스터디 2주차(2021.08.22)

섹션 3. 회원 관리 예제 - 백엔드 개발

섹션 4. 스프링 빈과 의존관계

섹션 5. 회원 관리 예제 - 웹 MVC 개발

섹션 3. 회원 관리 예제 - 백엔드 개발

일반적인 웹 애플리케이션 계층 구조

- **컨트롤러(Controller) : 클라이언트의 요청을 받아 Requestmapping을 수행하고, 응답을 전달**
  👉 요청 url에 따라 뷰와 매핑
  👉 API 서비스를 사용하는 경우 @ResponseBody를 통해 반환 값을 HTTP response에 바로 씀
- **서비스 : 비지니스 로직을 처리 (ex : 회원가입은 중복으로 할 수 없다.)**
- **DAO(리포지토리) : 실제 DB에 접근하는 객체로, 서비스와 DB 사이의 연결 고리 역할**
  👉 도메인 객체를 DB에 저장하고 관리함

- **도메인(Entity) : 비지니스 도메인 객체 (ex: 회원, 주문, 쿠폰 등 DB에 저장되고 관리되는 객체)**
- **DTO : 계층간 데이터 교환을 위한 객체(Java Beans)**
  👉 로직을 갖고 있지 않은 순수한 데이터 객체이며 getter/setter 메서드만 가짐
  👉 DTO는 도메인 모델을 복사한 형태라고 볼 수 있음

❓ **도메인(Entity)와 DTO를 분리하는 이유?**
1. View layer와 DB layer 역할을 철저히 분리하기 위해
2. Entity가 변경되면 DB와 연관된 여러 클래스에 영향을 끼치지만,
   View와 통신하는 DTO는 자주 변경되므로 분리시켜도 영향이 적음

TDD(Test Driven Development)

👉 테스트 주도 개발이란❓ 반복 테스트를 이용한 소프트웨어 방법론으로, 작은 단위의 테스트 케이스를 작성하고 이를 통과하는 코드를 추가하는 단계를 반복하여 구현한다.

즉, 테스트를 염두해둔 프로그램 개발 방법

- 테스트 코드는 빌드 코드에 포함되지 않는다
- 테스트 코드 메소드에 given, when, then 3요소 넣는다면 직관적스럽다.

❓ **TDD의 장점**
1. 테스트 코드를 먼저 작성한다면 명확한 기능과 구조를 설계할 수 있다
✅ 한 함수에 복잡한 기능을 몰아 넣는다면 테스트는 어려워지기 때문에 재사용성을 고려하게 된다.
2. 설계 수정시간을 단축할 수 있다
✅ 테스트 코드를 먼저 작성하기에 기능을 구현하면서 최초 설계안을 토대로 구조적 문제를 찾아냄.

Optional

메서드가 반환할 결과 값이 없다는 걸 표현하고 싶을 때, 혹은 null을 반환하면 에러가 발생할 가능성이 높은 상황에서 사용하기 위해 만든 반환 타입

```
// 같은 이름이 있는 중복 회원X
Optional<Member> result = memberRepository.findByName(member.getName());
result.ifPresent(m -> {
    throw new IllegalStateException("이미 존재하는 회원입니다.");
});
```

```
memberRepository.findByName(member.getName())
        .ifPresent(m -> {
    throw new IllegalStateException("이미 존재하는 회원입니다.");
});
```

Assertion

개발자는 해당 문이 그 문의 장소에서 언제나 참이라고 간주함. 런타임 중에 거짓으로 판단되면 실패를 초래하며 이 상황에서는 일반적으로 실행이 중단된다(Assert error)

👉 디버깅을 용이하게 함

Reference

Spring 서비스 구조

https://dahye-jeong.gitbook.io/spring/spring/2020-04-12-layer