andrena OBJECTS

Experts in agile software engineering

Infrastruktur für Tests: Einfaches Setup, bessere Asserts

Franziska Sauerwein, andrena objects ag

Michael Haug, andrena objects ag

Motivation, Problemstellung

„Testen ja, aber bei uns geht echtes TDD eh nicht“

„Unsere Strukturen sind viel zu komplex, um getestet zu werden“

„Integrationstests reichen aus, alles Weitere ist zu viel Aufwand“

Motivation, Problemstellung

Schwierigkeiten:

- Komplexes Objektmodell
- Komplexe Ergebnisse / Rückgabewerte
- Kein echtes OO / Logik in Services (hier nicht behandelt)

Lösungen:

- Erstellung und Überprüfung von Objekten vereinfachen
- Objektstrukturen für Tests vorgeben
- Wiederverwendbare Infrastruktur für Tests schaffen

Zusammenfassung Builder

Fokus auf relevante Daten für Test

- Kapselung von für Test nicht relevanter aber notwendiger Daten
- Hervorhebung für Test relevanter Daten

Hohe Flexibilität

- Leichte Anpassbarkeit des Testcodes
  - Änderung des Objektmodells
  - Änderung der Validierung für ein „gültiges“ Objekt

Zusammenfassung Builder

Hohe Wiederverwendbarkeit

- Vorteil gegenüber Oberklassen
- Vorteile gegenüber Testdaten-Ersteller
  - Entkopplung
  - Dynamische Erstellung

Vorteile von Matchern

- Wiederverwendbarkeit
- Bessere Lesbarkeit der Erwartung
- Präzisere Fehlermeldung bei Fehlschlag
- Kombinierbarkeit (z.B.: both… and… / contains… / anyOf, allOf)

Vorteile von Matchern

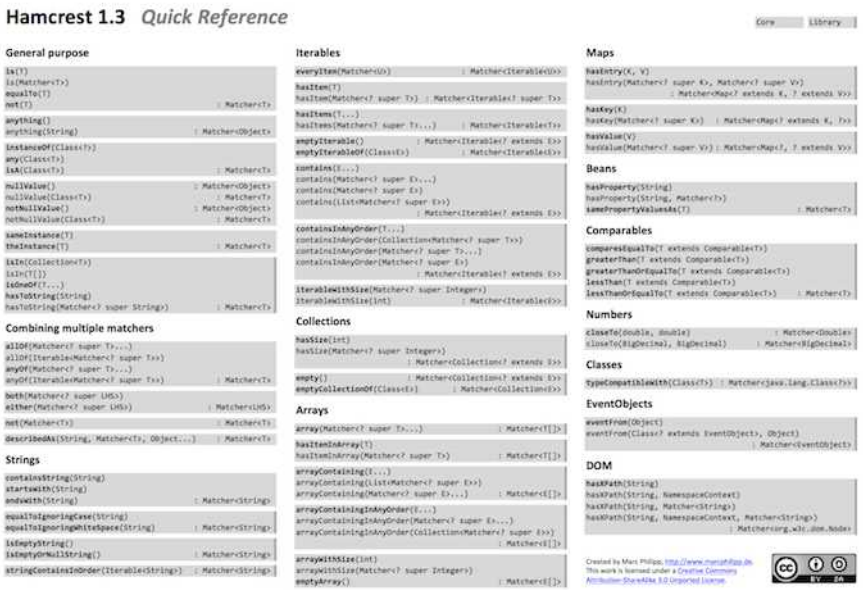
- Einfachere Formulierung von asserts durch Kombinierbarkeit

- Viele Standard-Matcher (hamcrest-library)
  - Wesentlich mächtiger als traditionelle assert-Methoden

- Actual und Expected sind typisiert (und damit nicht vertauschbar)

Zusammenfassung Matcher + Matcher Übersicht

- Matcher-Übersicht:
  http://www.marcphilipp.de/blog/2013/01/02/hamcrest-quick-reference/
- Einstiegspunkte:
  - FeatureMatcher
  - TypeSafeDiagnosingMatcher
- Traditionelle asserts sind einfach migrierbar



Vortragszusammenfassung

- Ziel: Gute und einfach lesbare Tests
  - So lang wie nötig (Eingaben und Erwartung lesbar)
  - So knapp wie möglich (unnötiges weggekapselt)
  - Der Leser des Tests versteht auf Anhieb, was der Test macht
  - Fehlschlag gibt Auskunft über Problem, ohne dass Test-Code analysiert werden muss

- Lösungsansatz:
  - Builder und Matcher

Testpyramide