PRESENTATION

3주차 주제 TDD

~테스트 코드 작성~

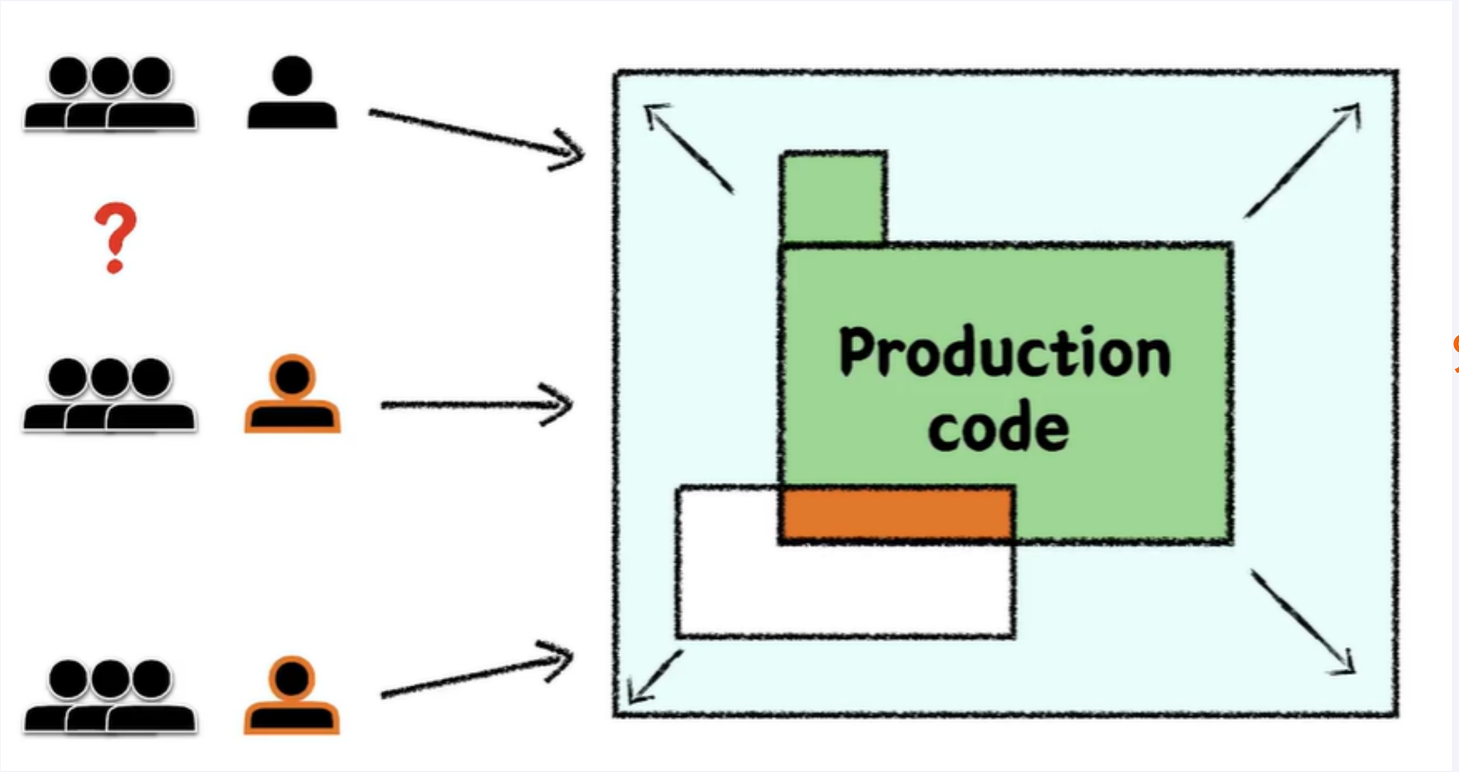
by mun

TDD란?

Test Driven Development

프로덕션 코드보다 테스트코드를 먼저 작성하여 테스트가 구현 과정을 주도하는 방법론

테스트는 왜 필요할까?



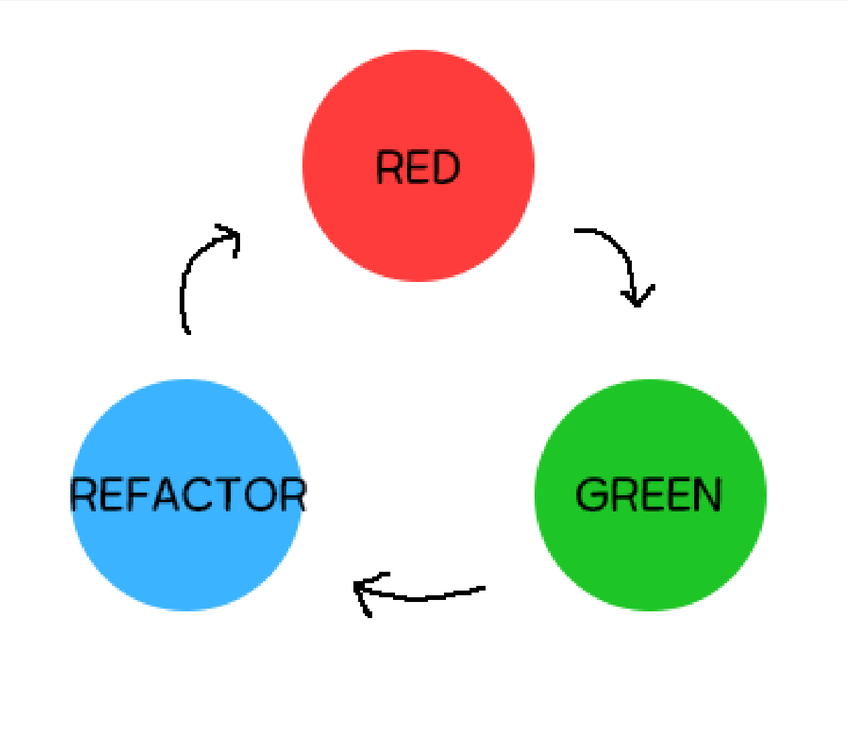
없으면...

- 또 테스트,유지보수 어려움
- 경험과 감에 의존
- 예외 케이스 검증 누락 가능성
- 커질수록 힘들어짐

있으면...

- 빠른 피드백, 자동화
- 사람->기계가 테스트 자동화
- ->안정감, 신뢰성
- 팀 차원의 이익(전체가 공유)
- 아 이기능엔 이런 케이스가 필요하구나

TDD의 기본 과정



RED

실패하는 테스트 코드 작성

GREEN

통과할 수 있는 최소한의 코딩

REFACTOR

구현 코드 개선 (리팩토링)

(초록불을 유지하면서)

테스트코드 작성해보자!

테스트 해보기

```java
@SpringBootTest
public class UserServiceTest {

    @Autowired
    private UserService userService;

    @MockBean
    private UserEntityRepository userEntityRepository;
    @MockBean
    private BCryptPasswordEncoder encoder;

    @Test
    void 회원가입이_정상적으로_동작하는_경우(){
        String userName = "userName";
        String password = "password";

        //mocking
        when(userEntityRepository.findByUserName(userName)).thenReturn(Optional.empty());
        when(encoder.encode(password)).thenReturn("encrypt_password");
        when(userEntityRepository.save(any())).thenReturn(UserEntityFixture.get(userName,password,1));

        Assertions.assertDoesNotThrow(()->userService.join(userName,password));
    }
```

@SpringBootTest

애플리케이션 전체를 로드하여 진행

Mock Object

실제 사용하는 모듈을 사용하지 않고
실제의 모듈을 흉내내는 가짜 모듈을 작성하여
테스트의 효용성을 높이는 데 사용하는 모의 객체

@Mock 이나
@MockBean 어노테이션을 통해 주입

테스트 해보기

```java
@SpringBootTest
public class UserServiceTest {

    @Autowired
    private UserService userService;

    @MockBean
    private UserEntityRepository userEntityRepository;
    @MockBean
    private BCryptPasswordEncoder encoder;

    @Test
    void 회원가입이_정상적으로_동작하는_경우(){
        String userName = "userName";
        String password = "password";

        //mocking
        when(userEntityRepository.findByUserName(userName)).thenReturn(Optional.empty());
        when(encoder.encode(password)).thenReturn("encrypt_password");
        when(userEntityRepository.save(any())).thenReturn(UserEntityFixture.get(userName,password,1));

        Assertions.assertDoesNotThrow(()->userService.join(userName,password));
    }
```

@SpringBootTest

애플리케이션 전체를 로드하여 진행

@MockBean

스프링 컨텍스트에 동일한 이름의 bean이 존재한다면 이를 MockBean 객체로 바꿔줌

테스트 해보기

```java
@SpringBootTest
public class UserServiceTest {

    @Autowired
    private UserService userService;

    @MockBean
    private UserEntityRepository userEntityRepository;
    @MockBean
    private BCryptPasswordEncoder encoder;

    @Test
    void 회원가입이_정상적으로_동작하는_경우(){
        String userName = "userName";
        String password = "password";

        //mocking
        when(userEntityRepository.findByUserName(userName)).thenReturn(Optional.empty());
        when(encoder.encode(password)).thenReturn("encrypt_password");
        when(userEntityRepository.save(any())).thenReturn(UserEntityFixture.get(userName,password,1));

        Assertions.assertDoesNotThrow(()->userService.join(userName,password));
    }
```

@SpringBootTest

애플리케이션 전체를 로드하여 진행

@MockBean

스프링 컨텍스트에 동일한 이름의 bean이 존재한다면 이를 MockBean 객체로 바꿔줌

given

when-then 메소드를 통해 대상의 given을 설정

when

given으로 정해진 상태를 테스트

then

when에서 얻은 결과를 테스트

Assertion

메소드명	설명	메소드명	설명
fail	무조건 실패	assertNotEquals	expected와 actual이 다르면 True
assertTrue	조건이 성공이면 True	assertSame	동일한 Object면 True
assertFalse	조건이 실패면 True	assertNotSame	다른 Object면 True
assertNull	조건이 Null이면 True	assertAll	여러 Assertion이 True면 True
aseertNotNull	조건이 Not Null이면 True	assertThrows	예상한 에러가 발생하면 True
assertEquals(expected, actual)	expected와 actual이 동일하면 True	assertDoesNotThrow	에러가 발생하지 않으면 True
assertArrayEquals	두 Array가 동일하면 True	assertTimeout	테스트가 지정한 시간보다 오래 걸리지 않으면 True
assertIterableEquals	두 Iterable이 동일하면 True		지정한 시간보다 오래 걸려도 테스트가 끝날 때까지 대기
assertLinesMatch	두 Stream이 동일하면 True	assertTimeoutPreemptively	테스트가 지정한 시간보다 오래 걸리지 않으면 True
			지정한 시간보다 오래 걸린 경우 바로 테스트 종료

테스트 해보기

```java
@SpringBootTest
public class UserServiceTest {

    @Autowired
    private UserService userService;

    @MockBean
    private UserEntityRepository userEntityRepository;
    @MockBean
    private BCryptPasswordEncoder encoder;

    @Test
    void 회원가입이_정상적으로_동작하는_경우(){
        String userName = "userName";
        String password = "password";

        //mocking
        when(userEntityRepository.findByUserName(userName)).thenReturn(Optional.empty());
        when(encoder.encode(password)).thenReturn("encrypt_password");
        when(userEntityRepository.save(any())).thenReturn(UserEntityFixture.get(userName,password,1));

        Assertions.assertDoesNotThrow(() ->userService.join(userName,password));
    }
```

@SpringBootTest

애플리케이션 전체를 로드하여 진행

@MockBean

스프링 컨텍스트에 동일한 이름의 bean이 존재한다면 이를 MockBean 객체로 바꿔줌

given

when-then 메소드를 통해 대상의 given을 설정

when

given으로 정해진 상태를 테스트

then

when에서 얻은 결과를 테스트

테스트 해보기 +

```
@Test
void 로그인이_정상적으로_동작하는_경우(){
    String userName = "userName";
    String password = "password";

    UserEntity fixture = UserEntityFixture.get(userName,password,1);

    //mocking
    when(userEntityRepository.findByUserName(userName)).thenReturn(Optional.of(fixture));
    when(encoder.matches(password, fixture.getPassword())).thenReturn(true);

    Assertions.assertDoesNotThrow(()->userService.login(userName,password));
}
```

1. given :
UserEntity

2. when :
userService.login

3. then :
assertDoesNotThrow

테스트 해보기 +

```
@Test
void 로그인시_userName_으로_회원가입한_유저가_없는_경우(){
    String userName = "userName";
    String password = "password";

    //mocking
    when(userEntityRepository.findByUserName(userName)).thenReturn(Optional.empty());


    Assertions.assertThrows(SnsApplicationException.class, ()->userService.login(userName,password));

}
```

1. given :
empty

2. when :
userService.login

3. then :
assertThrows

테스트를 도와주는 라이브러리

>spring-boot-starter-test

juit
테스트 프레임워크

assertj
테스트 코드를 좀 더 편하게 작성하게
도와주는 라이브러리

mockito
목 라이브러리

spring-test
스프링 통합 테스트 지원

잘 활용해서 테스트 코드를 작성하자

귀찮더라도
테스트 코드 작성은 필요!

~ 좋은 테스트를 만들어요 ~

감사합니다

참고 : 위키백과, https://www.nextree.io/mockito/, https://effortguy.tistory.com/123
https://site.mockito.org/, Practical Testing: 실용적인 테스트 가이드(박우빈)