스프링 부트와 JPA 활용 1

1. 프로젝트 환경 설정

1.1 사용 라이브러리

web(톰캣 포함)

thymeleaf(template engine)

jpa

h2(DB)

lombok(getter, setter같은 중복 코드를 annotation을 통해 쉽게 만들어주는 패키지)

validation

1.2 JPA와 DB 설정

.yaml 파일과 .properties 파일 중에 고르면 되는데, 설정이 많아지면 yaml 파일이 편하다.

```
#yaml 파일
spring:
  datasource:
    url: jdbc:h2:tcp://localhost/~/jpashop
    username: sa
    password:
    driver-class-name: org.h2.Driver

  jpa:
    hibernate:
      ddl-auto: create
# ddl-auto:create는 애플리케이션 실행 시점에 테이블을 드랍하고 새로 생성해주는 기능
    properties:
      hibernate:
#     show_sql: true
# System.out을 통해 log를 찍는다.
        format_sql: true
logging:
  level:
    org.hibernate.SQL: debug
#log를 logger를 통해 찍는다.
```

만약 설정 파일이 궁금하면 SPRING 공식 문서를 봐라!

커맨드와 쿼리를 분리해라. 사이드 이펙트가 없도록, 저장을 하고 나면 가급적이면 return값을 member 전체를 return하는 것이 아니라, ID 정도만 return한다.

entity manager를 통한 모든 데이터 변경은 항상 transaction 안에서 이루어져야 한다. 스프링 기본편에 잘 설명되어 있음!

@Transactional annotation이 @Test와 함께 있으면 테스트가 끝난 후 db를 roll back 시킨다.

만약 @Rollback(false) annotation이 같이 붙어 있으면 Rollback 하지 않고 commit 한다.

query parameter 찍는 법

application.yml에서 org.hibernate.type : trace로 주면, value의 로그를 찍어준다.

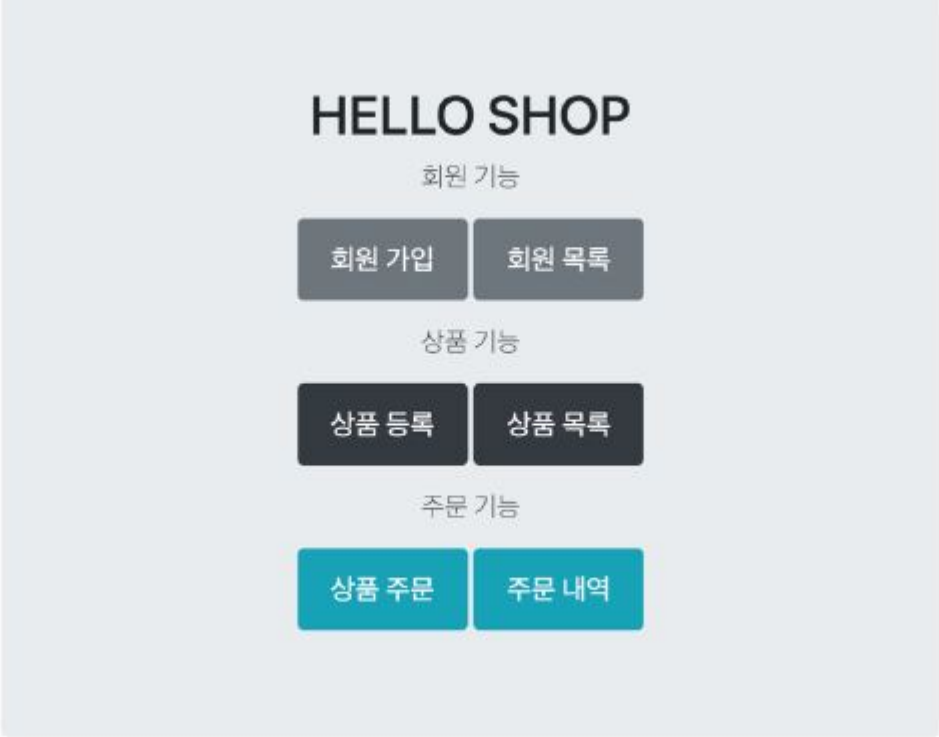
또는 https://github.com/gavlyukovskiy/spring-boot-data-source-decorator의 P6spy를 사용해서 로그를 찍는다.

gradle.build에서 버전이 있는 라이브러리는 스프링에서 미리 버전이 설정되지 않은 라이브러리들이다. 버전이 없는 라이브러리는 스프링에서 자동으로 잘 맞는 버전을 지정해 두었다.

쿼리 파라미터를 로그로 남기는 외부 라이브러리는 시스템 자원을 사용하므로, 개발 단계에서는 편하게 사용해도 된다. 하지만 운영시스템에 적용하려면 꼭 성능테스트를 하고 사용하는 것이 좋다.

2. 도메인 분석 설계

2.1 요구사항 분석



양방향 로직 보다는, 단방향 로직을 짜야 한다!

객체는 collectin을 이용해 다대다 관계를 그냥 만들 수 있지만, 관계형 데이터베이스는 중간에 테이블을 두고 다대다 관계를 1대다, 다대1 관계로 풀어내야한다.

1 대 1 관계에서는 보통 Access가 더 자주 일어나는 테이블에 foreign key를 둔다.

연관관계의 주인은 foreign key와 가까운 테이블로 하면 좋다.

실무에서는 @ManytoMany는 거의 안씀. 필드를 추가할 수 없다.

@Embeddable이 붙은 값 타입은 생성자로 값을 정하고, setter로 변경할 수 없어야한다.

X to One 관계는 fetchtype을 lazy로 바꿔줘야한다.

컬렉션은 생성자에서 주입 받지 말고 필드에서 바로 초기화한다.

컬렉션을 생성한 이후 바꾸지 말고, 그냥 사용해라!