개요

- JUnit Platform : 테스트를 실행해주는 런처 제공, TestEngine API 제공
- Jupiter: TestEngine API 구현체로 JUnit 5를 제공
- Vintage: JUnit4와 3을 지원하는 TestEngine 구현체

기초

- @Test
- @BeforeAll
- @AfterAll
- @BeforeEach
- @AfterEach
- @Disabled

테스트 이름 표기하기

- 일반적으로 테스트 메소드명은 camel case가 아닌 '_'로 한다.
- @DisplayNameGeneration(DisplayNameGenerator.ReplaceUnderscores.class)
  - 클래스명에 붙는 애노테이션
  - '_'를 공백으로 치환
- @DisplayName(“스터디 만들기 \uD83D\uDE31”) - 권장
  - 메소드명에 붙는 애노테이션
  - 한글, 이모지 사용 가능

Assertion

테스트 케이스	**테스트 방법**
실제 값이 기대한 값과 같은지 확인	assertEquals(expected, actual)
값이 null이 아닌지 확인	assertNotNull(actual)
다음 조건이 참(true)인지 확인	assertTure(boolean)
모든 확인 구문 확인	assertAll(executables...)
예외 발생 확인	assertThrows(expectedType, executable
특정 시간 안에 실행이 완료되는지 확인	assertTimeout(duration, executable)

대안으로 AssertJ, Hemcrest, TGruth라이브러리도 사용 가능

Assumption(조건에 따라 테스트 실행하기)

조건을 만족하면 아래의 테스트를 수행 조건을 만족하지 않으면 바로 종료(테스트를 수행하지 않으므로 성공도 실패도 아닌 Skip 상태임)

assumingThat를 사용하여 조건과 executable을 동시에 사용할 수도 있음

@EnabledOnOS, @DisabledOnOS({OS.MAC, OS.LINUE}), @EnabledOnJre({JRE.JAVA_8, ... }), @EnabledIfEnvironmentVariable(named = "", matches = "")

Test 태깅, 필터링

```
$> ./mvnw test -P ci
```

커스텀 태그

애노테이션을 생성해서 활용하자

테스트 반복하기

매개변수를 넣어서 계속 반복할 수 있다. 하고 싶은 것은 모두 가능할 것으로 판단된다.

테스트 인스턴스

기본 전략은 테스트 마다 인스턴스가 생성됨 ← 테스트간의 의존성을 없애기 위함
@TestInstance(TestInstance.Lifecycle.PER_CLASS) ← 하나의 인스턴스로 진행 @BeforeAll, @AfterAll 메소드가 static일 필요가 없음

테스트 순서

기본 전략은 각 메소드는 독립적으로 구동되어야 하므로, 순서에 크게 신경을 쓰고 있지 않음 유즈케이스 테스트를 하려면 테스트의 순서도 필요함
@TestMethodOrder(MethodOrderer.OrderAnnotation.class) @Order(1), @Order(2)... 순서가 같아도 오류는 나지 않음

Ref

- 더 자바, "코드를 테스트 하는 다양한 방법" 강의 노트
- JUNIT5 사용자 가이드(공홈)

주레피, junit5

From:
http://rwiki.repia.com/ - **문제를 잘 정의하는 것은 문제를 절반 해결한 것이다. - 2023.12**

Permanent link:
http://rwiki.repia.com/doku.php?id=wiki:java:junit:junit5&rev=1651050022

Last update: **2022/04/27 18:00**