4장 JUnit Test

목차

1. JUnit 개요

- TDD Test Driven Development는 특정 기능을 수행하는 단위 테스트를 검증하는 방식으로 테스트 코드를 반복하여 구현하는 개발 방식
- JUnit은 TDD를 수행하기 위한 Java 테스트 프레임워크
- Spring Boot는 JUnit 어노테이션으로 애플리케이션 테스트 수행

주요 Annotation	설명
@SpringBootTest	Spring Boot 애플리케이션의 통합 테스트 정의
@Test	JUnit에서 단위 테스트 메서드를 정의
@Before	단위 테스트 메서드가 실행되기 전에 실행할 메서드를 정의
@After	단위 테스트 메서드가 실행된 후에 실행할 메서드를 정의
@MockBean	테스트 중에 가짜(Mock) 빈을 생성하고 주입하기 위해 사용되는 어노테이션
@WebMvcTest	Spring MVC 통합 테스트 정의

2. 단위 테스트

- 단위 테스트 Unit Test 는 애플리케이션의 가장 작은 단위(메서드, 클래스) 를 검증하는 테스트
- 단위 테스트는 스프링 컨텍스트 로드 없이 빠르고 독립적으로 실행해서 개발 초기에 버그를 빠르게 발견 가능
- 단위 테스트 기본 구조는 Given - When - Then 패턴

```
@Test
void addTest() {
    // Given: 테스트 준비 (객체 생성, 입력값 설정)
    Calculator calculator = new Calculator();

    // When: 실제 실행 대상 메서드 호출
    int result = calculator.add(2, 3);

    // Then: 실행 결과 검증 (예상값과 실제값 비교)
    assertEquals(5, result);
}
```

3. 통합 테스트

- 통합 테스트 Integration Test 는 애플리케이션의 여러 모듈을 함께 실행하여 전체 동작 흐름을 검증하는 테스트
- 통합 테스트는 Controller, Service, Repository 등 실제 의존성을 포함해서 실제 환경과 유사하게 비즈니스 로직 전체 테스트
- 통합 테스트는 단위 테스트만으로는 확인할 수 없는 계층 간 상호작용 검증

```
@Test
@DisplayName("사용자 저장 및 조회 통합 테스트")
void saveAndFindUserTest() {
    // Given: 사용자 엔티티 생성
    User user = new User("홍길동", "test@test.com");

    // When: DB에 저장 후 ID로 조회
          저장된 유저                          insert할 유저
    User savedUser = userRepository.save(user);
    User foundUser = userRepository.findById(savedUser.getId())
          saveUser = foundUser                          .orElseThrow();

    // Then: 조회된 값 검증
    assertEquals("홍길동", foundUser.getName());
    assertEquals("test@test.com", foundUser.getEmail());
}
```