[사전학습] spring-boot 기본

spring-boot 의 기본 사용법을 익힌다.

maven 에서 스프링부트 어플리케이션 시작

- **터미널을 열어서 order 서비스를 실행해본다.**
    - **cd pre-lab-springboot/order**
    - **mvn spring-boot:run**
- **package 명령으로 jar 파일을 생성후 실행해 본다**
    - **mvn package**
    - **java -jar target/order-0.0.1-SNAPSHOT.jar**

pom.xml 에 기본 설정

- **order 서비스의 pom.xml 을 열어본다.**
- **디펜던시를 추가하여 필요한 라이브러리를 구성한다.**
- **spring-boot-starter-web 으로 변경 (Tomcat 시작)**
- **spring-boot-starter-webflux 으로 변경 (Netty 시작)**

리소스 설정

- **스프링부트의 기본 리소스 경로는 resource 이다.**
- **template/1-index.html 폴더의 내용을 복사하여 resource/static/index.html 파일 생성한다.**
- **서버 시작 후 http localhost:8080 호출하여 index.html 파일 내용을 확인 한다.**

포트 변경

- **application.yml 파일에 아래와 같이 8081 포트로 변경**

```
server:
      port: 8081
```

- **서버 시작 후 http localhost:8081 로 접속확인**

make REST API

Controller 패턴

- 2-controller 폴더 파일을 order 서비스에 복사한다
- OrderController.java 파일을 참조하여 REST 구성법을 익힌다.
- 서버를 실행 후 아래와 같은 명령으로 REST call 을 호출하여 테스트한다.
    - http localhost:8081/order
    - http POST localhost:8081/order orderId=2 productName=TV qty=2
    - http PUT localhost:8081/order/2 productName=RADIO qty=4

Repository 패턴

- pom.xml 에 다음과 같은 디펜던시 추가

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
	<groupId>com.h2database</groupId>
	<artifactId>h2</artifactId>
	<scope>runtime</scope>
</dependency>
```

- 3-repository 폴더 파일을 order 서비스에 복사한다
- (Order.java 파일을 open 하여 클래스 위에 @Entity 가 있는지 확인)
- Entity 와 Repository 패턴에 의하여 기본적인 REST API 가 생성된다.
    - http localhost:8081/orders
    - http POST localhost:8081/orders productName=TV qty=2
    - http PATCH localhost:8081/orders/1 productName=RADIO qty=4

call REST API

- 4-call-rest-api 폴더 파일을 order 서비스에 복사한다
- ApiController.java 파일을 참조하여 REST 호출법을 익힌다.
- product 서비스에 직접 호출하여 상품 정보 가져오기
    - http http://localhost:8082/products/2

- **order 서비스를 통하여 상품 정보 가져오기**
    - **http http://localhost:8081/getProduct/2**
- **order 서비스를 통하여 상품 정보 추가하기**
    - **http POST http://localhost:8081/saveProduct productName==“mic” stock==1**

기타

env 활용

- **product 서비스의 ProductController.java 파일을 열어본다.**
- **application.yml 파일의 설정값을 코드에서 호출하여 본다.**
    - **http http://localhost:8082/env**
    - **http http://localhost:8082/envDefault**
- **서비스를 실행시 환경변수를 주입하여 실행한다.**
    - **mvn spring-boot:run -Dspring-boot.run.arguments=–api.url.delivery=http://delivery:8080**
    - **http http://localhost:8082/envDefault**
- **서비스를 실행시 환경변수를 스프링부트 어플리케이션에서 읽어서 실행한다.**
    - **export _DELIVERY_URL=http://delivery:8080**
    - **mvn spring-boot:run**

repository 활용

- **product 서비스의 ProductRepository.java 파일을 참고한다.**
- **Repository 파일에 쿼리를 적을시 아래 url 에서 확인할 수 있다.**
    - **http http://localhost:8082/products/search**
    - **http http://localhost:8082/products/search/findByName?name=TV**
    - **http http://localhost:8082/products/search/findByName name==TV**
    - **http http://localhost:8082/products/search/findByStockAndName nameTV stock10**
    - **http http://localhost:8082/products/search/getName?name=TV**

checkPoints 작업가이드

- **상품서비스에 CUP 이라는 이름으로 post 로 등록한다.**
- **주문서비스의 Controller 에서 @GetMapping("/getProductByName") 을 생성한다.**
- **내부 구현체에 /products/search/findByName 을 호출하는 로직을 구현한다.**

CheckPoints

1. 상품서비스에 상품 이름 CUP 을 등록하시오. ☐

2. 주문서비스(8081)에 getProductByName 이라는 API 를 생성 해보시오. ☐

3. http://localhost:8081/getProductByName?productName=CUP 으로 호출 하였을때, 상품정보가 조회되도록 만드시오.