akquinet

Stefan Roock
stefan.roock@akquinet.de

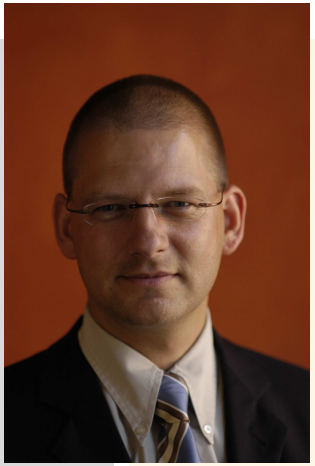
Bernd Schiffer
bernd.schiffer@akquinet.de

Beispiel Grails

Brauchen wir neue Technologien für agile Entwicklung?

Agile Alliance Corporate Member

Herzlich Willkommen!



Stefan Roock

- akquinet AG, Hamburg (D)
- Berater, Projektleiter und entwickelnder Architekt
- Agile Methoden (Schwerpunkte: XP und Scrum)
- J2EE-Projekte vereinfachen.

Herzlich Willkommen!

Bernd Schiffer

- akquinet AG, Hamburg (D)
- Softwareentwickler
- Agile Softwareentwicklung (Schwerpunkt: XP)

Akquinet AG

- Sitz in HH, tätig im deutschsprachigen Raum und angrenzendes Europa.
- Entwicklung, Beratung, Schulung, Outsourcing
- Agile Methoden, Java, .NET, SAP, Navision, zwei Rechenzentren
- > 150 Mitarbeiter
- http://www.akquinet.de

- it-agile als Schulungs- und Beratungsbaukasten rund um agile Softwareentwicklung.
- http://www.it-agile.de

Was Sie heute erwartet

- These: **„Grails unterstützt agile Entwicklung besser.“***
- Was ist Grails?
- Merkmale
- Unterstützung bei agiler Entwicklung

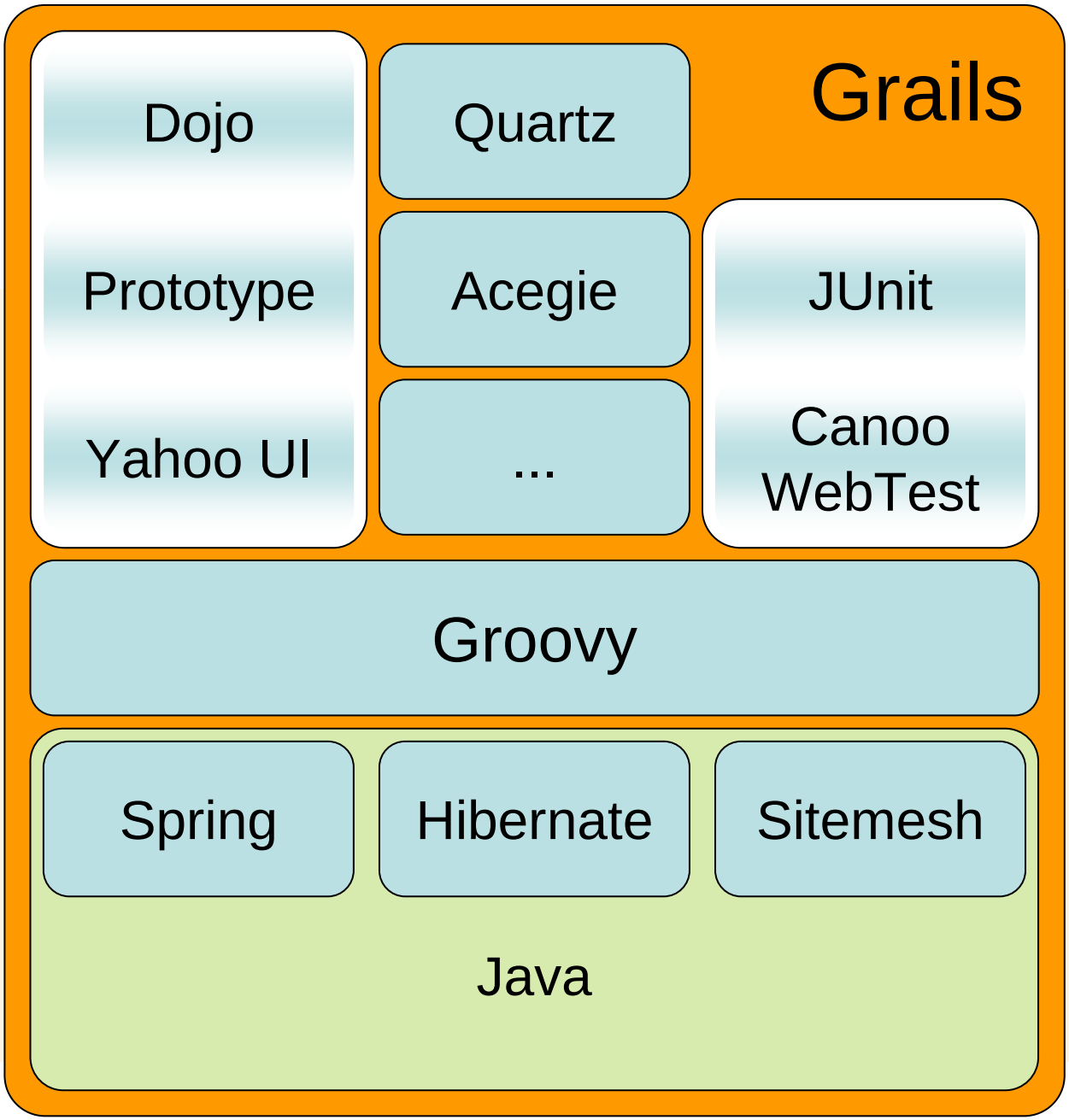
* ... als z.B. Java.

Was ist Grails?

- Yet Another (Web-)Application Framework
  - The Chosen One! (← das Auserwählte!)
- Open-Source
- geschrieben in und benutzt **Groovy**
- Prinzipien
  - Konvention statt Konfiguration (**Coding by Convention**)
  - Einfachheit (**KISS**)
  - keine Redundanz (**DRY**)
- ehem. Vorbild: Ruby on Rails

Grails – Architektur

- trittfester Boden



Was ist Groovy?

akquinet

- Skript-Sprache
- Open-Source
- nahtlose Integration in die Java-Welt
- Features, die wirklich helfen:
  - **Closures**
  - **GroovyBeans**
  - **Collections**
  - **Reguläre Ausdrücke**
- XML, SQL, Unittests, Mocks, und noch vieles mehr...

Groovy: Collections

- Collections zu deklarieren, initialisieren oder damit zu arbeiten ist **öde**:
  - Beispiel Deklaration + Intitialisierung in Java

```
List<Integer> list = new ArrayList<Integer>();
list.add(1);
list.add(2);

Map<String, Integer> map = new HashMap<String, Integer>();
map.put("eins", 1);
map.put("zwei", 2);

List oneItem = Collections.singletonList(1);
```

Groovy: Collections

- Collections zu deklarieren, initialisieren oder damit zu arbeiten ist **super**:
  - Beispiel Deklaration + Intitialisierung in Groovy

```
def list = [1, 2]
def map = [eins:1, zwei:2]
def oneItem = [1]
```

Groovy: Collections

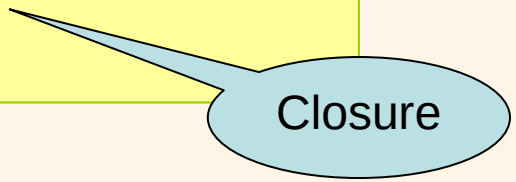
- Collections zu deklarieren, initialisieren oder damit zu arbeiten ist **öde**:
  - Beispiel Aufbauen und Suchen von Ranges in Java

```
List<Integer> list = new ArrayList<Integer>();
for(int index = 0; index <= 3000; index++) { list.add(index); }
List<Integer> gefilterteListe = new ArrayList<Integer>();
for (Integer integer : list) {
   if (1000 <= integer && integer <= 2000)
            gefilterteListe.add(integer);
}
assert gefilterteListe.size() == 1001;
```

Groovy: Collections

- Collections zu deklarieren, initialisieren oder damit zu arbeiten ist **super**:
  - Beispiel Aufbauen und Suchen von Ranges in

```
def list = (0..3000)
def gefilterteListe = list.findAll{ 1000 <= it && it <= 2000 }
assert gefilterteListe.size() == 1001
```



Groovy: Closures

- Closures sind ausführbare Codefragmente mit Zugriff auf den Kontext zur Laufzeit
  - vergl.
    - Java: anonyme Klassen
    - C++: Function Pointer

Groovy: Closures

- Beispiel:

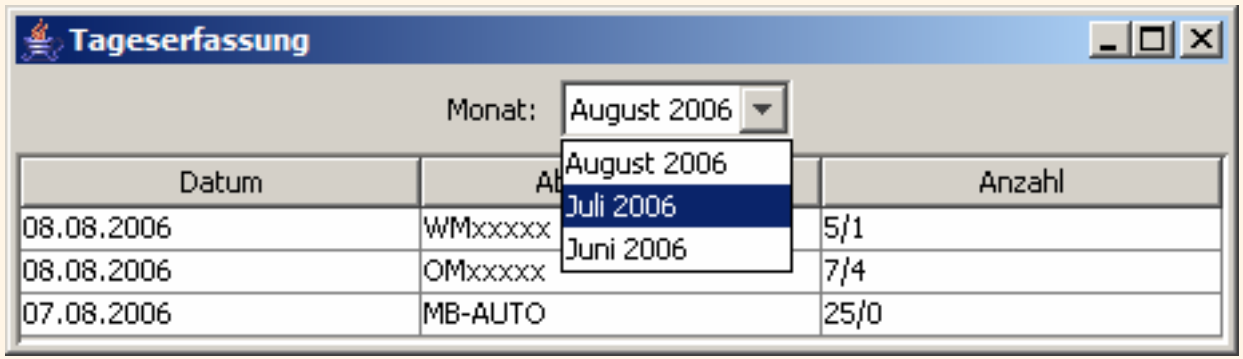
```
class Strecke {
   def länge
   def toString() { länge }
}

def strecken = [      new Strecke(länge:10),
                      new Strecke(länge:21.097),
                      new Strecke(länge:42.195)      ]
assert strecken[0].länge == 10

def längerAls(länge) { return { strecke -> strecke.länge > länge } }
def lang = längerAls(20)
def langeStrecken = strecken.findAll(lang)
assert strecken[1..2] == langeStrecken
```

Groovy-Prototyp: Swing

- aus dem echten Leben gegriffen
- dem Kunden mal etwas zeigen wollen
- Spikesolution mit **Groovys SwingBuilder**



Groovy-Prototyp: Swing

```
def swing = new SwingBuilder() // Importe ausgelassen
def frame = swing.frame(title:'Tageserfassung', location:[200,200]) {
    panel(layout:new BorderLayout()) {
        panel(constraints:BorderLayout.NORTH, layout:new FlowLayout()) {
            label(text:'Monat: ')
            comboBox(items:['August 2006', 'Juli 2006', 'Juni 2006'])
        }
        scrollPane(constraints:BorderLayout.CENTER) {
            table() {
                def model = [
                    [datum:'08.08.2006', ableseeinheit:'WMxxxxx', anzahl:'5/1'],
                    [datum:'08.08.2006', ableseeinheit:'OMxxxxx', anzahl:'7/4'],
                    [datum:'07.08.2006', ableseeinheit:'MB-AUTO', anzahl:'25/0'],
                ]

                tableModel(list:model) {
                    closureColumn(header:'Datum', read:{row -> return row.datum})
                    closureColumn(header:'Ableseeinheit', read:{row -> return row.ableseeinheit})
                    closureColumn(header:'Anzahl', read:{row -> return row.anzahl})
                }
            }
        }
    }
}
frame.pack()
frame.show()
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
```

Grails-Prototyp: Web

- Struktur vorgegeben: **Scaffolding**
- Beispiel: Buchhandel (auch online in Grails-Doku)
- → **Live-Demo**!

Unterstützung bei Agilität?

- Wo genau **unterstützt** uns denn Grails beim agilen Arbeiten?

Kürzere Turnaroundzeiten

- schnell beim Prototyp dank Scaffolding
  - es geht darum, dass man sich über etwas unterhalten kann
  - Prototyp als Vorlage für echtes Produkt
- keine langen Wartezeiten beim Entwickeln
  - beim Ändern der Domain-Class, Controler oder View
  - (noch) nicht beim Neuanlegen
  - Grund:
    - kein Kompilieren
    - dynamische Sprache

Architektur

- Best-off-Compilation
- Muster-Architektur
- vorgegebener, erweiterbarer Technologiestack
- Konventionen statt Konfigurationen

Das bedeutet

- Einfachheit
- Konzentration **aufs** Wesentliche

Lesbarkeit

- Bessere Lesbarkeit
  - [1]
    - statt Collections.singletonList(1);
    - oder List list = new ArrayList(); list.add(1);
  - extrem hoher Nutzen in Unittests!
  - Beispiel:

```
class Multiplicator {
   static multipliziere(list, factor) {
              list.collect{ it * factor }
   }
}
```

```
import Multiplicator as M

assert M.multipliziere([1], 1) == [1]
assert M.multipliziere([1], 2) == [2]
assert M.multipliziere([1], 0) == [0]
assert M.multipliziere([1, 2], 1) == [1, 2]
assert M.multipliziere([1, 2], 2) == [2, 4]
```

Lesbarkeit

- Konvention statt Konfiguration
  - ich kann mich auf entscheidende Dinge verlassen
  - nicht lange rumsuchen, wie wo wer mit wem was macht
  - ergo einfache Navigation im Code
- Pair-Programming
  - bessere Kommunikation **über** den Code **durch** den Code
  - weniger Aushandeln von Code-Konventionen
- kürzere Turnaroundzeiten bei TDD
  - kurze Turnaroundzeiten durch dynamische Sprache
  - daher schneller beim Test-Code-Test-Refactor-Test-Zyklus

Refactoring auf Algorithmusebene durch Closures

- Beispiel Sudoku oder Grafik in Java:

```
public static void main(String[] args) {
   Integer[][] matrix = { { 1, 2, 3 },
                          { 4, 5, 6 },
                          { 7, 8, 9 } };
   doubleFields(matrix);
   int sum = sum(matrix);
}

private static void doubleFields(
      Integer[][] matrix) {
   for (int y = 0; y < matrix.length; y++) {
      for (int x = 0; x < matrix[y].length; x++) {
         matrix[y][x] *= 2;
      }
   }
}
```

```
private static int sum(Integer[][] matrix) {
   int sum = 0;
   for (int y = 0; y < matrix.length; y++) {
      for (int x = 0; x < matrix[y].length; x++) {
         sum += matrix[y][x];
      }
   }
   return sum;
}
```

Refactoring auf Algorithmusebene durch Closures

- Beispiel Sudoku oder Grafik in Groovy (erster Wurf):

```
def matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

matrix.eachWithIndex{ ity, y ->
   ity.eachWithIndex{ itx, x ->
      matrix[y][x] *= 2
   }
}

def sum = 0
matrix.eachWithIndex{ ity, y ->
   ity.eachWithIndex{ itx, x ->
   sum += matrix[y][x]
   }
}
```

Refactoring auf Algorithmusebene durch Closures

- Beispiel Sudoku oder Grafik in Groovy (Refactoring mit Closures):

```
def matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]


eachWithCoords(matrix, { x, y ->
      matrix[y][x] *= 2 } )
def sum = 0
eachWithCoords(matrix, { x, y ->
      sum += matrix[y][x] } )

def eachWithCoords(matrix, closure) {
   matrix.eachWithIndex{ ity, y ->
      ity.eachWithIndex{ itx, x ->
         closure.call(x, y)
      }
   }
}
```

Refactoring auf Algorithmusebene durch Closures

- Beispiel Sudoku oder Grafik in Groovy (Refactoring mit Category):

```
class MatrixIterationCategory {
   static def eachWithCoords(self, closure) {
      self.eachWithIndex{ ity, y ->
         ity.eachWithIndex{ itx, x ->
               closure.call(x, y)
}}}}

def matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

def sum = 0
use(MatrixIterationCategory) {
   matrix.eachWithCoords{ x, y -> matrix[y][x] *= 2 }
   matrix.eachWithCoords{ x, y -> sum += matrix[y][x] }
}
```

Spikesolutions mit Prototypen

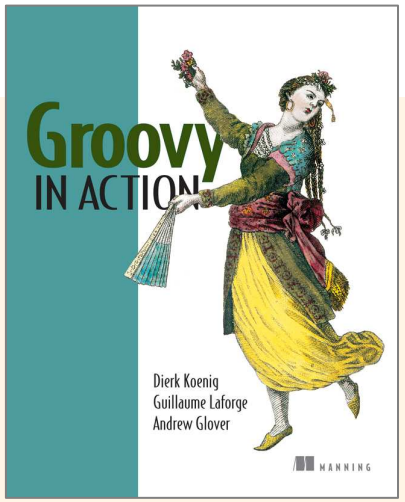
- Beispiele für
  - Swing mit Groovys SwingBuilder
  - Web mit Grails‘ Scaffolding
- „Ein Bild sagt mehr als tausend Worte.“
- ausführbare Spezifikation
- ABER: Prototypen nicht produktiv schalten!

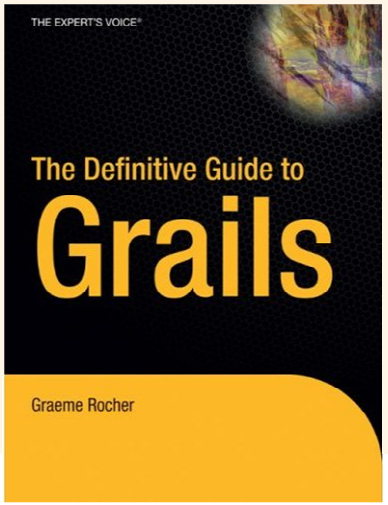
Nachteile

- wenig Unterstützung in IDEs (Code Completion, Assistenten, etc.)
  - ABER: Groovy- und Grails-Plugins für Eclipse und IntelliJs IDEA sind auf dem Weg
- keine automatischen Refactorings
  - Grund: dynamische Sprache
  - ABER: agiles Vorgehen hilft hier (Unit-Tests)

Weitere Informationen

- The Definitive Guide to Grails
  - (Apress, 2007) Graeme Rocher
- Agile Webentwicklung mit Grails
  - (Addison-Wesley, 2007) Roock, Schiffer
- Groovy in Action
  - (Manning, 2007)
    König, Glover, Laforge, King, Skeet
- Weitere Bücher in Arbeit





Weitere Informationen

- Webseiten
  - http://groovy.codehaus.org
  - http://grails.codehaus.org
  - Wikis, User Guides, Tutorials, Artikel, Howtos etc.
- Mailinglisten
  - http://groovy.codehaus.org/mail-lists.html (englisch)
  - http://grails.codehaus.org/Mailing+Lists (englisch)
  - http://de.groups.yahoo.com/group/grails-webframework (deutsch)

Vielen Dank für die Aufmerksamkeit

- Noch Fragen?

Wir bieten Schulungen zu Groovy und Grails an.

Schulung verlängerte Werkbank
agile Softwareentwicklung
Festpreisprojekte Coaching
RCP Systemintegration Eclipse
h3270 Hostintegration
Scrum Refactoring testgetriebene Entwicklung
Hibernate SAP-Netweaver OpenSource
Ajax JBoss/JEMS Groovy
eXtreme Programming