A JUnit egységteszt keretrendszer

Jeszenszky Péter

2023.03.24.

SUnit

Az SUnit egy egységteszt keretrendszer a SmallTalk programozási nyelvhez, melyet az egységteszt keretrendszerek ősének tartanak.

A szoftvert Kent Beck írta 1994-ben.

Webhely: http://sunit.sourceforge.net/

Lásd: Martin Fowler. *XUnit*. 17 January 2006.
https://martinfowler.com/bliki/Xunit.html

xUnit (1)

Az xUnit név olyan egységteszt keretrendszerek egy családját jelenti, melyek a SmallTalk programozási nyelvhez készült SUnit egységteszt keretrendszer felépítését követik.

A JUnit volt az xUnit család első tagja, mely széles körben elterjedtté és népszerűvé vált.

Az évek során a JUnit-ot sok más programozási nyelvre portolták.

xUnit (3)

Az xUnit család más nevezetes tagjai:

- C++:
  - GoogleTest (licenc: *BSD 3-Clause License*)
    https://google.github.io/googletest/
    https://github.com/google/googletest/
- .NET:
  - NUnit (licenc: *MIT License*) https://nunit.org/
    https://github.com/nunit/nunit
- Python:
  - unittest (licenc: *Python Software Foundation License*)
    https://docs.python.org/3/library/unittest.html

xUnit (4)

Az xUnit család más nevezetes tagjai (folytatás):

- PHP:
  - PHPUnit (licenc: *BSD 3-Clause License*) https://phpunit.de/ https://github.com/sebastianbergmann/phpunit

xUnit (5)

xUnit egységteszt keretrendszerek nem objektumorientált programozási nyelvekhez:

- Haskell:
  - HUnit (licenc: *BSD 3-Clause License*)
    https://hackage.haskell.org/package/HUnit
    https://github.com/hspec/HUnit

A JUnit története

A JUnit-ot Kent Beck és Erich Gamma írta 1997-ben.

Az aktuális fő verzió az 5 számú.

Rendelkezésre állás

- Licenc: *Eclipse Public License 2.0*
- Webhely: https://junit.org/junit5/
- Tároló: https://github.com/junit-team/junit5
- Maven Central: a JUnit termékek rendelkezésre állnak a központi tárolóban
  - A JUnit termékek listája: https://junit.org/junit5/docs/current/user-guide/#dependency-metadata
- Dokumentáció:
  - *JUnit 5 User Guide* https://junit.org/junit5/docs/current/user-guide/
  - Javadoc: https://junit.org/junit5/docs/current/api/

Architektúra (1)

A JUnit 5 moduláris felépítésű:

- JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage

Architektúra (2)

- **JUnit Platform**: Alapként szolgál teszt keretrendszerek a JVM-en történő elindításához. Meghatároz Egy `TestEngine` API-t a platformon futó teszt keretrendszerek fejlesztéséhez. Biztosít egy `ConsoleLauncher`-t a platform a parancssorból történő elindításához. A népszerű IDE-k (például IntelliJ IDEA, Eclipse, NetBeans) és fordítás automatizáló eszközök (például Apache Maven és Gradle) támogatják a JUnit Platformot.
- **JUnit Jupiter**: Egy programozási modellt és egy kiterjesztési modellt definiál tesztek és kiterjesztések a JUnit 5-ben történő írásához. Biztosít egy `TestEngine`-t is Jupiter-alapú tesztek a platformon történő futtatásához.
- **JUnit Vintage**: egy `TestEngine`-t biztosít JUnit 3 és JUnit 4 alapú tesztek a platformon történő futtatásához.

IDE támogatás

- Apache NetBeans: *Writing JUnit Tests in NetBeans IDE* http://netbeans.apache.org/kb/docs/java/junit-intro.html
- Eclipse IDE: *Writing and running JUnit tests* https://help.eclipse.org/latest/topic/org.eclipse.jdt.doc.user/gettingStarted/qs-junit.htm
- IntelliJ IDEA: https://www.jetbrains.com/help/idea/junit.html

Apache Maven támogatás (1)

Apache Maven Surefire https://maven.apache.org/surefire/
https://github.com/apache/maven-surefire

- Maven Surefire Plugin
  https://maven.apache.org/surefire/maven-surefire-plugin/
- Maven Surefire Report Plugin
  https://maven.apache.org/surefire/maven-surefire-report-plugin/

Apache Maven támogatás (2)

A `default` életciklus az egységteszteléshez kötődő fázisai és bővítmény kötéseik:

Fázis	Bővítmény cél
`generate-test-sources`	
`process-test-sources`	
`generate-test-resources`	
`process-test-resources`	`resources:testResources`
`test-compile`	`compiler:testCompile`
`process-test-classes`	
`test`	`surefire:test`

Használat módja az Apache Maven-nel (1)

Adjuk hozzá az `org.junit.jupiter:junit-jupiter-engine` terméket függőségként a `pom.xml`-hez:

```
<dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>${junit.jupiter.version}</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Használat módja az Apache Maven-nel (2)

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>${surefire.version}</version>
    </plugin>
  </plugins>
</build>
<reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-report-plugin</artifactId>
      <version>${surefire.version}</version>
    </plugin>
  </plugins>
</reporting>
```

Gradle támogatás

Lásd: *Testing in Java & JVM projects*
https://docs.gradle.org/current/userguide/java_testing.html

Tesztosztályok és -metódusok (1)

Tesztosztály: bármely felsőszintű osztály, statikus tagosztály vagy `@Nested` osztály, mely legalább egy tesztmetódust tartalmaz.

- Nem lehet absztrakt és egyetlen konstruktora kell, hogy legyen.

Tesztmetódus: a `@Test`, `@RepeatedTest`, `@ParameterizedTest`, `@TestFactory` vagy `@TestTemplate` annotációval megjelölt bármely példánymetódus.

Életciklus metódus: a `@BeforeAll`, `@AfterAll`, `@BeforeEach` vagy `@AfterEach` annotációval megjelölt bármely metódus.

- A `@BeforeAll` és `@AfterAll` annotációkkal jelölt metódusok statikusak kell, hogy legyenek (kivéve azt az esetet, amikor az `@TestInstance(Lifecycle.PER_CLASS)` annotációt használjuk).

Tesztosztályok és -metódusok (2)

Nem szükséges, hogy a tesztosztályok, tesztmetódusok és életciklus metódusok nyilvánosak legyenek, de nem lehetnek privát láthatóságúak.

Tesztmetódusok és életciklus metódusok:

- Deklarálhatók az aktuális tesztosztályon belül lokálisan, örökölhetők ősosztályból vagy interfészektől.
- Nem lehetnek absztraktak és nem adhatnak vissza értéket.

A tesztosztály konstruktoroknak és metódusok is meg van engedve, hogy paramétereik legyenek, mely lehetővé teszi a függőség befecskendezést.

Teszt végrehajtási életciklus (1)

Alapértelmezésben a JUnit egy új példányt hoz létre minden egyes tesztosztályból az egyes tesztmetódusok végrehajtás előtt, mely lehetővé teszi a tesztmetódusok izoláltan történő végrehajtását.

Ez a viselkedés megváltoztatható, az összes tesztmetódus ugyanazon a tesztpéldányon történő végrehajtásához a tesztosztályt a `@TestInstance(Lifecycle.PER_CLASS)` annotációval kell megjelölni.

Teszt végrehajtási életciklus (2)

Példa:

```
import org.junit.jupiter.api.*;

public class LifeCycleTest {

  LifeCycleTest() {
    System.out.printf("Constructor creates %s\n", this);
  }

  @BeforeAll
  static void beforeAll() { System.out.println("@BeforeAll static method invoked"); }

  @AfterAll
  static void afterAll() { System.out.println("@AfterAll static method invoked"); }

  @BeforeEach
  void beforeEach() { System.out.printf("@BeforeEach method invoked on %s\n", this); }

  @AfterEach
  void afterEach() { System.out.printf("@AfterEach method invoked on %s\n", this); }

  @Test
  void testMethod1() { System.out.printf("testMethod1() method invoked on %s\n", this); }

  @Test
  void testMethod2() { System.out.printf("testMethod2() method invoked on %s\n", this); }

}
```

Teszt végrehajtási életciklus (3)

Példa:

```
@BeforeAll static method invoked
Constructor creates LifeCycleTest@2145433b
  @BeforeEach method invoked on LifeCycleTest@2145433b
  testMethod1() method invoked on LifeCycleTest@2145433b
  @AfterEach method invoked on LifeCycleTest@2145433b
Constructor creates LifeCycleTest@fdefd3f
  @BeforeEach method invoked on LifeCycleTest@fdefd3f
  testMethod2() method invoked on LifeCycleTest@fdefd3f
  @AfterEach method invoked on LifeCycleTest@fdefd3f
@AfterAll static method invoked
```

Teszt végrehajtási sorrend

Alapértelmezésben a tesztmetódusok rendezése egy determinisztikus, de szándékosan nem nyilvánvaló algoritmussal történik. Ez biztosítja azt, hogy egy tesztkészlet egymást követő futtatásaikor a tesztmetódusok ugyanabban a sorrendben kerülnek végrehajtásra.

Noha a valódi egységteszteknek jellemzően nem szabad a végrehajtási sorrendtől függeniük, vannak esetek, amikor szükséges a tesztmetódusok egy bizonyos végrehajtási sorrendjének kikényszerítése.

A `@TestMethodOrder` annotáció szolgál a tesztmetódusok végrehajtási sorrendjének vezérlésére.

Kijelentések

Az org.junit.jupiter.api.Assertions osztály olyan segédmetódusok gyűjteménye, melyek feltételek teljesülésének kijelentésére szolgálnak tesztekben.

- Minden metódus statikus.

Teszteredmények

Lásd: http://xunitpatterns.com/Glossary.html

- **Siker** (***success***): amikor a teszt végrehajtásakor minden tényleges eredmény megegyezik a várt végeredménnyekkel.
  - Ekkor azt mondjuk, hogy a teszt átmegy (*passes*).
- **Bukás** (***failure***): amikor a teszt végrehajtásakor a tényleges eredmény nem egyezik meg a várt végeredménnyel.
  - A bukást egy elbukó állítás okozza.
  - Ekkor azt mondjuk, hogy a teszt megbukik (*fails*).
- **Hiba** (***error***): amikor a teszt végrehajtásakor egy hiba következik be, mely megakadályozza a befejeződést.
  - A hibát egy váratlan kivétel vagy hiba okozza.

Kapcsolódó könyvtárak és keretrendszerek (1)

Kijelentés/*matcher* könyvtárak:

- AssertJ (licenc: *Apache License 2.0*) https://assertj.github.io/doc/ https://github.com/assertj/assertj-core
- Hamcrest http://hamcrest.org/
  - Java Hamcrest (licenc: *BSD 3-Clause License*) http://hamcrest.org/JavaHamcrest/ https://github.com/hamcrest/JavaHamcrest
  - PyHamcrest (licenc: *BSD 3-Clause License*) https://github.com/hamcrest/PyHamcrest
  - ...
- Truth (licenc: *Apache License 2.0*) https://truth.dev/ https://github.com/google/truth/

Kapcsolódó könyvtárak és keretrendszerek (2)

Mock keretrendszerek:

- EasyMock (licenc: *Apache License 2.0*) https://easymock.org/ https://github.com/easymock/easymock
- Mockito (licenc: *MIT License*) https://site.mockito.org/ https://github.com/mockito/mockito
- PowerMock (licenc: *Apache License 2.0*) https://github.com/powermock/powermock

Kapcsolódó könyvtárak és keretrendszerek (3)

Adatbázisok:

- DbUnit (licenc: GNU LGPLv2.1) http://dbunit.sourceforge.net/

Fejnélküli böngészők:

- HtmlUnit (licenc: *Apache License 2.0*) https://htmlunit.sourceforge.io/ https://github.com/HtmlUnit/htmlunit

A `System.out` elkapása a System Lambda könyvtárral (1)

A System Lambda könyvtár a `java.lang.System` osztályt használó kód teszteléséhez ad támogatást.

- Licenc: *MIT License*
- Tároló: https://github.com/stefanbirkner/system-lambda
- Rendelkezésre állás a központi Maven tárolóban: `com.github.stefanbirkner:system-lambda:1.2.1`

A System.out elkapása a System Lambda könyvtárral (2)

Példa:

```
@Test
void testMain() throws Exception {
  String output = tapSystemOut(() -> HelloWorld.main(
      new String[] {}));
  assertEquals("Hello, World!\n", output);
}
```

A `System.out` elkapása a System Lambda könyvtárral (3)

A `Normalized` utótagú metódusok a sortöréseket `\n`-re normalizálják úgy, hogy ugyanazokkal a kijelentésekkel futtathatók a tesztek különböző operációs rendszereken.

Példa:

```
@Test
void testMain() throws Exception {
  String output = tapSystemOutNormalized(() ->
      HelloWorld.main(new String[] {}));
  assertEquals("Hello, World!\n", output);
}
```

Ipari példák: Apache Commons Lang (1)

`org.apache.commons.lang3.StringUtils`:

- StringUtils.java
- Javadoc
- JUnit 5 egységtesztek:
  - `StringUtilsContainsTest.java`
  - `StringUtilsSubstringTest.java`
  - `StringUtilsTest.java`

Ipari példák: Apache Commons Lang (2)

`org.apache.commons.lang3.BitField`:

- BitField.java
- Javadoc
- BitFieldTest.java

Ipari példák: Guava

`com.google.common.collect.Multisets`:

- Multisets.java
- Javadoc
- MultisetsTest.java (JUnit 4)

További ajánlott irodalom

- Cătălin Tudose. *JUnit in Action*. 3rd ed. Manning Publications, 2021. https://github.com/ctudose/junit-in-action-third-edition
- Boni Garcia. *Mastering Software Testing with JUnit 5*. Packt Publishing, 2017. https://github.com/PacktPublishing/Mastering-Software-Testing-with-JUnit-5