Szűgyi Zalán

Unit tesztek Java és C++ környezetben

Tartalom

- Tesztelésről: unit tesztek – teszt esetek
- JUnit
  - Konfigurálás
  - Keretrendszer
  - API
- gtest
  - Konfigurálás
  - Keretrendszer
  - API

Tesztelés

- Programok tesztelése (ISO/IEC TR 19759:2005)
  - Unit test
  - Integration test
  - System test
  - Acceptence test

Unit test

- Egy modult önmagában tesztel

- Segít minél előbb megtalálni a hibákat
- Segít a refaktorálásban
- Öndokumentálás
- Interfészek elválasztása az implementációtól

JUnit

- Unit teszt keretrendszer Java környezethez
- Open source
- Min. rendszer követelmények: JDK 1.5+
- Integrálható:
  - Eclipse
  - Maven
  - Ant

JUnit konfigurálás

- letölthető: http://junit.org

```
export PATH=$PATH:<java install path>/bin/

export CLASSPATH=$CLASSPATH:\
<junit path>/junit4.10.jar:.
```

- A 4.11-es verzióhoz a **hamcrest-core.jar** is szükséges

Példa teszteset
(TestJunit.java)

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestJunit {
   @Test
   public void testAdd() {
      String str =
         "Junit is working fine";
      assertEquals(
         "Junit is working fine",
          str);
   }
}
```

Példa tesztkörnyezet
(TestRunner.java)

```
import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner {
  public static void main(String[] args) {

    Result result =
      JUnitCore.runClasses(TestJunit.class);

    for (Failure failure: result.getFailures()){
      System.out.println(failure.toString());
    }
    System.out.println(result.wasSuccessful());
  }
}
```

Fordítás, futtatás

- Fordítás:

```
javac TestJunit.java TestRunner.java
```

- Futtatás:

```
java TestRunner
```

- Kimenet:

```
true
```

A tesztelés lépései

1. Adott egy osztály, amit tesztelni szeretnénk
2. Készítsünk egy Test Case osztályt
   - A teszt metódusokat annotáljuk @Test annotációval
   - A tesztfeltételeket ellenőrízzük az assert API segítségével.

A tesztelés lépései (folyt.)

3. A teszteket futtató osztály elkészítése
   - JUnitCore.runClasses(...)
   - Result objektum
   - getFailures()
   - wasSuccessful()

Assert API

- assertEquals
- assertArrayEquals
- assertFalse
- assertTrue
- assertNull
- assertNotNull
- assertSame
- assertNotSame
- assertThat
- fail

http://junit.sourceforge.net/javadoc/

Egyszerű példa

- MiniMath.java osztály
  - factorial
- TestMiniMath.java tesztosztály
- TestRunner.java futtató környezet

`forrás: java/minimath`

Több teszteset

- Gyakori, hogy egy osztályt sok tesztes tesztel
  - Közös erőforrások
  - közös inicializálás
- Annotációk:
  - **`Before`**
  - **`BeforeClass`**
  - **`Ignore`**
  - **`After`**
  - **`AfterClass`**

Példa több tesztesetre

- MiniMath.java osztály
  - factorial
  - gcd
- TestMiniMath.java tesztosztály
  - Annotációk
- TestRunner.java futtató környezet

`forrás: java/minimath2`

Test Suite

- Több összetartozó teszt-osztályokban definiált tesztesetek futtatása
- **@RunWith** és **@Suite** annotációk

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

@RunWith(Suite.class)
@Suite.SuiteClasses({
   TestJunit1.class,
   TestJunit2.class
})
public class JunitTestSuite {}
```

Példa Test Suite-re

- MiniMath.java osztály
  - factorial
  - gcd
- TestMiniMath1.java és TestMiniMath2.java tesztosztályok
  - Annotációk
- TestSuiteMath.java
- TestRunner.java futtató környezet

`forrás: java/minimath3`

Timeout a tesztekben

- Megadhatjuk, hogy maximum meddig futhasson az adott teszteset
  - millisec
  - időtúllépés esetén --> fail
  - **`@Test(timeout=1000)`**

Példa timeout-ra

- MiniMath.java osztály
  - factorial
  - gcd
  - fibonacci
- TestMiniMath.java tesztosztály
  - **`@Test(timeout=1000)`**
- TestRunner.java futtató környezet

`forrás: java/minimath4`

Kivétel dobás tesztelése

- Azt szeretnénk ellenőrízni, hogy dobott-e kivételt az adott metódus
- **`@Test(expected = <givenException>.class)`**

Példa kivételdobás tesztelésére

- MiniMath.java osztály
  - factorial
    - negatív számra ArithmeticException-t dob
  - gcd
  - fibonacci
- TestMiniMath.java tesztosztály
  - **`@Test(expected = ArithmeticException.class)`**
- TestRunner.java futtató környezet

`forrás: java/minimath5`

Mock

- Unit teszt egy osztályt önállóan tesztel
- Valóságban egy osztály felhasználhatja több osztály szolgáltatásait
- Mock-olás:
  - Ezen osztályok tagfüggvényeinek hívását lecseréljük
  - Dummy értékeket adunk vissza
  - Az osztály tesztelése nem függ a többi osztálytól (pl. nem kell hozzá adatbázis kapcsolat)

- Mock keretrendszer Java környezethez
- Egyszerű API
  - Verifikáció
  - Stub-ok
- Polimorfizmust használ, statikus, final metódusokat nem tudunk mock-olni vele
- Junit támogatás

Telepítés, beállítások

- Letölthető:
  http://code.google.com/p/mockito/downloads/list
- Adjuk hozzá a CLASSPATH-hoz a mockito-all-1.9.5.jar-t

Egyszerű verifikáció

```
import static org.mockito.Mockito.*;

//mock creation
List mockedList = mock(List.class);

//using mock object
mockedList.add("one");
mockedList.clear();

//verification
verify(mockedList).add("one");
verify(mockedList).clear();
```

Stub-olás

```
//A konkrét osztályt kell mock-olni nem elég az interfészt
LinkedList mockedList = mock(LinkedList.class);

//stub
when(mockedList.get(0)).thenReturn("first");
when(mockedList.get(1)).thenThrow(new
RuntimeException());

// kiírja hogy first
System.out.println(mockedList.get(0));
// runtime exception
System.out.println(mockedList.get(1));

// null, mert a 999 nincs stub-olva
System.out.println(mockedList.get(999));
```

Példa

- Program:
  - Person: egy dolgozót ír le
  - PersonPersister: dolgozót ment el és tölt be egy adatbázisból
  - Factory: gyár osztály, itt dolgoznak a dolgozók
- TestFactory:
  - Factory unit tesztje (nem akarunk adatbázist kezelni a tesztben)
  - Mock-olja a PersonPersister-t
- TestRunner: futtató környezet

`forrás: java/mocktest1`

Argumentum matcherek

- Ha nem konkrét értékekkel hanem általánosabban akarunk megadni egy argumentumok egy fvhívás mock-olásakor:

```
when(mockedList.get(anyInt())).
     thenReturn("element");
```

Matcherek

- anyBoolean, anyInt, anyDouble, ...
- anyCollection, anyList, anyMap, ...
- contain, startswith, endswith, matches
- isNull, isNotNull

http://docs.mockito.googlecode.com/hg/latest/org/mockito/Matchers.html

Függvényhívások számának ellenőrzése

```
mocledList.add("once");


mockedList.add("twice");
mockedList.add("twice");


mockedList.add("three times");
mockedList.add("three times");
mockedList.add("three times");
```

Függvényhívások számának ellenőrzése folyt.

```
//A következő két sor ekvivalens (a times(1) a default)
verify(mockedList).add("once");
verify(mockedList, times(1)).add("once");

//konkrét fvhívásszám ellenőrzése
verify(mockedList, times(2)).add("twice");
verify(mockedList, times(3)).add("three times");

//never() egy alias a times(0)-ra
verify(mockedList, never()).add("never happened");

//atLeast()/atMost()
verify(mockedList, atLeastOnce()).add("three times");
verify(mockedList, atLeast(2)).add("five times");
verify(mockedList, atMost(5)).add("three times");
```

Példa

```
verify(mockedPersonPersister, times(2)).
   load(anyString());
```

vs

```
verify(mockedPersonPersister).load("Jozsi");
verify(mockedPersonPersister).load("Dezso");
```

forrás: java/mocktest2

Redundáns hívások ellenőrzése

```
mockedList.add("one");
mockedList.add("two");

//egy add fv-hívást elvár
verify(mockedList).add("one");

//minden továbbit megtilt
verifyNoMoreInteractions(mockedList);
```

gtest – gmock

Konfigurálás

- Érdemes rögtön a gmock-ot letölteni, mert az tartalmazza a gtest-et is.
- Kitömörítés után:

```
g++ -I${GTEST_DIR}/include -I${GTEST_DIR} \
    -I${GMOCK_DIR}/include -I${GMOCK_DIR} \
    -c ${GTEST_DIR}/src/gtest-all.cc

g++ -I${GTEST_DIR}/include -I${GTEST_DIR} \
    -I${GMOCK_DIR}/include -I${GMOCK_DIR} \
    -c ${GMOCK_DIR}/src/gmock-all.cc

ar -rv libgmock.a gtest-all.o gmock-all.o
```

Teszteset fordításához

- Fordításhoz megadni:
  - **-I${GTEST_DIR}/include**
  - **-I${GMOCK_DIR}/include**
  - **-L${GMOCK_DIR}/lib**
  - **-lgmock**
  - **-lpthread**
  - **testfile.cpp**

Példa

- JUnit-nál megismert első példa C++-os környezetben:
  - minimath.cpp
  - minimath.h
  - test.cpp

`forrás: cpp/minimath1`

ASSERT API (fatal)

- ASSERT_TRUE
- ASSERT_FALSE
- ASSERT_EQ
- ASSERT_NE
- ASSERT_LT
- ASSERT_GT
- ASSERT_GE
- ASSERT_STREQ
- ASSERT_STRNE
- ASSERT_STRCASEEQ
- ASSERT_STRCASENE

EXPECT API (non fatal)

- EXPECT_TRUE
- EXPECT_FALSE
- EXPECT_EQ
- EXPECT_NE
- EXPECT_LT
- EXPECT_GT
- EXPECT_GE
- EXPECT_STREQ
- EXPECT_STRNE
- EXPECT_STRCASEEQ
- EXPECT_STRCASENE

Fixture-ök

- Ha több tesztesetet közösen szeretnénk inicializálni, közös erőforrásaik vannak...
- Annotációk helyett fixture osztály:
  - ::testing::Test-ből származik
  - adattagjai elérhetőek a tesztekből
  - TEST helyett TEST_F makró
    - első paramétere kötelezően az osztály neve

Fixture-ök folyt.

- Virtuális tagfüggvények:
  - SetUp: minden teszteset előtt lefut (mint a @Before annotáció JUnitban)
  - TearDown: minden testeset után lefut (mint az @After annotáció JUnitban)

`forrás: cpp/minimath2`

Mocking

- Virtuális fv-eken, vagy templateken
- Expectation
  - EXPECT_CALL
  - Argument matcher-ek (konkrét érték vagy _, `using ::testing::_`
  - Kardinalitás (Times(n))
  - Akciók

Példa

```
int n = 100;
EXPECT_CALL(Point, GetX())
.Times(4)
.WillRepeatedly(Return(n++));
```

Példa

- JMock példa gmock-al

`forrás: cpp/mocktest1`

Linkek – JUnit

- http://www.tutorialspoint.com/junit/
- http://junit.org
- http://junit.sourceforge.net/javadoc/
- http://code.google.com/p/mockito/
- http://docs.mockito.googlecode.com/hg/latest/org/mockito/

Linkek gtest

- http://code.google.com/p/googletest/
- http://code.google.com/p/googlemock/