EQUILIBRIUM

JDD 2006 Java Developers Day

PISANIE TESTÓW Z WYKORZYSTANIEM BIBLIOTEKI TESTNG

Janusz Marchewa

Kraków, 21.10.2006 r.

EQUILIBRIUM

Plan prelekcji

- Dlaczego TestNG?
- Porównanie z JUnit 3.8.1 i JUnit 4
- Dodatkowe zalety TestNG
- Migracja istniejących testów do TestNG

EQUILIBRIUM

Dlaczego TestNG?

- JUnit był przez lata bezkonkurencyjny, ale:
  - zarządzanie złożonym zestawem testów było czasochłonne i często irytujące
  - czas ujawnił pewne wady projektowe tej biblioteki
  - biblioteka nie była przez długi czas rozwijana, co pogłębiło frustrację części użytkowników
- Potrzeba stworzenia alternatywy
- Narodziny **Test**ing, the **N**ext **G**eneration

Główny autor – Cedric Beust

- Dawniej Sun Microsystems, BEA Systems
- Obecnie pracownik Google
- Uczestniczył w pracach nad różnymi JSR, m.in. JSR-175 (adnotacje) i JSR-220 (EJB3)
- Współautor *Professional Java Server Programming J2EE, 1.3 Edition*

EQUILIBRIUM

Sfrustrowany Cedric...

- Powody swoich frustracji biblioteką JUnit zaczął podawać na blogu już w 2003 roku
- Niektóre z nich:
  - nowy obiekt każdej klasy dziedziczącej z junit.framework.TestCase jest tworzony tyle razy, ile metod testujących posiada ta klasa
  - brak możliwości zmiany listy uruchamianych testów bez przekompilowania kodu
  - brak możliwości zdefiniowania zależności pomiędzy testami

EQUILIBRIUM

...staje się coraz bardziej sfrustrowany...

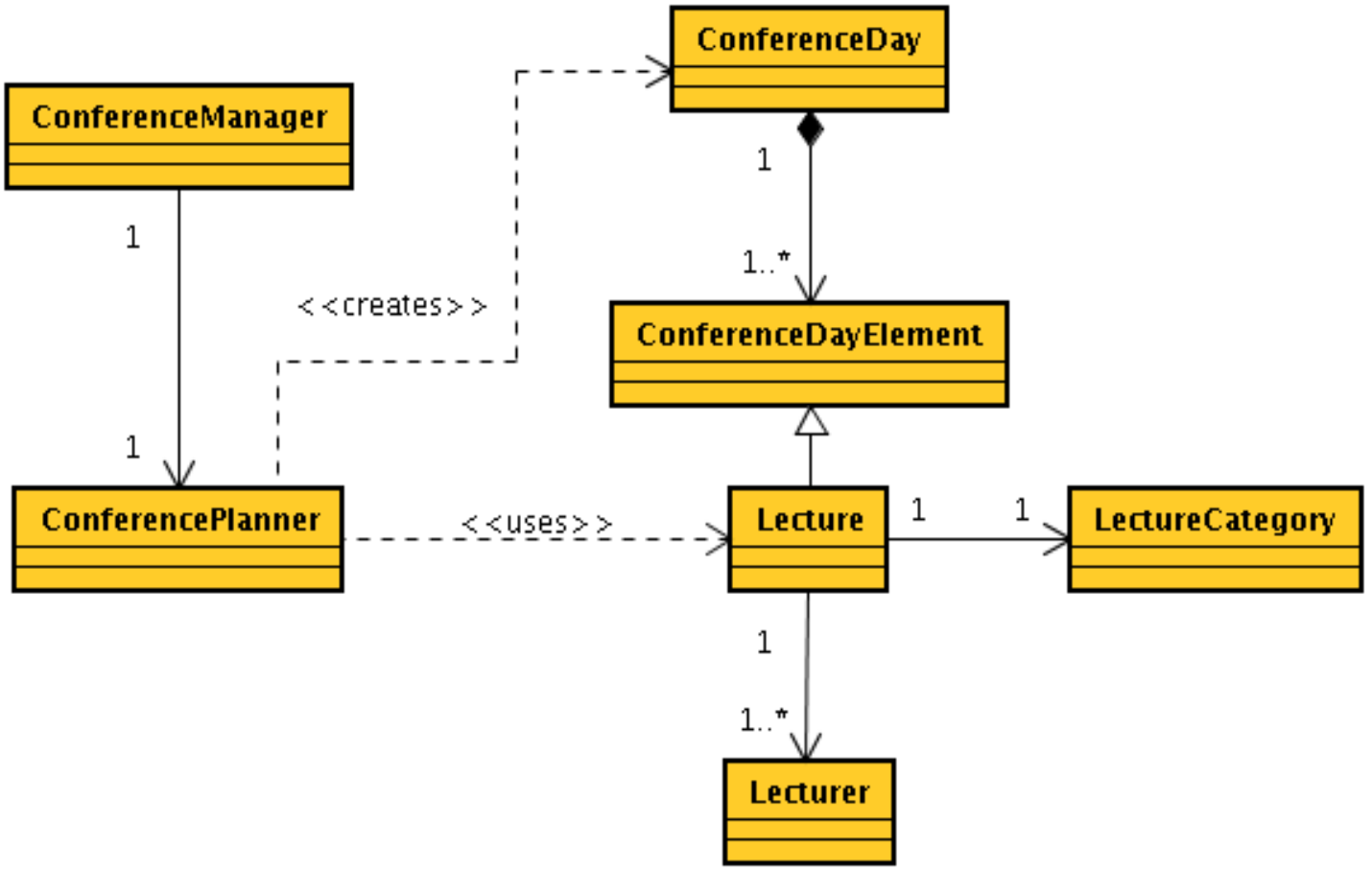
- I podaje kolejne wady JUnit:
  - brak prostego sposobu na tymczasowe wyłączenie niektórych testów
  - brak niezależnego systemu logowania informacji o wykonywanych testach
  - mało intuicyjne listy parametrów metod dokonujących asercji

EQUILIBRIUM

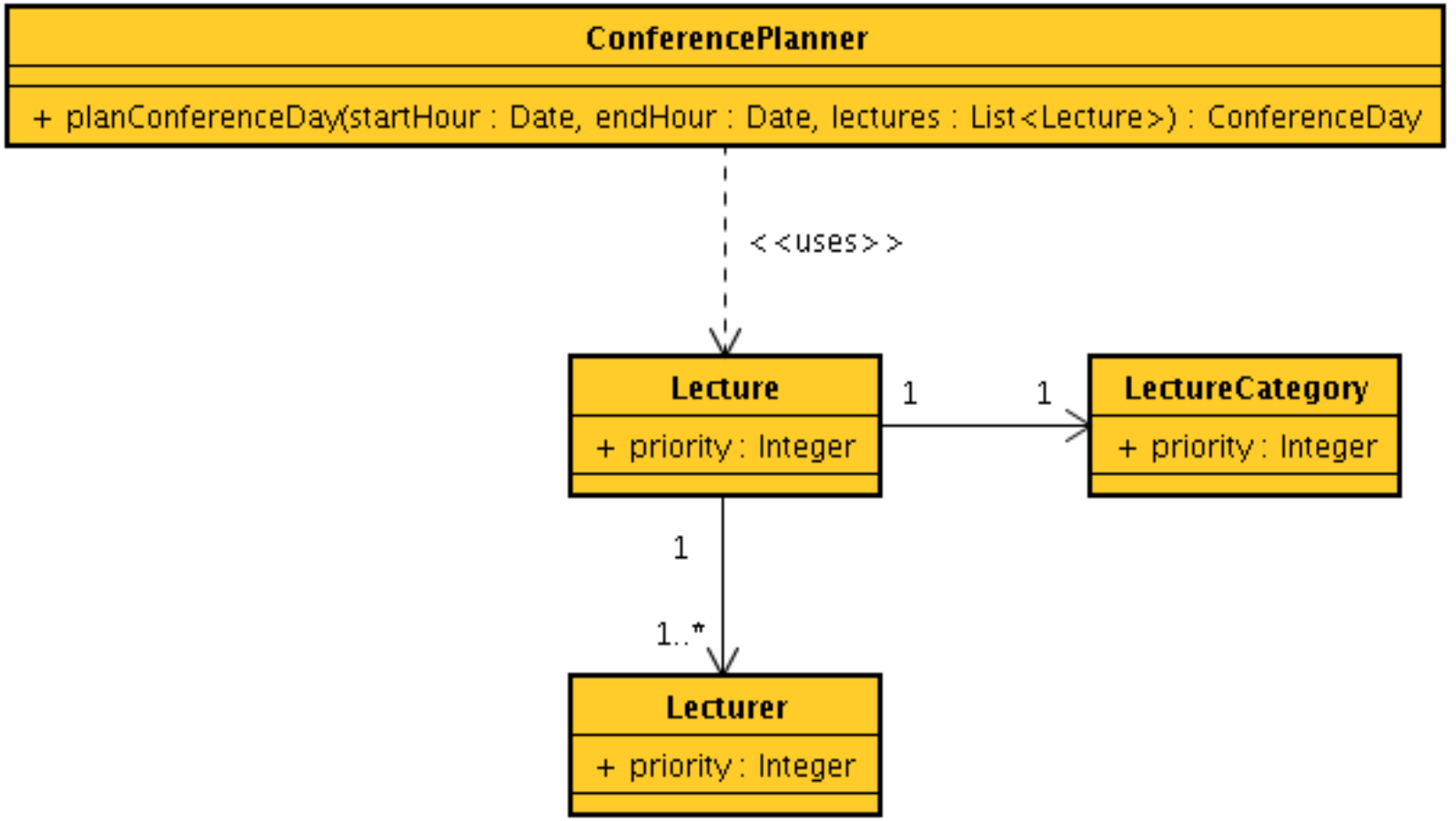
...aż decyduje się napisać TestNG

- Nowa biblioteka miała czerpać najlepsze rozwiązania z JUnit i NUnit
- Od samego początku miała wykorzystywać adnotacje (JSR-175)
- Pierwsza wersja w kwietniu 2004 roku
- Wrzesień 2006: wersja 5.2
- Jak wypada na tle JUnit?

Przykłady: planowanie konferencji



Przykłady: planowanie konferencji



EQUILIBRIUM

Prosty przykład (JUnit 3.8.1)

```
import junit.framework.TestCase;

public class LectureComparatorTest extends TestCase {
    private LectureComparator lectureComparator = new LectureComparator();

    public void testCompareLectureToItself() {
        Lecture lecture = new Lecture();

        int comparisonResult = lectureComparator.compare( lecture, lecture );

        assertEquals( 0, comparisonResult );
    }
}
```

Prosty przykład (TestNG)

```
import org.testng.AssertJUnit;
import org.testng.annotations.Test;

// obsługa adnotacji wymaga pliku testng-x.y-jdk15.jar
public class LectureComparatorTest {
    private LectureComparator lectureComparator = new LectureComparator();

    @Test
    public void compareLectureToItself() {
        Lecture lecture = new Lecture();

        int comparisonResult = lectureComparator.compare( lecture, lecture );

        // może być również: assert comparisonResult == 0;
        AssertJUnit.assertEquals( 0, comparisonResult );
    }
}
```

Wykorzystanie JDK 1.4 (TestNG)

```
import org.testng.AssertJUnit;

// obsługa komentarzy JavaDoc wymaga biblioteki testng-x.y-jdk14.jar
public class LectureComparatorTest {
    private LectureComparator lectureComparator = new LectureComparator();

    /**
     * @testng.test
     */
    public void compareLectureToItself() {
        Lecture lecture = new Lecture();

        int comparisonResult = lectureComparator.compare( lecture, lecture );

        // może być również: assert comparisonResult == 0;
        AssertJUnit.assertEquals( 0, comparisonResult );
    }
}
```

Konfiguracja TestNG - *testng.xml*

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite verbose="2" name="JDD2006 Suite">
  <test name="LectureComparator Test" annotations="JDK5">
    <classes>
      <class name="equilibrium.jdd2006.testng.LectureComparatorTest">
        <methods>
          <include name="compareLectureToItself"/>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

Uruchamianie TestNG

- z poziomu konsoli

```
$ java org.testng.TestNG sciezka-do-pliku-testng.xml [kolejne pliki]
```

- z poziomu Anta

```
<path id="classpath.build">
  <!-- niezbędne biblioteki, w tym TestNG
       (...)
   -->
  <pathelement location="${build.dir}"/>
</path>
<taskdef resource="testngtasks"
         classpathref="classpath.build"/>
<target name="testng-all">
  <testng classpathref="classpath.build" outputDir="${report.dir}">
    <xmlfileset dir="${build.dir}" includes="testng.xml"/>
  </testng>
</target>
```

- z poziomu IDE (Eclipse, IntelliJ IDEA, NetBeans)

Prosty przykład (JUnit 4)

```
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class LectureComparatorTest {
    private LectureComparator lectureComparator = new LectureComparator();

    @Test
    public void compareLectureToItself() {
        Lecture lecture = new Lecture();

        int comparisonResult = lectureComparator.compare( lecture, lecture );

        assertEquals( 0, comparisonResult );
    }
}
```

Sprawdzanie wyrzucanych wyjątków (JUnit 3.8.1)

```
public void testCompareLecturesWithUnsetCategory() {
    Lecture firstLecture = new Lecture();
    Lecture secondLecture = new Lecture();

    try {
        lectureComparator.compare( firstLecture, secondLecture );

        fail();
    } catch ( NullPointerException e ) {
        assertTrue( true );
    }
}
```

Sprawdzanie wyrzucanych wyjątków (TestNG)

```
@Test( expectedExceptions = { NullPointerException.class } )
public void compareLecturesWithUnsetCategory() {
    Lecture firstLecture = new Lecture();
    Lecture secondLecture = new Lecture();

    lectureComparator.compare( firstLecture, secondLecture );
}
```

Sprawdzanie wyrzucanych wyjątków (JUnit 4)

```
@Test( expected = NullPointerException.class )
public void compareLecturesWithUnsetCategory() {
    Lecture firstLecture = new Lecture();
    Lecture secondLecture = new Lecture();

    lectureComparator.compare( firstLecture, secondLecture );
}
```

Inicjalizowanie testów i sprzątanie po nich (JUnit 3.8.1)

```
public class ConferenceManagerTest extends TestCase {
    private ConferenceManager conferenceManager;

    public ConferenceManagerTest() {
        System.out.println( "Inicjalizacja obiektu klasy ConferenceManager" );

        conferenceManager = new ConferenceManager();
    }

    protected void setUp() throws Exception {
        super.setUp();

        System.out.println( "Przydział zasobów dla metody testującej" );
    }

    protected void tearDown() throws Exception {
        System.out.println( "Zwolnienie zasobów użytych w metodzie testującej" );

        super.tearDown();
    }

    public void testMethod1() {
        System.out.println( "Test pierwszej metody klasy ConferenceManager" );
    }

    public void testMethod2() {
        System.out.println( "Test drugiej metody klasy ConferenceManager" );
    }
}
```

Inicjalizowanie testów i sprzątanie po nich (TestNG)

```
public class ConferenceManagerTest {
    private ConferenceManager conferenceManager;

    @BeforeClass
    private void initConferenceManager() {
        System.out.println( "Inicjalizacja obiektu klasy ConferenceManager" );

        conferenceManager = new ConferenceManager();
    }

    @BeforeMethod
    private void setUpTestMethod() {
        System.out.println( "Przydział zasobów dla metody testującej" );
    }

    @AfterMethod
    private void tearDownTestMethod() {
        System.out.println( "Zwolnienie zasobów użytych w metodzie testującej" );
    }

    @Test
    public void method1() {
        System.out.println( "Test pierwszej metody klasy ConferenceManager" );
    }

    @Test
    public void method2() {
        System.out.println( "Test drugiej metody klasy ConferenceManager" );
    }
}
```

Inicjalizowanie testów i sprzątanie po nich (JUnit 4)

```
public class ConferenceManagerTest {
   private static ConferenceManager conferenceManager;

    @BeforeClass
    public static void initConferenceManager() {
        System.out.println( "Inicjalizacja obiektu klasy ConferenceManager" );

        conferenceManager = new ConferenceManager();
    }

    @Before
    public void setUpTestMethod() {
        System.out.println( "Przydział zasobów dla metody testującej" );
    }

    @After
    public void tearDownTestMethod() {
        System.out.println( "Zwolnienie zasobów użytych w metodzie testującej" );
    }

    @Test
    public void method1() {
        System.out.println( "Test pierwszej metody klasy ConferenceManager" );
    }

    @Test
    public void method2() {
        System.out.println( "Test drugiej metody klasy ConferenceManager" );
    }
}
```

Wyłączanie niektórych testów

- JUnit 3.8.1:

```
public class ConferencePlannerTest extends TestCase {
    private ConferencePlanner conferencePlanner;

    public void _testPlanConferenceDay() {
        // test nieadekwatny do kodu, tymczasowo wyłączony
    }
}
```

- TestNG – na poziomie *testng.xml* lub:

```
public class ConferencePlannerTest {
    private ConferencePlanner conferencePlanner;

    @Test( enabled = false )
    public void planConferenceDay() {
        // test nieadekwatny do kodu, tymczasowo wyłączony
    }
}
```

Wyłączanie niektórych testów

- JUnit 4:

```
public class ConferencePlannerTest {
    private ConferencePlanner conferencePlanner;

    @Ignore
    @Test
    public void planConferenceDay() {
        // test nieadekwatny do kodu, tymczasowo wyłączony
    }
}
```

EQUILIBRIUM

Zaawansowane możliwości TestNG

- Grupowanie testów
- Metody zapewniające dane dla testów
- Definiowanie zależności pomiędzy testami
- Szybkie uruchamianie tylko tych testów, które poprzednio zakończyły się błędem
- ...

Grupowanie testów

```
@Test( groups = "unitTests" )
public void compareLectureToItself() {
    Lecture lecture = new Lecture();

    int comparisonResult = lectureComparator.compare( lecture, lecture );

    AssertJUnit.assertEquals( 0, comparisonResult );
}
```

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="JDD2006 Suite" verbose="2">
  <test name="Unit Tests" annotations="JDK5">
    <groups>
      <run>
        <include name="unitTests"/>
      </run>
    </groups>
    <packages>
      <package name="equilibrium.jdd2006.testng"/>
    </packages>
  </test>
</suite>
```

Metody zapewniające dane dla testów

```
@Test( dataProvider = "lectureAndExpectedComparisonResultProvider" )
public void compareLectures( Lecture firstLecture, Lecture secondLecture,
        int expectedComparisonResult ) {
    int comparisonResult
        = lectureComparator.compare( firstLecture, secondLecture );

    AssertJUnit.assertEquals( expectedComparisonResult, comparisonResult );
}

@DataProvider( name = "lectureAndExpectedComparisonResultProvider" )
private Object[][] provideLecturesAndExpectedComparisonResults() {
    return new Object[][] {
        { buildLecture( 1, 3, 4 ), buildLecture( 2, 1, 1 ), -1 },
        { buildLecture( 5, 2, 1 ), buildLecture( 3, 5, 4 ), 1 },
    };
}

private Lecture buildLecture( int lectureCategoryPriority,
        int lecturerPriority, int lecturePriority ) {
    // budowanie obiektu Lecture z podanymi priorytetami
}
```

Definiowanie zależności pomiędzy testami

```
public class ConferenceManagerTest {
    private ConferenceManager conferenceManager;

    @Test
    public void checkConfiguration() {
        conferenceManager = new ConferenceManager();

        // test sprawdzający poprawność konfiguracji
    }

    @Test( dependsOnMethods = { "checkConfiguration" } )
    public void planConference() {
        // test metody odpowiedzialnej za zaplanowanie całej konferencji
    }
}
```

EQUILIBRIUM

Szybkie uruchamianie tylko testów zakończonych błędem

- TestNG w momencie uruchomienia zestawu testów tworzy w katalogu wyjściowym plik *testng-failed.xml* (lub *testng-failures.xml*)
- Plik ten zawiera konfigurację uruchomieniową tylko tych testów, które poprzednio zakończyły się błędem
- Tak zdefiniowana konfiguracja zawiera również te testy, które nie zostały uruchomione z powodu niespełnienia zależności

EQUILIBRIUM

Dodatkowe zalety TestNG w kontekście...

- ...testów integracyjnych
- ...programowania sterowanego testami (*test driven development*)
- ...badania pokrycia kodu przez testy (*code coverage*)
- ...wykorzystania rozszerzeń biblioteki JUnit

EQUILIBRIUM

TestNG a testy integracyjne

- TestNG ułatwia tworzenie i utrzymanie testów integracyjnych
- Szczególnie przydatne są metody zapewniające dane (*@DataProvider*)
- Do wydzielenia testów integracyjnych można wykorzystać mechanizm grupowania
- Przydatne może być też definiowanie zależności i szybkie uruchamianie tylko tych testów, które zakończyły się błędem

EQUILIBRIUM

TestNG a TDD

- TestNG upraszcza tworzenie testów, wpływa pozytywnie na proces *red-green-refactor*
- Dzięki metodom zapewniającym dane można bez problemów przekazać różne zestawy danych do pojedynczego testu – nie musimy już pisać osobnego przypadku testowego
- Wykorzystanie adnotacji ułatwia pisanie testów do nieistniejącego kodu (mniej błędów kompilacji)

EQUILIBRIUM

TestNG a *code coverage*

- TestNG współpracuje z istniejącymi narzędziami do badania pokrycia kodu przez testy
- Integracja przeprowadzana na poziomie zadania Anta wymaga uwzględnienia specyfiki zarówno TestNG, jak i narzędzia do badania pokrycia kodu przez testy
- Przykład (Cobertura, Emma): http://www-128.ibm.com/developerworks/forums/dw_thread.jsp?forum=812&thread=110765&cat=10

EQUILIBRIUM

TestNG a rozszerzenia JUnit

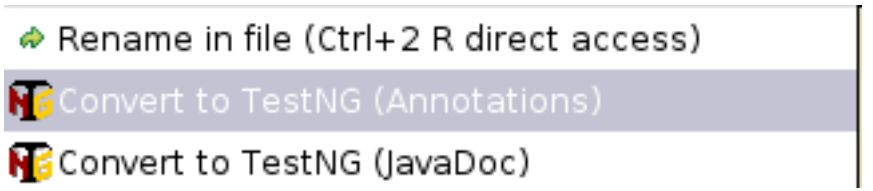
- Wiele rozszerzeń może po drobnych zmianach współpracować z TestNG
  http://thediscoblog.com/2006/03/27/using-junit-extensions-in-testng/
- Jeśli nie jest to możliwe, to zapewne dane rozszerzenie jest zbyt mocno uzależnione od JUnit, tak jak np. JMock:
  http://themindstorms.blogspot.com/2006/07/jmock-for-testng-or-junit-free-jmock.html

  http://themindstorms.blogspot.com/2006/08/more-on-jmock-and-testng.html

Migracja testów do TestNG

- Opcja pierwsza (łagodna): uruchamianie testów JUnit za pomocą TestNG i stopniowa migracja kodu

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="JDD2006 Suite" verbose="2">
  <test name="Unit Tests" annotations="JDK5" junit="true">
    <packages>
      <package name="equilibrium.jdd2006.junit3x"/>
    </packages>
  </test>
</suite>
```



EQUILIBRIUM

Migracja testów do TestNG

- Opcja druga (radykalna): wykorzystanie dołączonego do TestNG konwertera

```
$ java org.testng.JUnitConverter -overwrite -annotation -srcdir test
```

- W obu przypadkach nie obędzie się bez ręcznej ingerencji w kod testów, jednak bez wątpienia taka migracja w dużym stopniu zautomatyzowana stanowi kolejną zaletę TestNG

EQUILIBRIUM

Podsumowanie

EQUILIBRIUM

Więcej informacji

- http://testng.org
- http://www-128.ibm.com/developerworks/java/library/j-cq08296/?ca=dgr-lnxw07JUnit4vsTestNG
- http://www.theserverside.com/tt/articles/article.tss?l=MigratingtoTestNG
- http://www.realsolve.co.uk/site/tech/blog.php?name=philzoio&mydate=20050826
- http://www-128.ibm.com/developerworks/java/library/j-testng/
- http://membres.lycos.fr/testng
- http://www.devx.com/Java/Article/31983/

Dziękujemy za uwagę!