2023년 2월 16일
JUnit과 AssertJ
보고서

JUnit과 AssertJ

임원호

-목차-

[참고 문헌]

Ⅰ. JUnit

1. JUnit 정의

'JUnit은 자바 프로그래밍 언어용 유닛 테스트 프레임워크이다. 테스트 결과는 Test클래스로 개발자에게 테스트 방법 및 클래스의 History를 공유할 수 있다. 단정(assert) 메소드로 테스트 케이스의 수행 결과를 판별하여, 테스트 케이스마다, 통과 또는 오류 검출을 출력한다.'[1] 어노테이션으로 간결하게 지원하게 되었다. 유닛 테스트란 프로그래밍에서 모든 함수와 메소드, 개별 코드에 대한 테스트 케이스(Test Case)를 작성하여, 의도된 대로 잘 동작하는지 검증하는 절차를 지닌다. 프로그램을 작은 단위로 쪼개어 각 단위가 정확하게 동작하는지 검사함으로써 프로그램의 안정성을 높이게 된다. 외부 API와의 연동이 필수라든가 DB 데이터 접근 등 외부 리소스를 직접 사용해야 하는 테스트라면 단위 테스트가 아니고, 단위 테스트에서 외부와의 연동이 필요하다면, 테스트 대역을 사용하면된다. 버퍼라이터 또는 System.out.println()등과 같은 번거로운 디버깅 없이가볍게 단위 테스트를 진행해, 실제 프로그램을 모두 완성하고 디버깅하는 시간을 단축할 수 있다.

---

[1] ehddek, "JUnit이란" https://velog.io/@ehddek/JUnit-%EC%9D%B4%EB%9E%80, (2023.02.17.최종검색).

2. JUnit은 왜 사용 할까?

‘Junit 등장 이전으로는 Main을 이용해 테스트를 진행했었고, 많은 문제가 있었다. 구현 코드와 테스트 코드가 하나의 클래스에 존재하는 점, 클래스의 크기가 커지면서 복잡도도 당연하게 증가하게 되었다.’[2] 또한 하나의 Main에서 여러 개의 기능 테스트를 하면서 복잡도는 더 증가하게 되었고, 함수 이름을 통해 어떤 부분을 테스트하는지에 대한 의도를 드러내기 힘들었다. 또한 테스트의 결과는 사람이 수동으로 일일이 확인해야 했기에.. 개발을 진행하면서, 디버깅하는 데 상당한 시간이 소요되었다. 따라서 이런 문제점을 보완하기 위해 JUnit이 등장하였고, 사용되기 시작했다.

3. JUnit의 사용

‘JUnit은 기본 Annotation을 가지게 된다.’[3]

1. @Test : 메소드를 선언하기 위해 사용한다.
2. @ParameterizedTest : @Test와 달리 @Parameterized 어노테이션은 테스트 메소드가 인자를 받을 수 있도록 한다.
3. @DisplayName : 테스트 클래스나 테스트 메소드에 이름을 붙여줄 때 사용한다.
4. @Nested : Test 클래스안에 Nested 테스트 클래스를 작성할 때 사용되며, static이 아닌 중첩 클래스, 즉 Inner클래스이어야 한다. 테스트 인스턴스 라이프 사이클이 pre-class로 설정이 되어있지 않다면, @BeforeAll , @AfterAll가 동작하지 않으니 주의해야 한다.
5. @Tag : 테스트를 필터링할 때 사용된다. 클래스 또는 메소드 레벨에 사용한다.
6. @Disabled : 테스트 클래스나, 메소드의 테스트를 비활성화한다.
7. @Timeout : 주어진 시간 안에 테스트가 끝나지 않으면 실패한다.
8. @ExtendWith : extension을 등록한다. 이 어노테이션은 상속이 된다. 확장팩이라 생각하면 될 것 같다.
9. @RegisterExtension : 필드를 통해 extension을 등록한다. 이런 필드는 private가 아니라면 상속이 된다.
10. @TempDir : 필드 주입이나 파라미터 주입을 통해 임시적인 디렉토리를 제공할 때 사용한다.

JUnit의 라이프 사이클 메소드 Annotation

1. @BeforeEach : 각각 테스트 메소드가 실행되기전에 실행되어야 하는 메소드들을 명시해 준다. Ex) @Test, @RepeatedTest, @ParameterizedTest, @TestFactory가 붙은 테스트 메소드가 실행하기전에 실행된다.
2. @BeforeAll : @BeforeEach는 각 테스트 메소드 마다 실행되지만, 이 어노테이션은 테스트가 시작하기 전 딱 한 번만 실행된다.
3. @AfterEach : @Test, @RepeatedTest, @ParameterizedTest, @TestFactory가 붙은 테스트 메소드가 실행되고 나서 한번만 실행된다.
4. @AfterAll : @Test, @RepeatedTest, @ParameterizedTest, @TestFactory가 붙은 테스트 메소드가 실행되고 나서 한번만 실행된다.

---

[2] miot2j, “단위 테스트”, https://velog.io/@miot2j/TDD-JUnit5, (2023.02.17.최종검색).
[3] miot2j, “기본 어노테이션”, https://velog.io/@miot2j/TDD-JUnit5, (2023.02.17.최종검색).

Ⅱ. AssertJ

1. AssertJ 정의

AssertJ란 단언문(assertion)을 작성하기 위한 풍부한 인터페이스를 제공하는 자바 라이브러리로, 테스트 코드의 가독성을 향상하고, 테스트 유지 관리를 더 쉽게 만드는 것을 목표로 한다. JUnit의 단언(assertThat)에 대한 표현력 부족을 보완하였고, Spring-boot-starter-test에 기본적으로 포함이 되었다.

2. AssertJ는 왜 사용 할까?

JUnit은 테스트 실행을 위한 프레임 워크를 제공하지만 한 가지 부족한 점이 있다. 그것은 단언에 대한 표현력이 부족하다는 점인데, asserTrue(id.contains("a"); 이 처럼 코드는 값이 true인지 여부를 확인하는 asserTrue()를 사용하고 있어 실제 검사흔ㄴ 내용을 표현하고 있지는 않다. 이를 AssertJ를 사용하면, 다음과 같이 바꿀 수 있다. asserThat(id).contains("a"); AssertJ를 사용하면 테스트 코드의 표현력이 높아질 뿐만 아니라 개발 도구의 자동완성 기능을 사용할 수 있다. '메소드 체이닝을 지원하여 깔끔하고 읽기 쉬운 테스트 코드를 작성이 가능하게 된다. 타입별로 다양한 검증 메소드를 제공한다.'[4]

[4] scshim, "AssertJ의 사용", https://scshim.tistory.com/422, (2023.02.17.최종검색).

[참고 문헌]

ehddek, “JUnit이란” https://velog.io/@ehddek/JUnit-%EC%9D%B4%EB%9E%80, (2023.02.17.최종검색).

miot2j, “단위 테스트”, https://velog.io/@miot2j/TDD-JUnit5, (2023.02.17.최종검색).

miot2j, “기본 어노테이션”, https://velog.io/@miot2j/TDD-JUnit5, (2023.02.17.최종검색).