스프링시작하기

스프링 TDD

박명회

1.1 What Is TDD?

TDD(Te st-Driven Development)는 소프트웨어 개발 방법론 중 하나로, 테스트를 먼저 작성하고 해당 테스트를 통과시키기 위한 코드를 작성하는 방식을 강조합니다. 이러한 접근 방식은 코드의 품질을 향상시키고 버그를 줄이는 데 도움이 됩니다. TDD를 사용하여 스프링 부트 애플리케이션을 개발하는 경우 다음 단계를 따를 수 있습니다.

테스트 작성 (Red 단계):
TDD의 첫 번째 단계는 테스트를 작성하는 것입니다. 테스트는 애플리케이션의 기능이나 모듈이 요구 사항을 충족하는지 검증합니다.
예를 들어, 스프링 부트 애플리케이션의 특정 컨트롤러 메소드에 대한 HTTP 요청을 시뮬레이션하는 테스트를 작성할 수 있습니다.

테스트 통과시키기 (Green 단계):
작성한 테스트를 통과시키기 위한 최소한의 코드를 작성합니다. 이 코드는 테스트를 통과할 정도의 기능만을 구현합니다.
스프링 부트 애플리케이션의 경우, 이 단계에서는 컨트롤러 메소드나 서비스 메소드 등에 대한 실제 구현을 작성합니다.

1.1 TestLife Cycle Bean

<라이브 사이클 애너테이션>

https://github.com/dron512/mhkakaoJWT/blob/main/src/test/java/org/mh/studyspringbootsecurity/example/TestLifeCycle.java

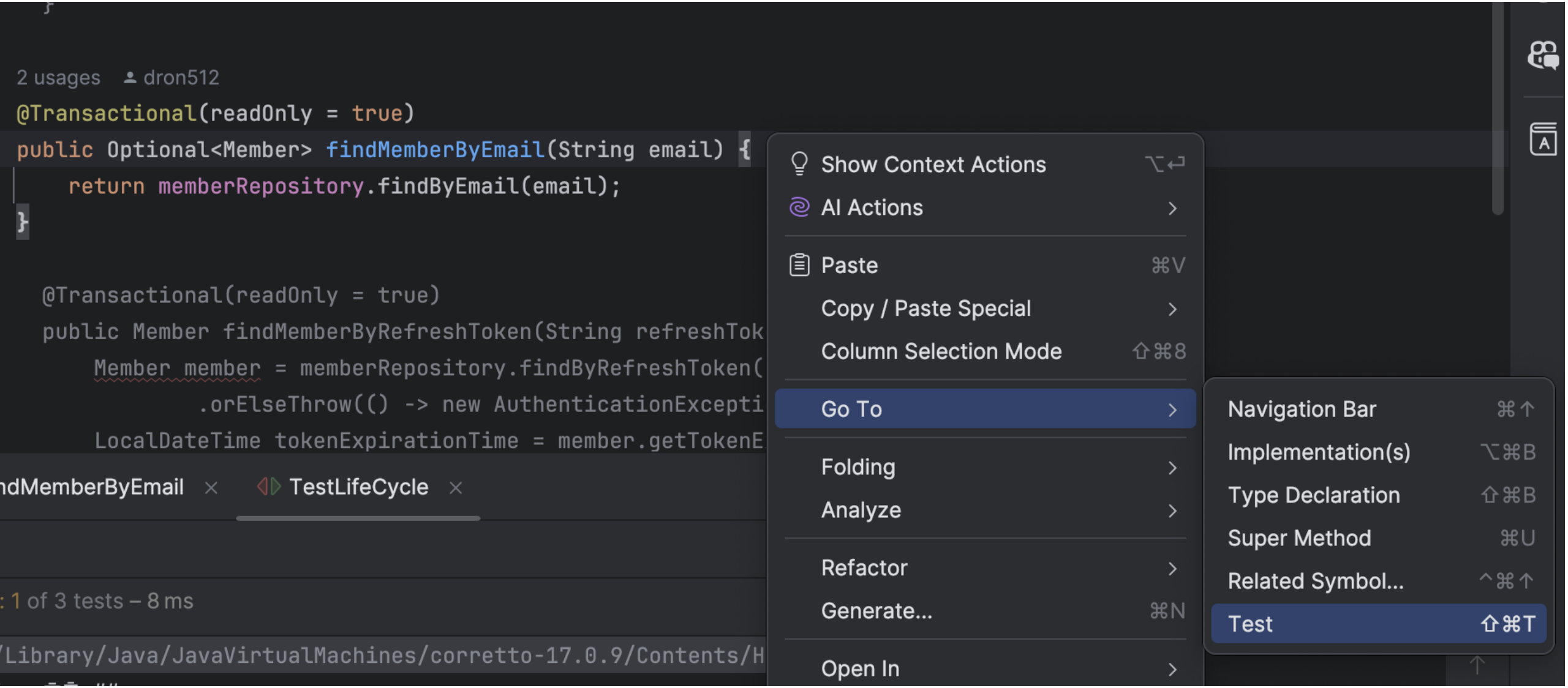
@BeforeAll
테스트 클래스 인스턴스를 초기화할 때 가장 먼저 실행됨
테스트 클래스에 포함된 테스트 메서드가 여러 개 있어도 한 번만 실행됨
객체를 생성하기 전에 미리 실행되어야 하므로 메서드는 static 접근 제어자로 정의해야 함

@BeforeEach
모든 테스트 메서드가 실행되기 전 각각 한 번씩 실행됨
테스트 클래스에 포함된 테스트 메서드가 여러 개라면 여러 번 실행됨

@AfterEach
모든 테스트 메서드가 실행되고 난 후 각각 한 번씩 실행됨
테스트 클래스에 포함된 테스트 메서드가 여러 개라면 여러 번 실행됨

@AfterAll
테스트 클래스의 모든 테스트 메서드가 실행을 마치면 마지막으로 한 번만 실행됨
테스트 메서드는 static 접근 제어자로 정의해야 함

1.1 테스트 코드 작성방법



1.2 MemberRepositoryTest

https://github.com/dron512/mhkakaoJWT/blob/main/src/test/java/org/mh/studyspringbootsecurity/domain/member/MemberRepositoryTest.java

```java
@SpringBootTest
public class MemberRepositoryTest {

    @Autowired
    MemberRepository memberRepository;
    dron512 *
    @Test
    public void saveProductTest() {
        //given
        Member optionalMember = memberRepository.save(Member.builder()
                .email("aaa@naver.com")
                .memberName("aaa")
                .memberType(MemberType.KAKAO)
                .role(Role.USER)
                .build());
        Assertions.assertEquals(optionalMember.getMemberName(), actual: "aaa");
    }
```

1.1 MemberServiceTest

https://github.com/dron512/mhkakaoJWT/blob/main/src/test/java/org/mh/studyspringbootsecurity/domain/member/MemberServiceTest.java

```java
@SpringBootTest
class MemberServiceTest {

    @Autowired
    MemberService memberService;

    @Test
    void findMemberByEmail() {
        Optional<Member> optionalMember = memberService.findMemberByEmail("aaa@naver.com");
        if(optionalMember.isEmpty())
            System.out.println("optionalMember is empty");
        else
            System.out.println(optionalMember.get().getMemberName());
    }

}
```

1.3 ControllerTest

https://github.com/dron512/mhkakaoJWT/blob/main/src/test/java/org/mh/studyspringbootsecurity/domain/member/MemberControllerTest.java

```java
dron512 *
@SpringBootTest
@AutoConfigureMockMvc
public class MemberControllerTest {
    @Autowired
    MockMvc mockMvc;
    @Autowired
    MemberRepository memberRepository;
    dron512
    @Test
    void name() throws Exception {
        //when
        ResultActions resultActions = mockMvc.perform(MockMvcRequestBuilders.get( urlTemplate: "/member/1"));

        //then
        resultActions
                .andDo(print())
                .andExpect(status().isOk());
        System.out.println("MemberRepositoryTest.name");
    }
}
```

1.3 시큐리티 테스트와 jsonPath

JsonPath 확인하기

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

```
@Test
@WithMockUser(username = "user", roles = "USER")
void member() throws Exception {
    ResultActions resultActions = mockMvc.perform(MockMvcRequestBuilders.get( urlTemplate: "/member"));
    resultActions
            .andDo(print())
            .andExpect(status().isOk())
            .andExpect(jsonPath( expression: "$[0].email").value( expectedValue: "user"))
            .andExpect(jsonPath( expression: "$[0].password").value( expectedValue: "1234"));
    ;
}
```

고맙습니다.