[학습내용]스프링 부트와 JPA 활용 1

▼ 프로젝트 환경 설정

java 11

빌드도구: gradle

설정 파일: yml

▼ 활용 라이브러리

- 스프링 부트 라이브러리
  - spring-boot-starter-web
    - spring-boot-starter-tomcat: 톰캣 (웹서버)
    - spring-webmvc: 스프링 웹 MVC
  - spring-boot-starter-thymeleaf: 타임리프 템플릿 엔진(View)
  - spring-boot-starter-data-jpa
    - spring-boot-starter-aop
    - spring-boot-starter-jdbc
      - HikariCP 커넥션 풀 (부트 2.0 기본)
    - hibernate + JPA: 하이버네이트 + JPA
    - spring-data-jpa: 스프링 데이터 JPA
  - spring-boot-starter(공통): 스프링 부트 + 스프링 코어 + 로깅
    - spring-boot
      - spring-core
    - spring-boot-starter-logging
      - logback, slf4j

- 테스트 라이브러리
  - spring-boot-starter-test
    - junit: 테스트 프레임워크
    - mockito: 목 라이브러리
    - assertj: 테스트 코드를 좀 더 편하게 작성하게 도와주는 라이브러리
    - spring-test: 스프링 통합 테스트 지원

▼ 도메인 분석 설계

▼ 도메인 모델 및 테이블 설계

다대일 [N:1]

일대다 [1:N]

일대일 [1:1]

다대다 [N:M] (실무에서 사용 X)

▼ 엔티티 클래스 개발

- 도메인 모델 패턴

  비즈니스 로직 대부분이 엔티티에 존재

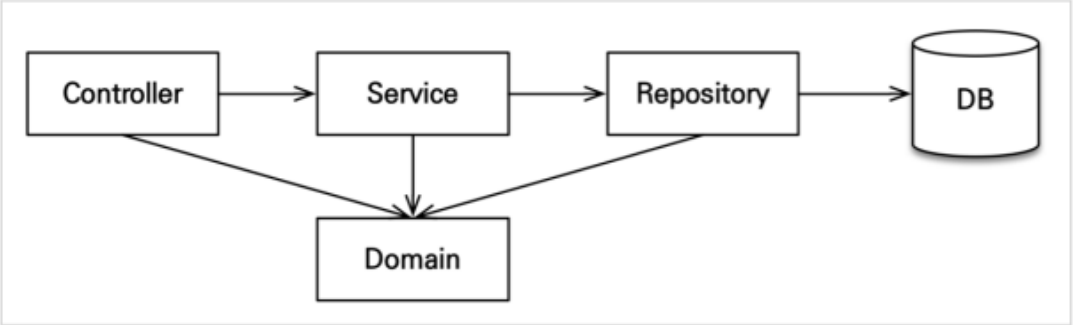
  서비스 계층은 단순히 엔티티에 필요한 요청을 위임하는 역할

- 트랜잭션 스크립트 패턴

  서비스 계층에서 대부분의 비즈니스 로직 처리

▼ 애플리케이션 구현 준비

- 구현 요구사항 (기능)
- 애플리케이션 아키텍처
  - 계층형 구조



controller, web: 웹 계층

service: 비즈니스 로직, 트랙잭션 처리

repository: JPA를 직접 사용하는 계층, 엔티티 매니저 사용

domain: 엔티티가 모여있는 계층, 모든 계층에서 사용

- 패키지 구조
- 개발 순서
  1. 개발한 엔티티 코드 확인 및 비즈니스 로직 추가
  2. 리포지토리 개발
  3. 서비스 개발
  4. 기능 테스트
  5. 웹 계층에 적용 (웹 계층 개발)

▼ 도메인 개발

(도메인: 해결하고자 하는 문제의 영역)

1. 개발한 엔티티 코드 확인 및 비즈니스 로직 추가
2. 리포지토리 개발
   - 기술

     @Repository: 스프링 빈으로 등록, JPA 예외를 스프링 기반 예외로 예외 변환

     @PersistenceContext: 엔티티 메니저(EntityManager) 주입

     @PersistenceUnit: 엔티티 메니터 팩토리(EntityManagerFactory) 주입

3. 서비스 개발

- 기술

  @Service

  @Transacional: 트랜잭션, 영속성 컨텍스트

  readOnly=true: 데이터의 변경이 없는 읽기 전용 메서드에 사용, 영속성 컨텍스트를 플러시 하지 않으므로 약간의 성능 향상

  @Autowired: 생성자

  Injection 많이 사용, 생성자가 하나면 생략 가능

  ** lombok @RequiredArgsConstructor 로 생략 가능

4. 기능 테스트

- 순서

  1. 테스트 요구사항 정의
  2. 테스트 케이스 작성
  3. 검증

- 기술

  @RunWith(SpringRunner.class): 스프링과 테스트 통합

  @SpringBootTest: 스프링 부트 띄우고 테스트(이게 없으면 @Autowired 다 실패)

  @Transactional: 반복 가능한 테스트 지원, 각각의 테스트를 실행할 때마다 트랜잭션을 시작하고 테스트가 끝나면 트랜잭션을 강제로 롤백 (이 어노테이션이 테스트 케이스에서 사용될 때만 롤백)

▼ 웹 계층 개발

등록, 조회, 수정(Dirty checking vs Merge)

Controller, Form class, html(thymleaf)