TDD By Example

Daniel Hiller, dbap GmbH

dbap.

Über mich

- Java seit 1996 (mit Unterbrechungen)
- Seit 2002 bei dbap
  - Backend Administration Software
  - Webservice als Zwischenschicht zur Datenermittlung für Frontends und externe Dienstleister
  - Schwerpunkte: Qualitätssicherung, R&D

Was ist ein Test?

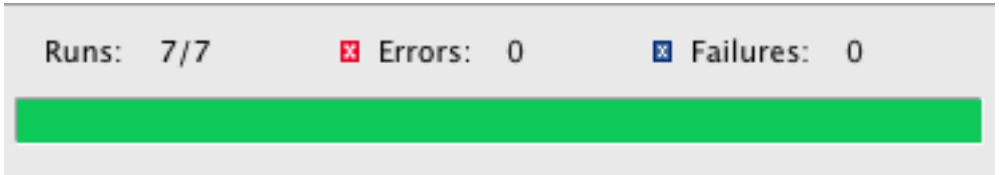
Was ist ein Test?

*„Ein **Test** … ist ein **Versuch**, mit dem größere Sicherheit darüber gewonnen werden soll, ob … ein Vorgang innerhalb der geplanten Rahmenbedingungen **funktioniert** bzw. ob bestimmte Eigenschaften vorliegen oder nicht.“*

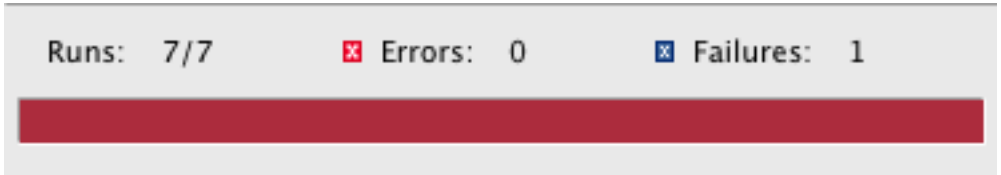
Quelle: http://de.wikipedia.org/wiki/Test

Ergebnis

Funktioniert



Funktioniert nicht



Konventionelle Entwicklung

- Big Design Upfront
- Feedback erst, wenn alles fertig ist
- „Wird das Design die Anforderungen erfüllen?“

- Tests
  - Entweder „machen wir später“ (also nie)
  - Oder nur Integrations-Tests

Test-Getriebene Entwicklung

TDD = Test-First = Test-Driven

- Tests **sind** Anforderungen
- Feedback zu jeder Zeit
- Evolutionäres Design
- YAGNI

Eigenschaften eines Tests

- deterministisch
- isoliert
- schnell
- einfach
- leicht verständlich

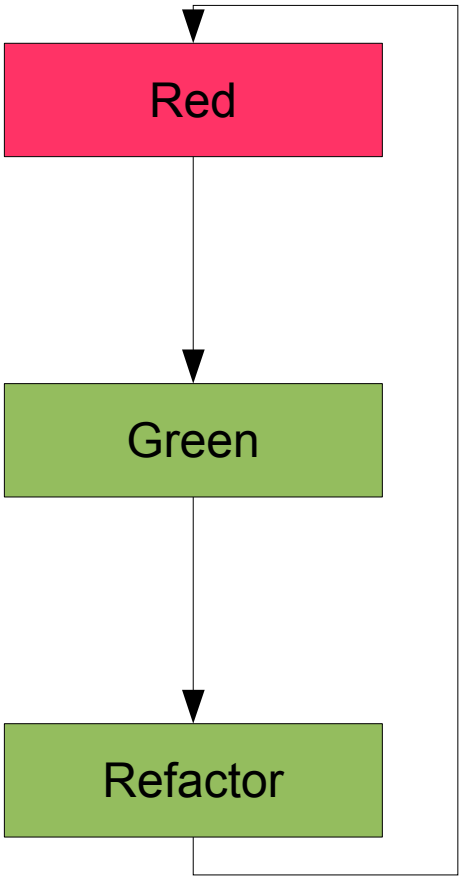
Prinzipien

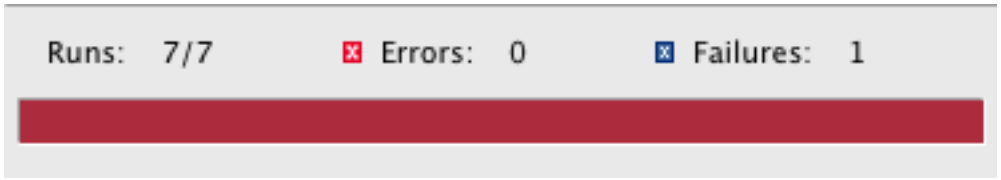
- Ein Test pro „Thema“
- **Red – Green – Refactor**
  - Zuerst der Test
  - Dann der Code
  - Dann das Aufräumen

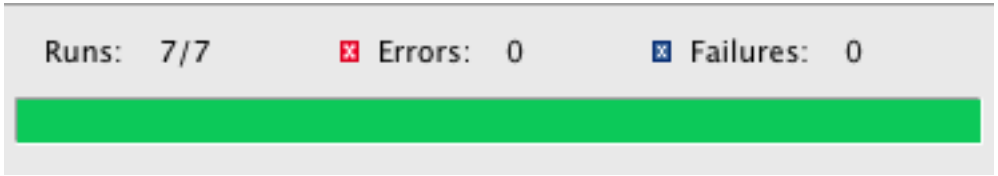
Man kann nicht jeden Fall testen

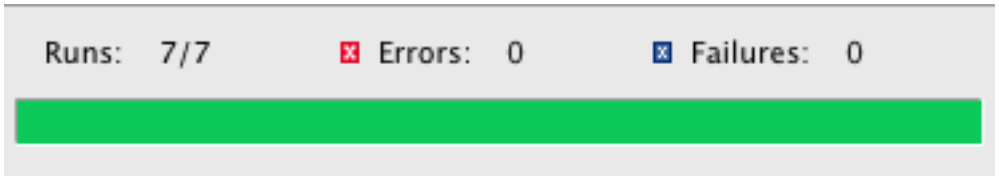
- Keine trivialen Tests
- Grenzfälle

Iteration


Red
Green
Refactor

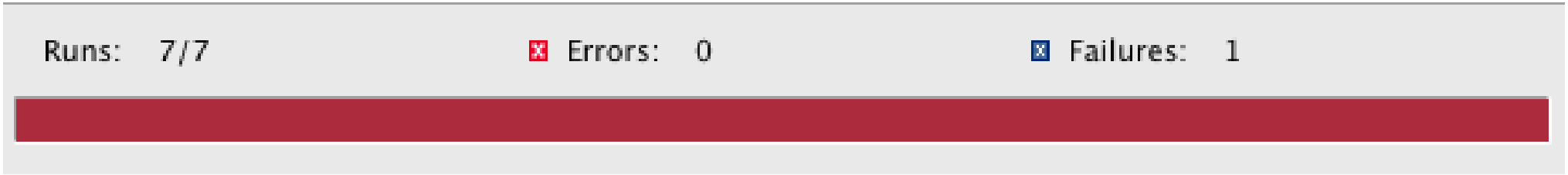

Runs: 7/7
Errors: 0
Failures: 1


Runs: 7/7
Errors: 0
Failures: 0


Runs: 7/7
Errors: 0
Failures: 0

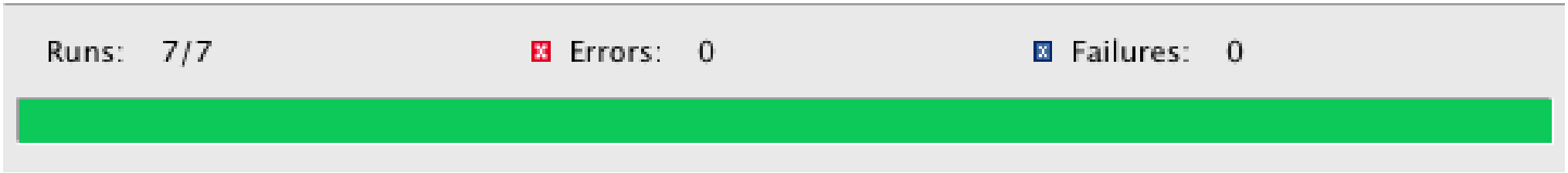
Red

- Schreibe den neuen Test
- Führe die Test-Suite aus
- Stelle sicher, dass der neue Test fehlschlägt
- Stelle sicher, dass keiner der alten Tests fehlschlägt



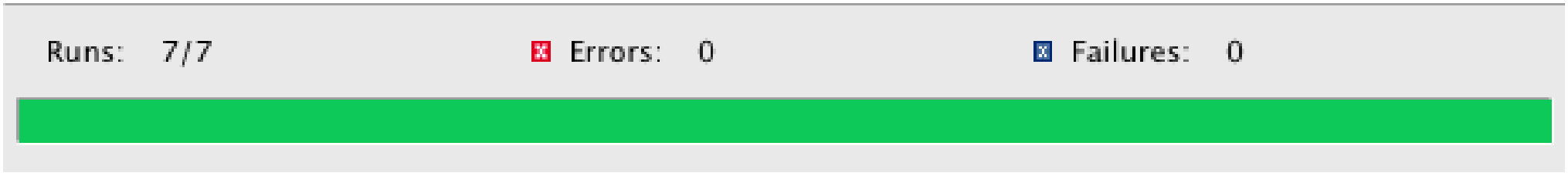
Green

- Schreibe gerade so viel Code, um den Test zu erfüllen
- Stelle sicher, dass kein anderer Test fehlschlägt



Refactor

- Strukturverbesserung von Programm-Quelltexten
- Beibehaltung des beobachtbaren Programm-Verhaltens
- Verbesserung der Lesbarkeit, Verständlichkeit, Wartbarkeit und Erweiterbarkeit
- Haupt-Ziel: Aufwand für Fehleranalyse und funktionale Erweiterungen senken



Wie sieht ein Test aus?

```
package de.jug_muenster.tdd_talk;

import org.junit.Test;

public class UnitTest {

  @Test
  public void testUnit() throws Exception {
    new Unit();
  }

  @Test(expected = NullPointerException.class)
  public void testGetPartWithNull() throws Exception {
    new Unit().getPart( null );
  }

}
```

Code Kata

- Learning By Practicing statt Learning By Doing
- „Hands-On“ Beispiel eines realen Problems
- Verbesserung durch Wiederholung

Kata: Args

```
java MyClass -l -p 8080 -d /usr/logs
```



```
for (String arg:args) { … }
```

```
[
   Flag [name='l', value=TRUE (Boolean)],
   Flag [name='p', value=8080 (Integer)],
   Flag [name='d', value='/usr/logs' (String)]
]
```

Coding...

Beobachtungen?

Pro - Contra

- Sicherheit durch sofortiges Feedback nach Änderungen
- Testfälle für Bugs verhindern Regression
- Leicht erhöhter Aufwand, der sich langfristig auszahlt

- Maintenance der Tests fällt zusätzlich an
- Vergrößerte Infrastruktur zum Testen
- Weiterer Lernaufwand durch Testing Frameworks
- Widerstand der Geschäftsführung („unnötiger Aufwand“)

Fragen? Meinungen?

Kontakt:

- daniel.hiller.1972@googlemail.com
- http://twitter.com/dhiller72

Quellen

Literatur

- eXtreme Programming explained (Kent Beck)
- Refactoring (Martin Fowler)
- Clean Code (Robert C. Martin)

Links

- http://en.wikipedia.org/wiki/Test-driven_development
- http://de.wikipedia.org/wiki/Testgetriebene_Entwicklung
- http://de.wikipedia.org/wiki/Grey-Box-Test
- http://de.wikipedia.org/wiki/Refactorings
- http://www.junit.org/
- http://codekata.pragprog.com/

In eigener Sache...

Auf Jobsuche?

dbap hat Stellen zu besetzen

Interesse?

http://www.dbap.de/jobs

Danke für's Zuhören