springboot test 작성 연습

springboot 에서 테스트를 활용하는 자세한 방법은 공식문서를 참고하면 된다.

- https://docs.spring.io/spring-boot/docs/2.3.2.RELEASE/reference/html/spring-boot-features.html#boot-features-testing

테스트 설정

- springboot (ver 2.3.x) 는 JUnit4 와 JUnit5 를 동시에 사용이 가능하다.
- 두가지 버전을 동시에 사용하기때문에 import 시 주의를 해야한다.
- JUnit4 를 사용하기 위해서는 테스트클래스 상단에 @RunWith 어노테이션을 반드시 붙여야한다.
- JUnit5 를 사용하기 위해서는 어노테이션을 안붙이면 된다.
- 라이브러리 충돌을 피하기 위해서, JUnit5 만 사용하기를 원한다면 아래와 같이 설정을 하여준다.

```
<dependency>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-test</artifactId>
	<scope>test</scope>
	<exclusions>
		<exclusion>
			<groupId>org.junit.vintage</groupId>
			<artifactId>junit-vintage-
engine</artifactId>
		</exclusion>
	</exclusions>
</dependency>
```

Unit test

1. product 서비스의 src/test/java/shopmall/ProductTest 를 연다.

- @DataJpaTest 는 JPA 의 Entity 를 테스트 한다.
- TestEntityManager 를 통하여 가상의 데이터를 데이터베이스에 넣은 후에, 해당 데이터를 조회해 보는 테스트 이다.

- 테스트방법
    - 상품_생성_테스트() 를 run 하여 본다.
    - mvn test -Dtest=shopmall.ProductControllerTest
    - then 부분에 틀린 데이터를 넣어서 에러를 살펴 본다.

2. product 서비스의 src/test/java/shopmall/ProductRestTest 를 연다.

- ProductTest 클래스와는 다르게, @RunWith 가 없어서 JUnit5 로 동작한다.
- JUnit4 : @Before , JUnit5 : @BeforeEach 로 어노테이션이 변경됨
- @RestClientTest 는 Rest call 을 테스트한다.
- MockRestServiceServer 를 통하여 가상의 서버를 생성 한 후, 특정 url 로 호출시 productsData.json 데이터를 리턴하는 코드를 생성하였다.
- restTemplate 을 사용하여 특정 url 을 호출 하고, 결과값을 비교한다.
- 테스트방법
    - getProductsTest() 를 run 하여 본다.
    - mvn test -Dtest=shopmall.ProductRestTest
    - assertThat 부분에 틀린 데이터를 넣어서 에러를 살펴 본다.

3. product 서비스의 src/test/java/shopmall/ProductControllerTest 를 연다.

- @WebMvcTest 는 MVC 패턴의 Controller 를 테스트 할때 사용한다.
- Controller 에는 여러 서비스들을 호출 하는 경우가 많아서, 해당 모듈들을 모두 로딩할 수 없으니, @MockBean 객체로 로딩한다.
- given 으로 @MockBean 으로 선언된 객체들의 가상의 값을 선언하여 준다.
- 컨트롤러에 선언된 api 들을 테스트 한다.
- 테스트방법
    - productStockCheck() 를 run 하여 본다.
    - mvn test -Dtest=shopmall.ProductControllerTest
    - andExpect 부분에 틀린 데이터를 넣어서 에러를 살펴 본다.
    - andExpect 부분에 stock 을 추가하고, 테스트를 성공시킨다.

Integration test

1. product 서비스의 src/test/java/shopmall/ProductController2Test 를 연다.

- springboot 에서 통합테스트는 @SpringBootTest 어노테이션을 붙인다.
- 통합테스트는 단위테스와는 다르게, 테스트에 필요한 모든 의존성을 제공한다.
- 해당 코드는 스프링부트가 기동되면서 application.yaml 파일의 설정값을 읽고, 컨트롤러에서 해당 설정값을 rest call 로 가져오는 예제이다.
- 테스트방법
    - getEnvValue() 를 run 하여 본다.
    - mvn test -Dtest=shopmall.ProductController2Test
    - application.yaml 파일의 api.url.order 의 값을 다르게 변경 한 후 테스트를 다시 돌려서, 테스트 실패하는지 확인한다.
    - MockMvcRequestBuilders.get("/envValue") 부분을 "/env" 로 변경하여 테스트를 돌리고, 실패시 어떻게 고칠수 있는지 고민한다.

해당 서비스가 카프카를 사용시 테스트 코딩 방법

1. 단위테스트시 해당 서비스가 카프카를 사용중이라면 별도의 코딩이 필요 없다.

- order 서비스의 src/test/java/shopmall/OrderTest 참고

2. 통합테스트시 해당 서비스가 카프카를 사용중이라면 별도의 코딩이 필요 하다.

- order 서비스의 src/test/java/shopmall/OrderControllerTest 를 연다.
    - 해당 파일을 run 한다.
    - mvn test -Dtest=shopmall.OrderControllerTest
    - localhost:9092 에 카프카가 기동되어있는 상태에서는 테스트가 성공한다.
- 하지만 실제 CI 환경에서는 카프카가 기동되어있지 않고, 카프카를 중지시, 서비스는 카프카를 연결하려고 계속 시도하게 된다.
    - 카프카 서비스를 중지한다.
        - $kafka_home/bin/kafka-server-stop.sh
    - OrderControllerTest 를 run 한다.
    - mvn test -Dtest=shopmall.OrderControllerTest

    - could not be established. Broker may not be available 에러가 뜨면서 카프카 연결을 실패하여 테스트가 진행이 안된다.
    - ctrl+c 로 서비스를 중지 한다.

- 서비스에서 카프카를 연결시 정상적으로 통합 테스트가 되게 하려면 kafka test 디펜던시가 필요하다.
    - pom.xml 에 아래 디펜던시 추가

```
<dependency>
        <groupId>org.springframework.kafka</groupId>
        <artifactId>spring-kafka-test</artifactId>
        <scope>test</scope>
</dependency>
```

- application.yaml 에 별도의 profle 을 만들고, 카프카 주소를 ${spring.embedded.kafka.brokers} 로 변경한다.

```
spring:
  profiles: test
  cloud:
    stream:
      kafka:
        binder:
          brokers: ${spring.embedded.kafka.brokers}
```

- 이제 통합 테스트를 하려는 파일 (@SpringBootTest) 에서 @EmbeddedKafka 를 선언하여 내부 카프카로 동작을 시킨다.
- 내부 카프카의 주소는 profiles: test 에 있기 때문에 테스트시 프로파일을 변경하도록 한다. -> @ActiveProfiles("test") 적용

checkPoints 작업가이드

- 카프카 서비스 중지
    - $kafka_home/bin/kafka-server-stop.sh
- 카프카 서비스 실행
    - IDE 상단의 labs -> start kafka server
    - netstat -anp | grep 9092 실행시 LISTEN 이 되어있으면 됨

Checkpoints

1. 카프카 서비스를 중지하시오.(체크박스는 체크안됩니다.)

2. 카프카 서비스를 중지한 후 order 서비스의 통합 테스트코드를 완성 하여 테스트를 성공시키시오.