

Unit Tests

Einführung

Praxisprojekt des Studienganges Angewandte Informatik im Wintersemester 2003/04

Gudrun Fischer

Was sind Unit Tests?

- Unit = Komponente
- Test:
 1. Jede Komponente einzeln
 2. Zusammengesetztes System
- Testsprache =
Implementierungssprache
→ bei uns Java
- Design-Konzept: Test First

Test First

- Was soll meine Komponente können?
- Wie beweise ich, dass sie es kann?
 - Erfolgreicher Test
- Zyklus
 - Tests schreiben
 - Komponente besteht Tests nicht
 - Komponente erweitern
 - Tests erfolgreich

Entwicklung

- Kurze Zyklen
 - Ca. 1 Woche
 - Funktionalität in „Happen“
- Mikrozyklus
 - Ca. 10 Minuten
 - Funktionalität in „Häppchen“
 - Test erweitern
 - Komponente besteht Test nicht
 - Komponente erweitern
 - Test erfolgreich

Beispiel

- Makroschritt: Klasse Record
 - Was soll ein Record können?
- Mikroschritte:
 - Erzeugen eines Records
 - Aus String
 - Aus InputStream
 - Parsen
 - Zugriff auf Felder
 - ...

Beispiel: Erzeugen aus String

- Wird überhaupt ein Record-Objekt erzeugt?
- Hat das Objekt die richtigen Eigenschaften?
- Hat ein weiteres, erzeugtes Objekt die richtigen, *anderen* Eigenschaften?
- Werden ggf. die richtigen Exceptions erzeugt?

Konventionen

- Package `de.unidu.is.invex`
 - Test-Package `test.de.unidu.is.invex`
- Klasse `Record`
 - Test-Klasse `RecordTest`
- Methode/Mikroschritt `bla`
 - Test-Methode `testBla`
- Klasse `AllTests`

Aufbau einer Test-Klasse

- Erbt von TestCase
- Konstruktor Klassenname(String arg0)
- setUp()
- testBlaWithParameters()
 - assert(...)
- tearDown()
- Wichtig:

Jedes testBla läuft unabhängig!

```
public class StemmerFilterTest extends TestCase {

    private StemmerFilter filter;

    /**
     * Constructor for SoundexFilterTest.
     * @param arg0
     */
    public StemmerFilterTest(String arg0) {
        super(arg0);
    }

    /**
     * @see TestCase#setUp()
     */
    protected void setUp() throws Exception {
        filter = new StemmerFilter(null);
    }
}
```

```
public void testRun() {
    assertEquals(filter.run("jumps"),
        filter.run("jump"));
}

public void testRunIdempotency() {
    assertEquals(filter.run("jump"),
        filter.run("jump"));
}

public void testRunNotSame() {
    assertNotSame(filter.run("hello"),
        filter.run("world"));
}
}
```

Wie geht es weiter?

- Noch nicht angesprochen
 - Tests mit Dateien
 - Dummy- und Mock-Objekte
 - Vererbung
 - Interfaces
- Wie viel ist genug?
- Verwendung im Projekt
 - Tests für eigene Klassen
 - Tests für fremde Klassen

Quellen

- Unit Tests mit Java

 - Der Test-First-Ansatz

 - Johannes Link

 - dpunkt 2002

- Package `test.de.unidu.is.text`

 - H. Nottelmann