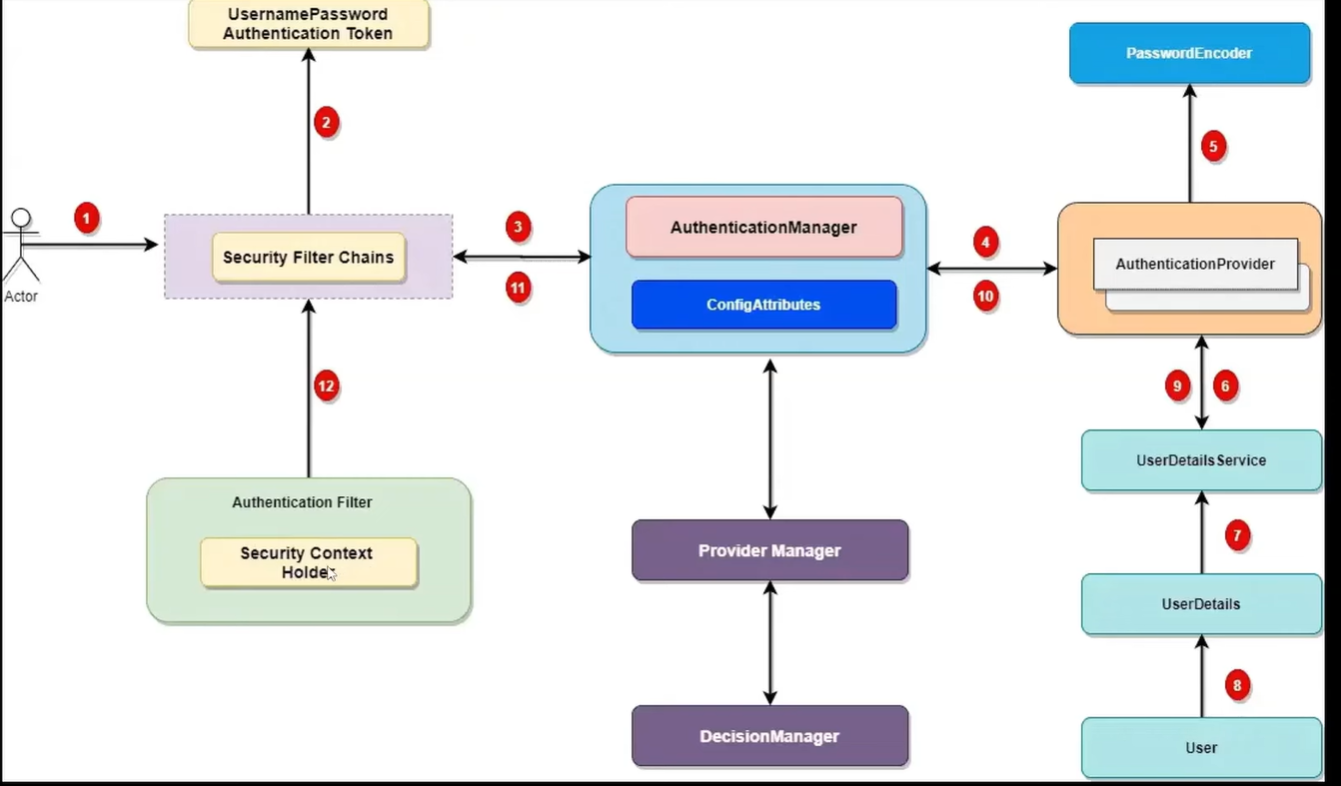
+ payload

* Sign token

Spring Security Architecture

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-oauth2-resource-server</artifactId>  
</dependency>





**@RequiredArgsConstructor  
@FieldDefaults**(**level** = AccessLevel.*PRIVATE*, **makeFinal** = true) *// Thay the AutoWide*

**@Slf4j => Log**

**SPRING SECURITY**

* Chỉ lấy được thông tin của chính mình

**@PostAuthorize**("returnObject.username == authentication.name") *// Goi method truoc va sau do kiem tra xem co phai Role mong muon hay khong moi tra ket qua?, authentication.name la USER hien tai*public UserResponse getUser(String id){  
 *log*.info("In method gerUsersByID");  
 return userMapper.toUserResponse(userRepository.findById(id). *// Nếu tìm thấy thì trả về* orElseThrow(() -> new AppException(ErrorCode.*USER\_NOT\_EXISTED*))); *// Còn không tìm thấy thì báo lỗi*}

Trong springSecurity khi mà User được xác thực thành công để đăng nhập thì thông tin của User đó được lưu trữ trong SecurityContextHolder

GraalVM

Bài 14: Validation bằng custom anotation

*Vấn đề: Tạo 1 custom anotation để validation DOB*

1. Tạo Package validator

1.1 Interface Anotation

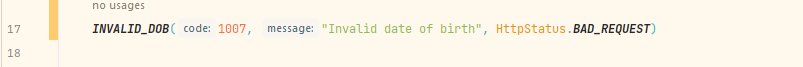
1.2 Class Validator

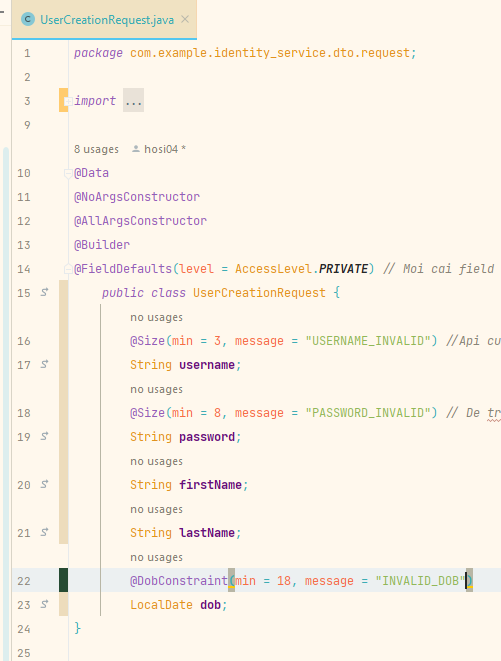
1. Click nào 1 cái anotaion nào đó có sẵn để xem cấu trúc khai báo
2. Target: tức là cái anotaion này sẽ được apply ở đâu? (ở 1 biến => field)
3. Retention: anotation này sẽ được xử lý lúc nào?
4. Constraint: Class sẽ chịu trách nhiệm validation cho anotation này

- Cấu trúc cở bản của 1 anotation









ChronoUnit: Có nhiều phép toán hữu ích trong ngày tháng

Bài 14: Validation bằng custom anotation (ADVANCE)

Cải tiến trong GlobalExceptionHandler

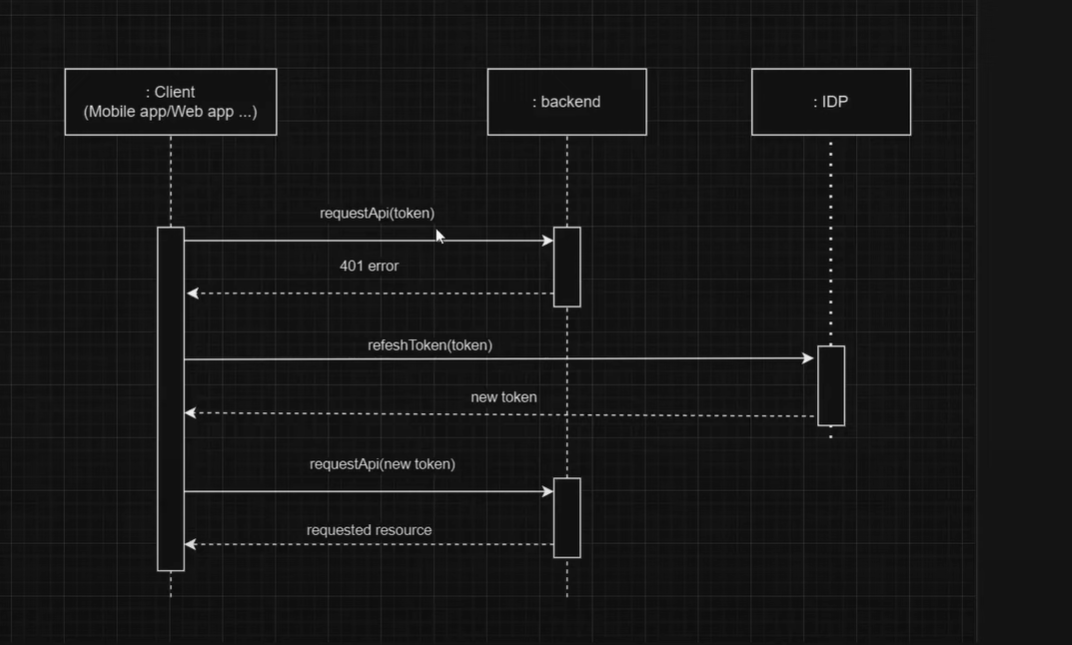
BÀI 15: logout with JWT token

* Thêm jwtID vào generateToken
* Thêm 1 entity InvalidatedToken (id, expiryTime)
* Tạo 1 repository InvalidatedTokenRepository
* Tạo 1 hàm Logout trong AuthenticationService để thực hiện chức năng logout

refresh JWT token

* Tạo 1 function refreshToken trong AuthenticationService(refrestRequest req)
* Tạo 1 req nữa
* Tạo 1 controller trong AuthenticationController

refresh JWT token (phần 2) align với Oauth2



Unit Test

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
</dependency>

<dependency>  
 <groupId>com.fasterxml.jackson.datatype</groupId>  
 <artifactId>jackson-datatype-jsr310</artifactId>  
</dependency>

* 1 cái req đi vào thi đầu tiên nó sẽ qua Controller sau đó quá Service sau đó đến repository (Gần như là lớp cuối cùng chứa req)
* Tập trung viết Unit Test cho Controller và Service

Ví dụ tạo Test Controller mà trong Controler có gọi đến function của Service thì chúng ta sẽ cần Mock cái service đấy. mục đích kiểu như là nó sẽ bỏ qua cái function của Service và chỉ test cái Controller

METHOD Builder là của Lombok

Serrie UnitTest và testing mới học 1 video đầu thôi, nhớ phải quay lại học serrie này

SONARQUBE cũng chưa làm xong, sau nhớ tìm hiểu và quay lại học

format code with Spotless

<properties>

<java.version>17</java.version>  
 <projectlombok-lombok.version>1.18.30</projectlombok-lombok.version>  
 <mapstruct.version>1.5.5.Final</mapstruct.version>  
 <lombok-mapstruct-binding.version>0.2.0</lombok-mapstruct-binding.version>  
 <spotless.version>2.43.0</spotless.version>  
   
</properties>

<plugin>  
 <groupId>com.diffplug.spotless</groupId>  
 <artifactId>spotless-maven-plugin</artifactId>  
 <version>${spotless.version}</version>  
 <configuration>  
 <java>  
 <removeUnusedImports />  
 <toggleOffOn/>  
 <trimTrailingWhitespace/>  
 <endWithNewline/>  
 <indent>  
 <tabs>true</tabs>  
 <spacesPerTab>4</spacesPerTab>  
 </indent>  
 <palantirJavaFormat/>  
 <importOrder>  
 *<!-- Specify either order or file, but not both -->* <order>java,jakarta,org,com,com.diffplug,</order>  
 </importOrder>  
 </java>  
 </configuration>  
 *<!-- <executions>-->  
 <!-- <execution>-->  
 <!-- <phase>compile</phase>-->  
 <!-- <goals>-->  
 <!-- <goal>check</goal>-->  
 <!-- </goals>-->  
 <!-- </execution>-->  
 <!-- </executions>-->*</plugin>

//spotless:off

Code trong này sẽ không bị format bởi spotless

//spotless:on

Unique field trong JPA và bài toán concurrent request.

Cài đặt Jmetter

<https://www.youtube.com/watch?v=VPDaC2JtmeM&list=PL2xsxmVse9IaxzE8Mght4CFltGOqcG6FC&index=28>

* number of threads: Giả lập có bao nhiêu user cùng thực hiện req 1 lúc
* ramp-up: thời gian để init lên số lượng thread ví dụ number of threads = 10 và ramp-up = 1 thì có nghĩa là trong 1s sẽ tạo ra 10 req
* loop count: Số lần chạy

Khi chưa xử lý **concurrent** khi có nhiều req thực hiện cùng 1 lúc thì việc kiểm tra existsByUser là **không** có tác dụng

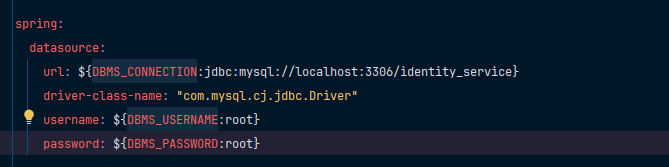
* Khắc phục: Gán field userName là UNIQUE

Tìm hiểu về Profile và biến môi trường

Cách set profiles

run/debug configuration => edit => modify options => add VM options => -Dspring.profiles.active=ten\_profile

BIẾN MÔI TRƯỜNG

* Thông thường thì chương trình sẽ được deloy dưới 1 hdh hoặc máy ảo nào đó, trong hdh thì có nhiều biến môi trường mà từ đó chương trình có thể đọc và get thông tin ra
* Thay vì đọc thông tin từ **file yaml** thì ta sẽ đọc thông tin từ **biến môi trường**
* Từ biến local chuyển sang biến môi trường thì phải theo các quy tắc sau:
  + Dấu **chấm** chuyển thành dấu **gạch dưới**
  + VD: JAVA\_HOME
* url: ${DBMS\_CONNECTION:jdbc:mysql://localhost:3306/identity\_service}
  + DBMS\_CONNECTION là biến môi trường và nếu như ta khai báo biến môi trường này thì chương trình sẽ overwrite là sử dụng giá trị của biến đó, còn nếu ta không khai báo biến môi trường thì chương trình nó sẽ lấy giá trị sau dấu “:”
* Tùy theo các dự án khác nhau mà ta có thể dùng hay không dùng biến môi trường
* Cách khai báo biến môi trường
  + 
  + Ví dụ muốn khau báo biến url thì tên biến môi trường đó là: SPRING\_DATASOURCE\_URL, hoặc có thể sử dụng cách như hình trên

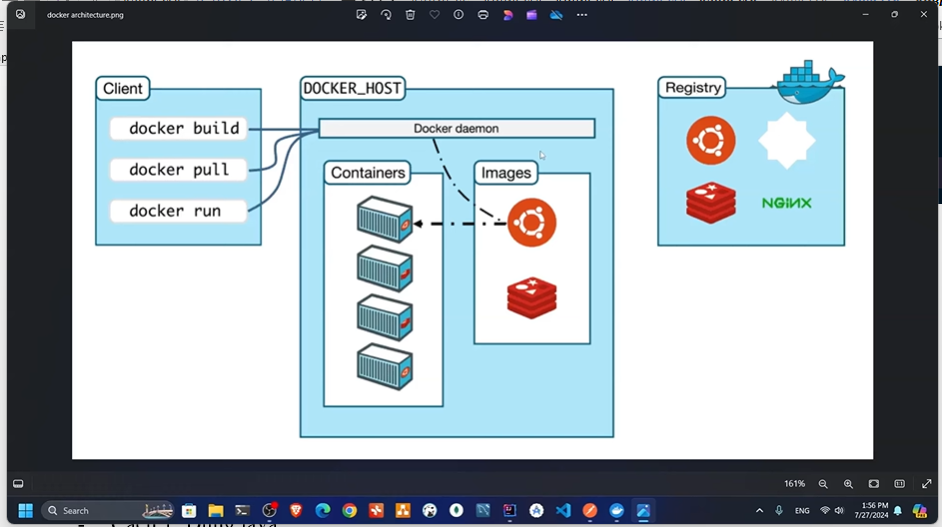
#27 Đóng gói và triển khai ứng dụng

* ./mvnw package trong local, lệnh này build cả UnitTest
* ./mvnw clean => sẽ clean thư mục trong target
* ./mvnw package -DskipTests => Build và bỏ qua UnitTest

Để chạy file .jar

* Cách 1: Dùng java
  + Mở terminal gõ lệnh java -jar ten\_file.jar
  + Ctrl C để tắt
* Cách 2: Dùng maven
  + Máy phải có maven
  + Mvn package -DskipTests
* Cách 3: Dùng tool của Intelij
  + Maven ở thanh bên phải => liftCycle => package

#28 build Docker image



* Image: Là file chứa toàn bộ hệ điều hành và chương trình của chúng ta
* Containers (1 hệ điều hành hoàn chỉnh): Là các images sau khi được chạy, và các containers có thể chạy độc lập với nhau
* Registry: Là nơi chứa các images, phổ biến là DockerHub
* Docker pull: Để tải các images trên Registry về local của chúng ta

DOCKER IMAGE

**Lệnh xem danh sách các image trong terminal**

Docker image ls

**Docker guideline**

**Build docker image**

docker build -t <account>/identity-service:0.9.0 .

**Push docker image to Docker Hub**

docker image push <account>/identity-service:0.9.0

**Create network:**

docker network create hosi-network

**Xem danh sách các Network**

Docker network ls

**Start MySQL in devteria-network**

docker run --network hosi-network --name mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=root -d mysql:8.0.36-debian

**Run your application in devteria-network**

docker run --name identity-service --network hosi-network -p 8080:8080 -e DBMS\_CONNECTION=jdbc:mysql://mysql:3306/identity service -d hosinguyen/identity-service:0.9.0

EC2

EC2: Là 1 máy chủ ảo, dùng để deploy ứng dụng

Termius

Sudo docker ps -a => Danh sách các images đang chạy

Sudo docker rm NAMES => remove images