



Extreme Programming mit Rails

xpdays, 23. November 2007
Tammo Freese



Agenda

- Ruby on Rails wird als agil vermarktet
- Aber: Was ist Ruby on Rails agil, was nicht?
- Konkret: Wie gut ist Ruby on Rails für XP geeignet?
- Gegenüberstellung: Extreme Programming mit Java/Eclipse gegenüber Ruby on Rails



Mein Background

- In den letzten vier Jahren:
 - ca. 20 Monate in einem Java/Eclipse Projekt
 - ca. 20 Monate in zwei Ruby on Rails-Projekten
- In allen Projekten: Software Engineering mit XP-Techniken, insbes. Test-Driven Development, Incremental Design, Pair Programming



Extreme Programming

- Zugrundeliegende Werte
- Konkrete Techniken
- Verbindende Prinzipien

- Welcher Punkte aus jeder Kategorie werden durch die Programmiersprache am meisten beeinflusst?
 - Hier nur eine kleine Auswahl!



Techniken

- Sit Together, Whole Team, Informative Workspace, Energized Work, Pair Programming, Stories, Weekly Cycle, Quarterly Cycle, Slack, Ten-Minute Build, Continuous Integration, Test-First Programming, Incremental Design



Techniken

- Sit Together, Whole Team, Informative Workspace, Energized Work, Pair Programming, Stories, Weekly Cycle, Quarterly Cycle, Slack, Ten-Minute Build, Continuous Integration, **Test-First Programming, Incremental Design**
- Zusammengefasst: Test-Driven Development



Prinzipien

- Humanity, Economics, Mutual Benefit, Self-Similarity, Improvement, Diversity, Reflection, Flow, Opportunity, Redundancy, Failure, Quality, Baby Steps, Accepted Responsibility



Prinzipien

- Humanity, Economics, Mutual Benefit, Self-Similarity, Improvement, Diversity, Reflection, Flow, Opportunity, Redundancy, Failure, Quality, Baby Steps, Accepted Responsibility



Werte

- Communication
- Simplicity
- Feedback
- Courage
- Respect
- Others (Safety, Predictability, Quality-of-Life, ...)



Werte

- Communication
- **Simplicity**
- Feedback
- Courage
- Respect
- Others (Safety, Predictability, Quality-of-Life, ...)



Gegenüberstellung

- Java/Eclipse vs Ruby/Rails:
 - Test-Driven Development
 - Simplicity



Test-Driven Development

- Programmierzüge:
 - grün-rot: Neuen Test schreiben, der fehlschlägt
 - rot-grün: neuen Test zum Laufen bringen
 - grün-grün: Refactoring



TDD in Java/Eclipse

- Demo!



TDD in Java/Eclipse

- grün-rot: Code Generation: Nicht vorhandene Klassen/Methoden werden aus der ersten Verwendung generiert
- rot-grün: Code Completion: Vorhandene Klassen/Methoden werden kontextsensitiv angeboten
- grün-grün: Automatisiertes Refactoring



TDD in Ruby

- (Am Beispiel des Editors TextMate, mit dem Rails selbst implementiert wird)
- Demo!



TDD in Ruby

- grün-rot: keine Code Generation, nur Copy-Paste
- rot-grün: keine echte Code Completion, nur Word Completion (aka "Hippie Completion")
- grün-grün: keine Refactoring-Unterstützung, nur projektweites Suchen und Ersetzen



TDD in Ruby on Rails

- Starke Kopplung von Model- und Controllerschicht
- Eine Datenbank-Fixture für alle Tests
- Folge:
 - Viele Tests gehen gegen die Datenbank
 - Tests dadurch langsam, beschleunigen kostet Zeit
 - Oft entstehen Abhängigkeiten zwischen den Tests



TDD in Ruby on Rails

- Refactoring wird durch die Conventions erschwert
- Beispiel Umbenennung einer Action:
Der Name der Action findet sich als
Methodenname im Controller (`def article...`), als
Teil vom Dateinamen in Views (`show.html.erb`), als
Symbol (`:article`) und in Strings ("`articles/article`")
- Andererseits:
durch Conventions auch weniger Refactoring



Test-Driven Development

- Java/Eclipse liegt vorn



Simplicity

- „the art of maximizing the amount of work not done“ (Principles behind the Agile Manifesto)



Beispiel: Admins

- Aufgabe: Finde alle Administratoren in users

- Java:

```
List<User> admins = new ArrayList<User>();  
for (User user : users) {  
    if (user.isAdmin()) admins.add(user);  
}
```



Beispiel: Admins

- Aufgabe: Finde alle Administratoren in users
- Ruby:
`users.find_all { |user| user.admin? }`
- In Ruby on Rails auch möglich:
`users.find_all &:admin?`



Beispiel: Namen

- Aufgabe: Kommaseparierte Liste aller Vornamen der User in users, sortiert, doppelte raus
- Ruby
`users.map { |user| user.name }.sort.uniq.join(',')`
- In Ruby on Rails auch möglich:
`users.map(&:name).sort.uniq.join(',')`
- Java: Nächste Folie



Beispiel: Namen

```
List<String> names = new ArrayList<String>();
for (User user : users) names.add(user.getFirstName());
Collections.sort(names);
String lastName = null;
for (Iterator<String> it = names.iterator(); it.hasNext();) {
    String name = it.next();
    if (name.equals(lastName)) it.remove();
    lastName = name;
}
StringBuffer join = new StringBuffer();
for (Iterator<String> it = names.iterator(); it.hasNext();) {
    join.append(it.next());
    if (it.hasNext()) join.append(",");
}
String result = join.toString();
```



Beispiel: Zeit/Datum

- Ruby on Rails:
`yesterday = 1.day.ago`
`tomorrow = 1.day.from_now`
- Oder:
`yesterday = Time.now.yesterday`
`tomorrow = Time.now.tomorrow`



Beispiel: ActiveRecord

```
class User < ActiveRecord::Base
  validates_presence_of :login, :email
  validates_uniqueness_of :login, :email,
    :case_sensitive => false
  before_save :encrypt_password
  ...
end
```



Simplicity

- Ruby/Rails liegt vorn



XP mit Ruby on Rails

- Nachteile bei Test-Driven Development (Probleme bei Code Generation, Code Completion, Refactoring, Testgeschwindigkeit)
- Vorteile bei Simplicity



Wer ist der Gewinner?

- Beide Sprachen haben Vorteile, u.a.
- Java hat eine hervorragende Tool-Unterstützung
- Ruby ermöglicht, mit wenig Code viel zu erreichen
- Eigentlich möchte ich auf keinen der Vorteile verzichten ;)
- Mein Gewinner für XP: Ruby



Ausblick

- Annäherung
- Bewegung auf beiden Seiten:
 - Java 7: invokevirtual, Closures
 - Ruby: Verbesserung der IDEs – 3rdRails, Eclipse DLTK, Netbeans, IntelliJ IDEA, ...



Vielen Dank!

- Fragen?
- Entweder jetzt, oder an business@tammofreese.de