Testgetriebene Web-Entwicklung mit Ruby on Rails

Thomas Baustert

www.b-simple.de

Agenda

- Rails Einführung
- Test-Unterstützung in Rails
- Testgetriebene Web-Entwicklung mit Rails
- Live Demo

Eindruck von Rails und dessen Unterstützung für umfangreich getestete Web-Anwendungen

Ruby on Rails („Ruby auf Schienen“)

Ruby on Rails ist ein Framework für die Entwicklung von datenbankbasierten Web-Anwendungen in Ruby.

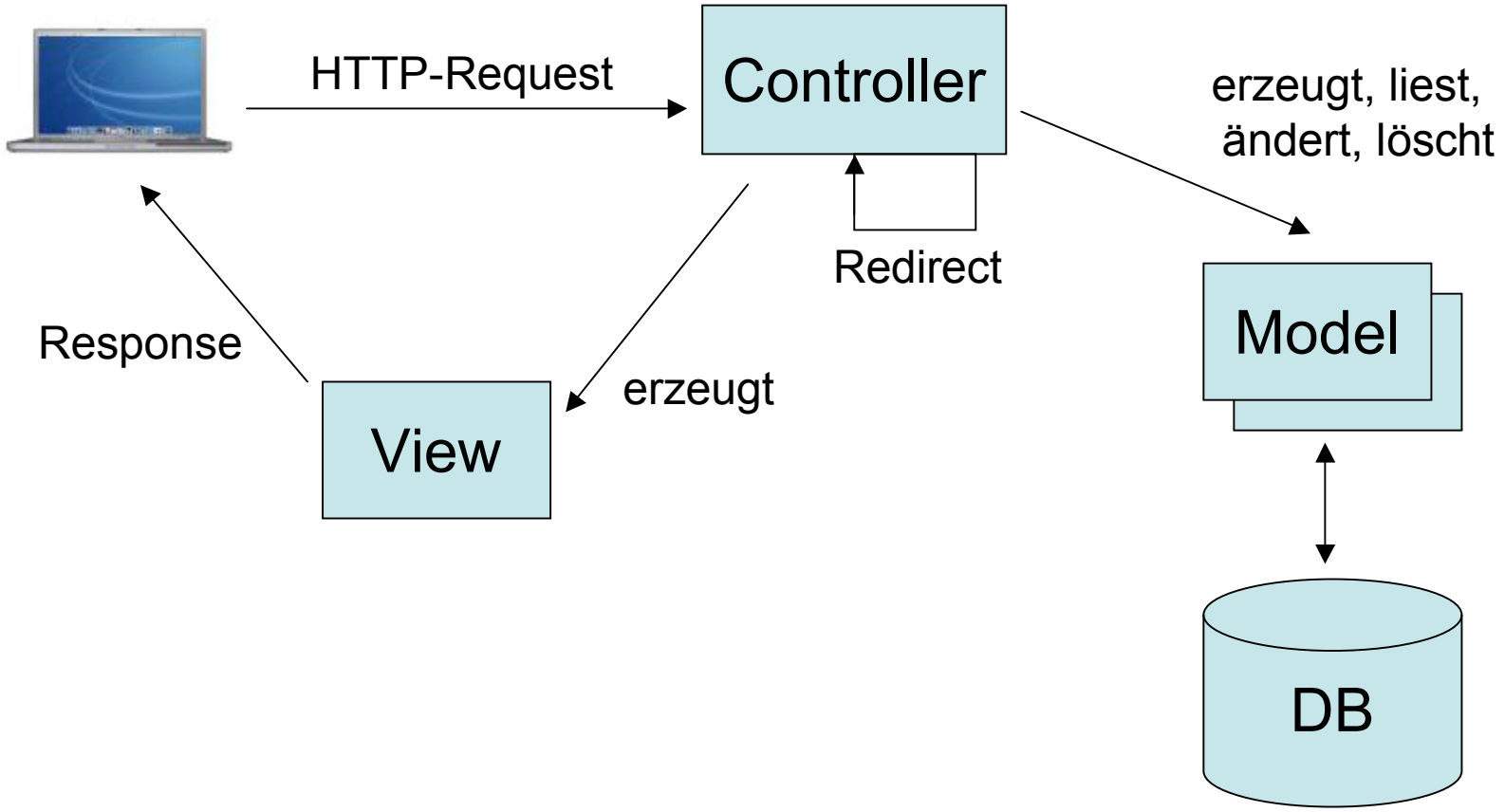
- MVC-Architektur
- Konvention statt Konfiguration
- DRY-Prinzip
- Weniger Code
- Extrahiert
- Unmittelbares Feedback
- Hohe Testbarkeit
- Ruby
- Open Source
- David Heinemeier Hansson

Ruby

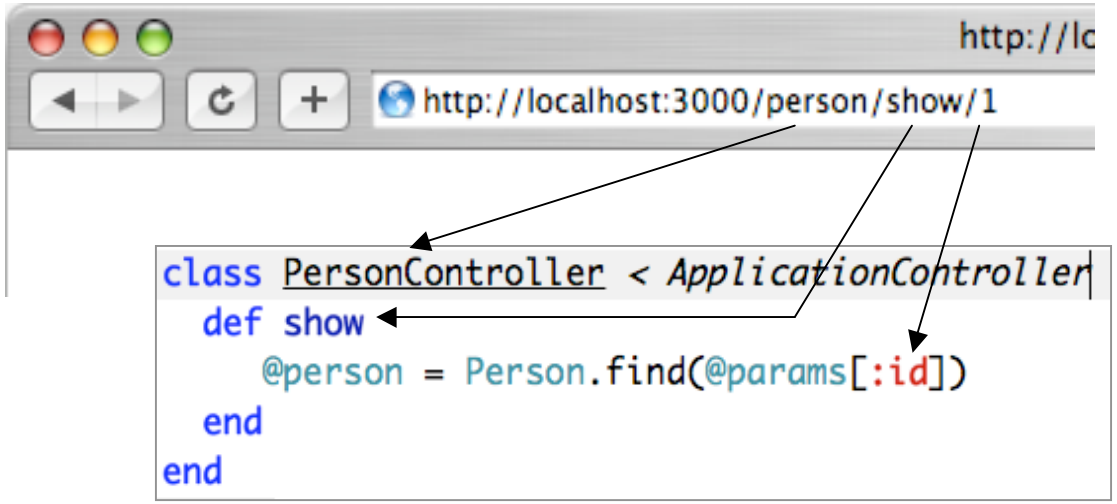
- Dynamisch typisierte Sprache
- Rein objektorientiert
- Einflüsse von Perl, Smalltalk, LISP, u.a.
- Ca. 1995 veröffentlicht
- Yukihiro Matsumoto („Matz“)
- Garbage Collection, Ausnahmen, Reguläre Ausdrücke, Introspektion, Erweiterung von Klassen, Code-Blöcke, usw.

```
class RubyClass < SuperClass
  include RubyModul

  def method(param)
    @instance_var = param
  end

  def RubyClass.class_method
    ...
  end
end

obj = RubyClass.new
obj.method("Ruby")
```

Rails MVC


HTTP-Request
Controller
erzeugt, liest,
ändert, löscht
Redirect
Model
Response
View
erzeugt
DB

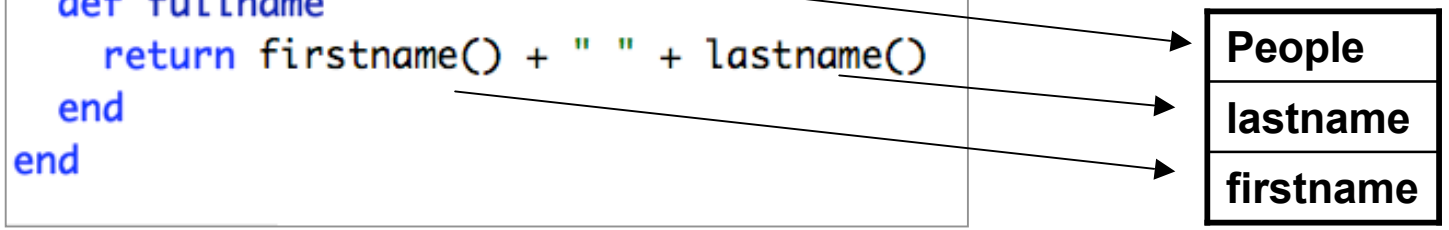
Controller



- Steuern Kontrollfluss (Action)
- Bearbeiten HTTP-Requests
- Aufruf per Reflection => keine Konfiguration
- Erzeugen Views
- View-Daten über Instanzvariablen

Modelle

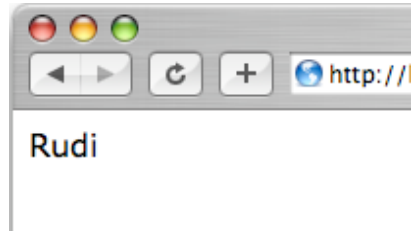
```
class Person < ActiveRecord::Base

  def fullname
    return firstname() + " " + lastname()
  end
end
```



- Domainobjekte
- Active Record Pattern
- Tabellenname = plural(Modellname)
- Beziehen Attribute aus Datenbanktabelle (DRY)
- Rails erzeugt dynamisch Getter und Setter
- Modell-Relationen (1:1, 1:N, N:M)
- Vererbung (Single Table Inheritance)

Views

```
<div id="show">
    <span class="person">
        <%= @person.name %>
    </span>
</div>
```




```
class PersonController < ApplicationController
  def show
    @person = Person.find(@params[:id])
  end
end
```

- Präsentation von Modellen in HTML
- Daten über Instanzvariablen
- Eingebetteter Ruby-Code (keine neue Sprache)

Testgetriebene Softwareentwicklung

- Entwurfs- und Programmiermethode
- Tests treiben die Softwareentwicklung (Design)
- TDD-Kreislauf:
  1. **Test**
     Schreibe einen fehlschlagenden Test
  2. **Code**
     Schreibe gerade soviel Code, dass der neue Test und alle vorhandenen Tests fehlerfrei ausgeführt werden
  3. **Refactoring**
     Entferne Code Duplizierung und andere „Code Smells“

→ Software wird änderbar und lebt lange

Unit Tests

- Basis für testgetriebene Softwareentwicklung
- Programmierte, automatisch ausführbare Tests
- Selbstständige Überprüfung der Ergebnisse
- Voraussetzung für Änderungen (z.B. Refactoring)
- Dynamische Sprachen
  - Zusätzliche Fehlerquelle (?)
  - Tests „notwendiger“ (?)
- Ruby Test::Unit

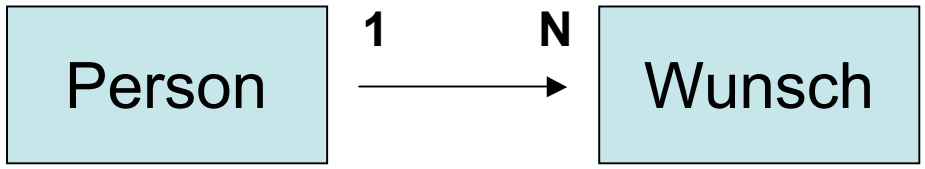
```ruby
class MoneyTest < Test::Unit::TestCase
  def setup ... end

  def teardown ... end

  def test_init
    money = Money.new(100.0, 'EUR')
    assert_not_nil money
    assert_in_delta 100.0, money.amount
    assert_equal 'EUR', money.currency
  end
end
```

Test-Unterstützung in Rails

- Web-spezifische Erweiterung von Test::Unit
  - assert_redirected_to
  - assert_template
  - assert_tag
  - assigns, session, cookie
  - u.a.
- Test-Unterstützung:
  - Modelle, Controller, Views
  - Mocks
  - *Action Mailer*
  - *Action Web Services*

Demo „Wunschzettel“

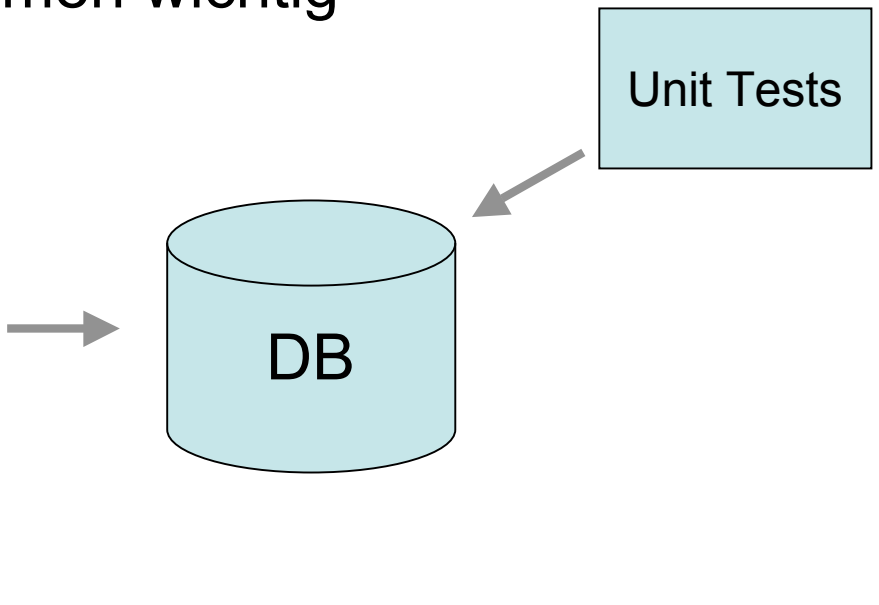
- Papa: Socken und Krawatte
- Mama: Lockenwickler und Schal
- Bruder: Computer und Freundin
- Schwester: ...



Testen und Fixtures

- Separate Testdatenbank
- Tabellen werden vor jedem Test automatisch gefüllt
- Aussagekräftige Namen wichtig

```
# Fixture people.yml
papa:
  id: 1
  name: Rudi
  birthday: 1960-05-10

mama:
  id: 2
  name: Gundula
  birthday: 1962-03-19
```



Modelle testen

- Neues Modell + Assoziation

```ruby
def test_save
  person = Person.new
  assert !person.save, "kein speicher ohne namen"
  person.name = people(:papa).name
  assert person.save
end

def test_has_wishes
  person = Person.new
  assert_equal 0, person.wishes.length
end
```

```ruby
# Fixture people.yml
papa:
  id: 1
  name: Rudi
```

Controller testen

- Action prüfen
- Zuweisung von Daten an Instanzvariablen prüfen
- Web-Server muss nicht laufen

```
def test_show
  get :show, { :id => 1 }
  assert_template "show"
  assert_not_nil assigns(:person)
end
```

```
class PersonController < ApplicationController
  def show
    @person = Person.find(@params[:id])
  end
end
```

Views testen

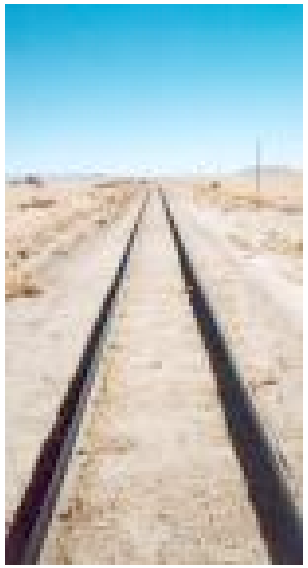
- HTML Code prüfen (Tags, Attribute, Daten, ...)

```
def test_show
  get :show, { :id => 1 }
  ...
  assert_tag :tag => "div", :attributes => { :id => "show"},
    :descendant => { :tag => "span" ,
      :attributes => { :class => "person"}, :child => /Rudi/ }
end
```

```
<div id="show">
    <span class="person">
        <%= @person.name %>
    </span>
</div>
```

Und sonst?

- AJAX-Unterstützung (Asynchronous JavaScript And XML)
  - Hoch interaktive Applikationen ähnlich Rich Client
  - Drag & Drop, Visuelle Effekte
- Action Mailer
  - E-Mail versenden und empfangen
- Action Web Services
  - Anbindung anderer Systeme

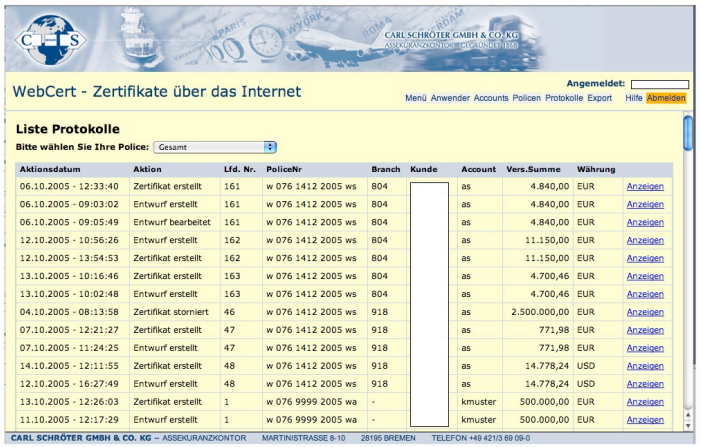


Einsatz pro/contra

- Für einen Großteil von Web-Anwendungen
- Einfaches Datenmodell
- Keine verteilte Transaktion
- Apache/LightTPD und FastCgi
- Performanz?

- Anbindung externer Systeme (Host, SAP)
- Rich Client Anbindung
- Fehlende Tools, APIs (z.B. PDF-Server)
- Keine Integration möglich oder nur Neuimplementierung?
- Performanz?

Unsere Erfahrungen



- Rails seit Januar 2005
- csWebCert:
  - Konsequent testgetrieben
  - ca. 400 Tests
    ca. 3000 Assertions
  - Test-Code > Anwendungs-Code
- Änderungen sicherer und schneller durchführbar
- Extrem geringe Fehlerrate in Akzeptanztests
- Produktion: Apache/FastCgi

Anwendungen:

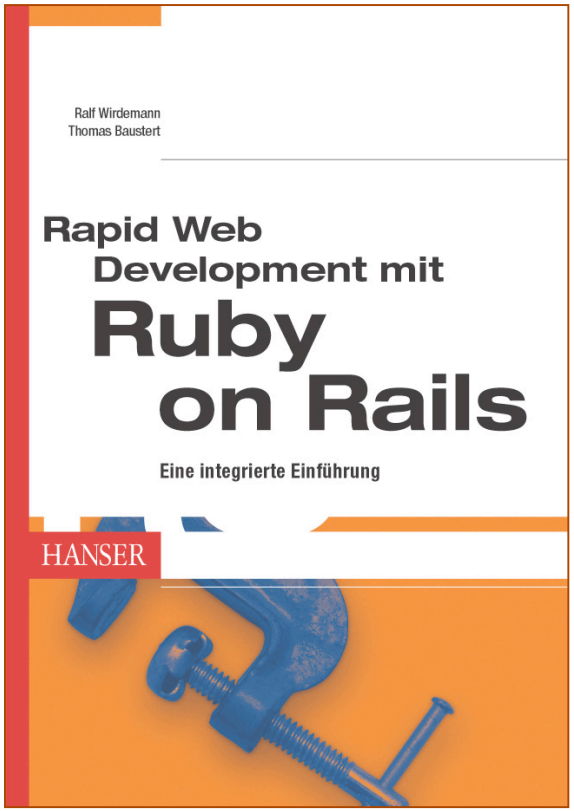
- Basecamp
- bellybutton
- Odeo
- ...

Fazit

- Schneller, kostengünstiger, früherer ROI
- Hohe Testbarkeit, Optimale Test-Unterstützung
- Sichere Nutzung dynamischer Sprachen durch konsequentes Testen
- Auf Änderungen vorbereitet
- Wartbarkeit und Langlebigkeit
- Großteil aller Web-Anwendungen können mit Rails entwickelt werden

Quellen

- www.rubyonrails.org
- www.rubyonrails-ug.de
- www.rubyonrails.de
- www.ruby-lang.org/en



Hanser 01/2006

www.rapidwebdevelopment.de

Rails.

Is it love?