Title

JUnit 4 ve JUnit 5 ile Unit Test

. . .

Merhaba, bu yazımızda JUnit 4 ve JUnit 5 ile Unit Test kavramını ele alacağız. Hadi başlayalım.

. . .

JUnit Framework Nedir ?

JUnit; nesne yönelimli programlama (OOP) ile tasarladığımız uygulamalarımızın, iş yapan en küçük birimleri olan methodları test etmek için kullandığımız bir frameworkdür. Bu test aşamasını daha kolay hale getirmek için anotasyonlardan yardım alırız. Anotasyonlara yazımızın devamında değineceğiz.
JUnit 5 ile birlikte JUnit Platform, JUnit Jupiter, JUnit Vintage gibi alt parçalara ayrılarak platform haline gelmiştir.
Github üzerinde açık kaynak kodları bulunmaktadır. Tıklayarak bu projeye ulaşabilir ve inceleyebilirsiniz. (JUnit 4 , JUnit 5)

JUnit 4 kullanmak için (Gradle):

```
apply plugin: 'java'

dependencies {
  testImplementation 'junit:junit:4.12'
}
```

JUnit 4 kullanmak için (Maven):

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
</dependency>
```

JUnit 5 kullanmak için (Maven):

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.7.0</version>
    <scope>test</scope>
</dependency>
```

Biraz da implementasyona göz atalım. Bu örneklerde JUnit 4 kullanacağız:

Calculator adında bir sınıfımız olsun ve içerisinde bir method barındırsın:

```
package com.batuhankiltac.junit4;

public class Calculator {

    public int multiply(int a, int b) {
        return a * b;
    }
}
```

Calculator sınıfımız için test sınıfımız şu şekilde gözükecektir:

```
package com.batuhankiltac.junit4;

import static org.junit.Assert.assertEquals;

import org.junit.Before;
import org.junit.Test;

public class CalculatorTest {

    private Calculator calculator;

    @Before
    void setUp() throws Exception {
        calculator = new Calculator();
    }

    @Test
    void testMultiply() {
        assertEquals( "Regular multiplication should work", calculator.multiply(4,5), 20);
    }

    @Test
    void testMultiplyWithZero() {
        assertEquals("Multiple with zero should be zero",0,  calculator.multiply(0,5));
        assertEquals("Multiple with zero should be zero", 0, calculator.multiply(5,0));
    }
}
```

Eğer birkaç test sınıfınız varsa, bunları bir test paketinde birleştirebilirsiniz:

```
package com.batuhankiltac.junit4;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;
import org.junit.runners.Suite.SuiteClasses;

@RunWith(Suite.class)
@SuiteClasses({MyClassTest.class, MySecondClassTest.class})

public class AllTests {
    ...
}
```

. . .

Yukarıda JUnit framework'ün 5. versiyondan sonra parçalara ayrılıp platform haline geldiğini söylemiştik. Peki bu ne demek ? Biraz detay verelim.

JUnit 5, üç farklı alt projeden ve birkaç farklı modülden oluşur:

- **JUnit Jupiter**: Public methodlarda kullanacağımız nesneyi mocklayarak testimizi yazmaya olanak sağlar.
- **JUnit Platform**: Private ve Public methodları birlikte test etmeyi sağlar. Private methodlar için PowerMockito kullanıyoruz. (JUnit4 de çalışıyor) Public methodlar için Mockito kullanıyoruz. (JUnit5 de çalışıyor)
- **JUnit Vintage**: Private methodlarda Powermockito kullanarak database ile bağlantımızı kesip mocklamamızı sağlıyor.

Anotasyonlardaki değişikleri de gösterelim:

JUNIT 4	JUNIT 5
@Test	@Test
@BeforeClass	@BeforeAll
@AfterClass	@AfterAll
@Before	@BeforeEach
@After	@AfterEach
@Ignore	@Disabled
NA	@TestFactory
NA	@Nested
@Category	@Tag
NA	@ExtendWith

. . .

Umarım bu makale Java geliştiricilerine bir nebze de olsa katkı sağlamıştır.

Başka bir yazıda görüşmek üzere.

İstek ve önerileriniz için bana batuhankiltac@gmail.com adresinden ulaşabilirsiniz.