



J-Fall

11 oktober 2007 Spant! - Bussum



JUnit 4

Vereenvoudigde manier van Unit Testen



Levente Bokor

.nl.
jug



JUnit 4

- Test Driven Development
- Unit testing
- JUnit 4 stap voor stap... - demo



Test Driven Development

- Geen code zonder test
- Red/Green/Refactor
- Maakt onderdeel uit agile development
- Geautomatiseerd testen / test harness
- Toegevoegde waarde in het design proces
- Kwaliteit verhogend



.nl.
jug



Unit testing

- Wat is unit testen?
- Voordelen
 - Ondersteuning in het wijzigingsproces
 - Goed uitgangspunt voor integratie(testen)
 - Documentatie
 - Scheiding interface van implementatie
- Beperkingen
 - Geeft geen garantie voor een foutloos systeem
 - Het laat je de aanwezigheid van fouten zien, maar niet de afwezigheid



JUnit 4 stap voor stap...

- Wat is JUnit?
 - Unit test framework voor de Java taal
 - Kan ook gebruikt worden voor integratie testen
 - Gemaakt door Kent Beck and Erich Gamma
 - Het meest succesvolle xUnit framework
- Versie 3.x vs. versie 4.x
 - Versie 4.x backward compatible
 - Versie 4.x gebaseerd op java 5
 - Versie 4.x makkelijker in gebruik



JUnit 4 stap voor stap...

- Opzetten omgeving:
 - Eclipse 3.3
 - JUnit library (versie 4.4)
- Use case
 - Bouw een calculator met een beperkt aantal functies: delen, kwadrateren, is een getal priemgetal of niet, aan/uit
- Aanpak: TDD (eerst testcase → implementatie)



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van `@NotNull`
- `setUp` en `tearDown` methodes (fixtures)
- `@BeforeClass` en `@AfterClass`
- Exception handling
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
package junit3;

import calc.Calculator;
import junit.framework.TestCase;
import org.junit.*;

public class CalculatorTest extends TestCase
{
    public class CalculatorTest4 {

    }
}
```



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van **@annotaties**
- setUp en tearDown methodes
- @BeforeClass en @AfterClass annotaties
- Exception handling
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
@?????eClass  
public static void switchOnCalculator() {  
  
    @AfterClass  
    public static void switchOffCalculator {  
        ...  
    }  
  
    @Before  
    public void clearCalculator() {  
        ...  
    }  
  
    @Test()  
    public void add() {}  
}
```



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van @annotaties
- **setUp en tearDown methodes (fixtures)**
- @BeforeClass en @AfterClass annotaties
- Exception handling
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
public void setUp() {  
    calculator.clear();  
}  
  
public void tearDown() {  
    @After calculator.clear();  
}  
  
public void clearCalculatorAfter() {  
    calculator.clear();  
}
```



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van @annotaties
- setUp en tearDown methodes (fixtures)
- **@BeforeClass en @AfterClass annotaties**
- Exception handling
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
@BeforeClass
public static void switchOnCalculator() {
    calculator.switchOn();
}

@AfterClass
public static void switchOffCalculator() {
    calculator.switchOff();
}
```

?



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van @annotaties
- setUp en tearDown methodes (fixtures)
- @BeforeClass en @AfterClass annotaties
- **Exception handling**
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
@Test(expecteds=ArithmeticException.class)
public void divideByZero() {
    calculator.divide(0);
    fail();
} catch (ArithmeticException e) {
}
}
```



JUnit 4 stap voor stap...

- Nieuwe package naam
- Gebruik van @annotaties
- setUp en tearDown methodes
- @BeforeClass en @AfterClass
- Exception handling
- **Testen overslaan**
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
@Test  
@Ignore("not ready yet")  
public void multiply() {  
    calculator.add(10);  
    calculator.multiply(10);  
    assertEquals(calculator.getResult(), 100);  
}
```



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van @annotaties
- setUp en tearDown methodes (fixtures)
- @BeforeClass en @AfterClass annotaties
- Exception handling
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
@Test(timeout=2000)
public void testPrime() {
    assertTrue(calculator.isPrime(15485863) );
}
```



JUnit

- Nieuwe package
- Gebruik van @ar
- setUp en tearDown
- @BeforeClass
- Exception han
- Testen overs
- Performance tests
- **Geparameteriseerde tests**
- Test suite
- Assert methodes

```
public void testSquare() {  
    int[][] params = new int[][] { { 1, 1 },  
        { 2, 4 }, { 3, 9 }, { 4, 16 },  
        { 5, 25 }, { 6, 36 } };  
    Calculator calculator = new Calculator();  
    for (int i = 0; i < params.length; i++) {  
        calculator.square(params[i][0]);  
        assertEquals(params[i][1], calculator.getResult());  
    }  
    return Arrays.asList(new Object[][] { { 1, 1 },  
        { 2, 4 }, { 3, 9 }, { 4, 16 }, { 5, 25 }, { 6, 36 } });  
}  
} // of  
public void testSquare3() {param, int result) {  
    calculator.square(param);  
    assertEquals(9, calculator.getResult());  
}  
} // of  
public void testSquare4() {  
    calculator.square(4);  
    assertEquals(16, calculator.getResult());  
}  
} etc.
```



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van @annot
- setUp en tearDown
- @BeforeClass en @AfterClass
- Exception handling
- Testen overslaan
- Performance tests
- Geparametriseerde tests
- **Test suite**
- Assert methodes

```
public class AllCalculatorTests {  
    @Sui public static TestSuite suite() {  
        TestSuite suite = new TestSuite("Calculator Tests");  
        suite.addTest(new CalculatorTest());  
        suite.addTest(new SquareTest());  
    }  
    public return suite; }  
}
```



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van @annotaties
- setUp en tearDown methodes (fixtures)
- @BeforeClass en @AfterClass annotaties
- Exception handling
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes

```
public void testAdd() { assertEquals(  
    calculator.add(1), 2, Object[] actual)  
    public calculator v = new calculator();  
    assertEquals(  
        v.add(1), 2, Object[] actual)  
    assertEquals(calculator.getResult(), 2);  
    }  
    Object[] expected, Object[] actual)
```



JUnit 4 stap voor stap...

- Nieuwe package naam/ static import
- Gebruik van @annotaties
- setUp en tearDown methodes (fixtures)
- @BeforeClass en @AfterClass annotaties
- Exception handling
- Testen overslaan
- Performance tests
- Geparameteriseerde tests
- Test suite
- Assert methodes



JUnit versie 4.4

```
assertThat(string, containsString("abc"));
```

- Assertion syntax

```
assumeThat(File.separatorChar, is('/'));
```

- Expliciet ondersteuning voor aannames

```
@Theory public void filenameIncludesUsername(String username) {  
    assumeThat(username, not(containsString("/")));  
    assertThat(new User(username).configFileName(),  
        containsString(username));  
}
```

- Theories (globale waarheden)

.nl.
jug



Samenvatting

- Test Driven Development
 - Onmisbaar voor kwaliteit
 - Hoe eerder je ermee start des te beter
- JUnit 4.x vs. oudere versie(s)
 - Gebaseerd op Java 5
 - Flexibel
 - Nieuwe features
 - Backward compatibel
 - Eclipse integratie

.nl.
jug



Vragen

