

Ruby on Rails

Chemnitzer Linuxtage

2007

Peter Dickten
www.dcs-fuerth.de

Agenda

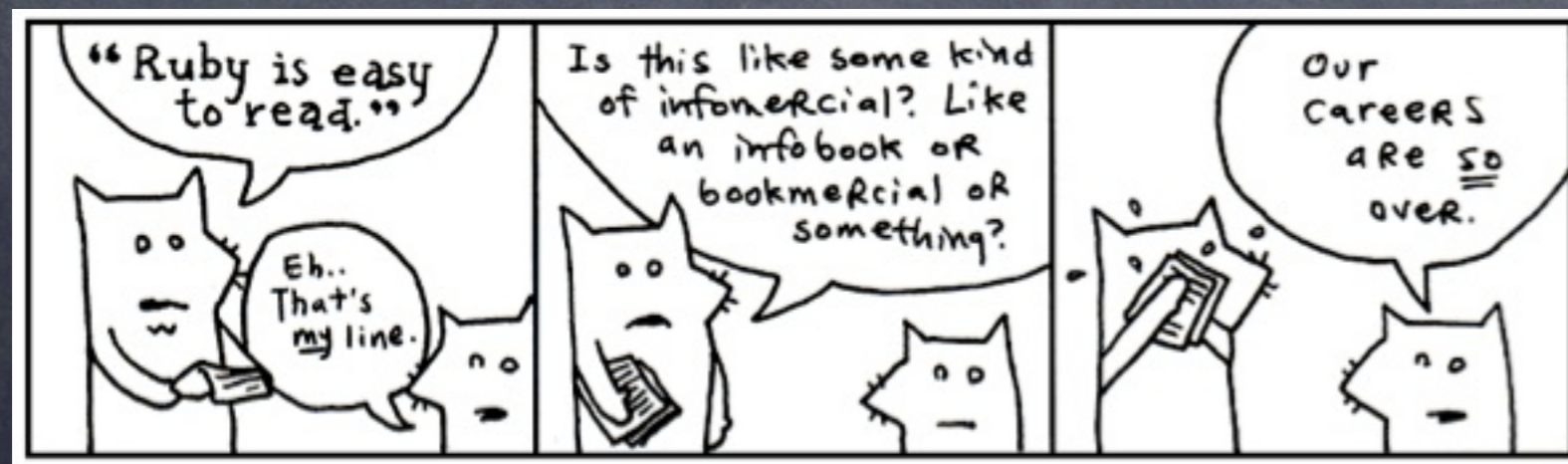
- Geschichte
- Warum Ruby on Rails (RoR)?
- Beispielanwendung (Adreßverwaltung)

Ruby...

- 1995 von Yukihiro Matsumoto entwickelt
- Skriptsprache (=> IRB shell)
- strikt objektorientiert
- Einflüsse von Scheme, Python, Objective C
- erst nur geringe Verbreitung

why's (poignant) guide to ruby

=> <http://poignantguide.net>



Ruby on Rails (RoR)

- Webframework
- David Heinemeier Hansson (37signals.com)
- entstanden durch Extraktion aus bestehenden Projekten
- Reduktion auf das Wesentliche ("37signals esthetics")

The Top Five Technologies You Need to Know About in '07

Five Hot Technologies for 2007

1. Ruby on Rails

Faster, easier Web development

2. NAND drives

Bye-bye, HDD?

3. Ultra-Wideband

200x personal-area networking

4. Hosted hardware

Supercomputing for the masses

5. Advanced CPU architectures

Penryn, Fusion and more

warum so beliebt?

- look ma (almost) no code
- DRY-Prinzip (don't repeat yourself)
- MVC ohne 3721 XML-Dateien
- ActiveRecord-Pattern
- (...)

dry

In RoR muss ein Sachverhalt genau einmal programmiert bzw. beschrieben werden.

Dieser Sachverhalt ist damit festgeschrieben und dadurch für alle anderen Teil der Anwendung bereits erklärt.

Gegenbeispiel: J2EE (speziell EJB)

MVC ohne XML-Orgien

Klassische MVC-Frameworks (Struts) verknüpfen die Schichten durch XML-Dateien.

Fehler in diesen Dateien sind schwer zu finden.

=> convention over configuration

Active Record

- lesender und schreibender Zugriff auf Datenbank ohne SQL
- Datenbank kann leicht gewechselt werden
- Mapping zwischen Objekten und Datenbanktabellen ohne XML-Zwischenschicht (Hibernate)
- Das Ganze klappt auch wunderbar mit Fremdschlüsselrelationen

benötigt:

- Ruby 1.8.x (oft in Linux-Distrib. enthalten)
(oder: `apt-get install ruby irb rdoc`)
- Ruby on Rails: <http://www.rubyonrails.org/down>
(`gem install rails --include-dependencies`)
- Texteditor oder IDE (RadRails, Komondo, Locomotive, [...])
- evtl. DB-Bibliothek (z.B. für Oracle)

Anwendung erstellen:

- Anwendung(srahmen)
- Datenbanktabelle(n), DB-Konfiguration
- Klasse(n)
- Oberflächenprototyp (scaffold)

Anwendungsrahmen erstellen

```
rails adressliste
```

- Erstellt die Verzeichnisstruktur der Anwendung

Verzeichnisstruktur

Verzeichnisname	Bedeutung
• Adressliste	Projektverzeichnis
• app	
• controllers	Steuerlogik
• addresses_controller.rb	
• application.rb	
• helpers	Hilfsklassen und -methoden
• addresses_helper.rb	
• application_helper.rb	
• models	Modellklassen
• address.rb	
• views	Sichten
• addresses	
• _form.rhtml	Teilmaske, verwendet von new.rhtml und edit.rhtml
• edit.rhtml	Bearbeiten
• list.rhtml	Listenanzeige
• new.rhtml	Neueingabe
• show.rhtml	Einzelanzeige
• layouts	
• addresses.rhtml	Allgemeines Layout
• config	Konfigurationsdaten
• database.yml	Datenbankkonfiguration
• public	Statische HTML-Seiten
• index.html	RoR-Startseite ("welcome aboard")
• test	Testfälle, Mockobjekte

Datenbanktabelle(n) anlegen

```
CREATE TABLE `adressliste`.`addresses` (
```

```
  `id` INT NOT NULL AUTO_INCREMENT,
```

```
  `NACHNAME` VARCHAR(50), `VORNAME` VARCHAR(50),
```

```
  `ANREDE` VARCHAR(15), `STRASSE` VARCHAR(40),
```

```
  `PLZ` VARCHAR(5), `ORT` VARCHAR(40),
```

```
  `TELEFON` VARCHAR(20),
```

```
  PRIMARY KEY(`id`)
```

```
);
```

Konvention:
Primär-
schlüssel
heißt id

hier
verwendet:
MySQL

Datenbankkonfigurationen

- befindet sich in `config/database.yml`
- 3 verschiedene Konfigurationen:
 - `development` `<--` Vorgabe bei neuem Projekt
 - `test`
 - `production`

Beispielkonfiguration

development:

adapter: mysql

database: adressliste

username: benutzername

password: kennwort

host: localhost

Diese
Attribute sind
abhängig vom
DBMS

Klasse zur Tabelle erstellen

Unterscheide:

- Datenbanktabelle Plural
- Klasse: Singular

```
ruby script/generate model Address
```

... erzeugt u.a. in app/models/ ...

```
class Address < ActiveRecord::Base  
  
end
```

Ruby als Skriptsprache

Ruby-Shell: `ruby script/console`

```
adr=Address.new
```

```
adr.vorname='Manfred'
```

```
adr.vorname='Mustermann'
```

```
adr.save
```

Weboberfläche erstellen "scaffolding"

```
ruby script/generate scaffold Address
```

```
exists (...)
```

```
create app/views/addresses
```

```
create app/views/addresses/_form.rhtml
```

```
(...)
```

erzeugt eine elementare Weboberfläche für
unsere Klasse

Webserver ?

Um die Anwendung anzusehen brauchen wir einen laufenden Webserver

- WEBrick ist eingebaut, aber nur für Entwicklungszwecke gedacht
- lighttpd für Produktionssysteme

WEBrick starten

```
ruby script/server
```

=> Booting WEBrick...

=> Rails application started on http://0.0.0.0:3000



Standard-
Port 3000

Anderen Port verwenden:

```
ruby script/server -p 80
```

Aufruf der Anwendung

Allgemein:

<http://ipadresse:port/controller>

hier:

<http://127.0.0.1:3000/addresses>

URL-Aufbau in RoR

http://ipadresse:portnummer/controller/aktion/id

http://127.0.0.1:3000/addresses/edit/1

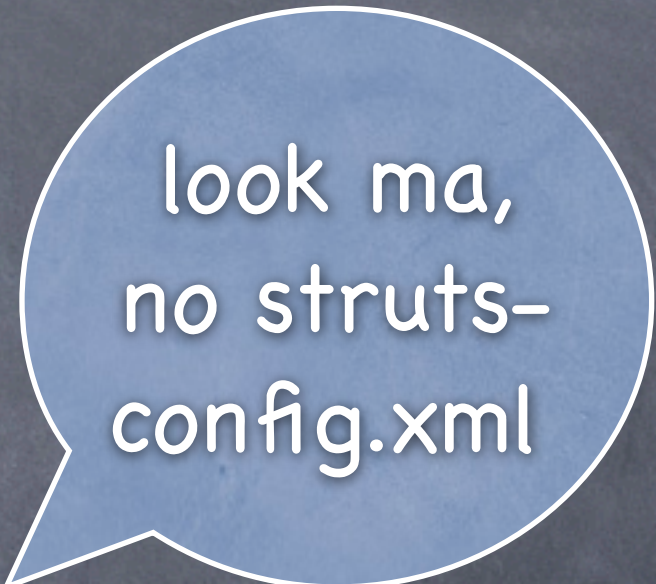
controller -> beim Scaffolding erzeugt

aktion -> was der Controller macht: Show, Edit, Destroy, New (Index ist Vorgabe)

id -> wenn ein einzelner Datensatz betroffen ist (z.B. beim Löschen)

(Vorgabe-)Aufrufkette

`http://ip:port/addresses/edit/42`



look ma,
no struts-
config.xml

- ruft den in `app/controller` in den `addresses_controller` die Methode `edit` auf und übergibt ggf. den Parameter `42`
- im Anschluß wird in `/app/views/addresses` der Teilview `edit.rhtml` aufgerufen und das Ergebnis in `/app/views/layouts/addresses.html` eingebettet

Dateitypen:

- .rb – Rubycode (Klassen)
- .rhtml – (partielles) HTML mit eingebettetem Rubycode (analog PHP, JSP ...)
- .html – Reines HTML
- .yml – Konfigurationsdateien

eine Controller-Aktion im Tiefflug

```
def list  
  @address_pages, @addresses =  
    paginate :addresses, :per_page => 10  
end
```

@ - Variablen sind außerhalb sichtbar

() sind optional bei Methodenaufrufen

:xyz sind benannte Parameter (wie ObjectiveC)

... mehr unter <http://www.ruby-lang.org/de/>

ein View ...

```
<% for address in @addresses %>
```

```
  <% for column in Address.content_columns %>
```

```
    <%=h address.send(column.name) %>
```

<%= %> Auswertung eines Ausdrucks

h (...) HTML-Encoding - hilft gegen code injecton

Aktionen durch Links aufrufen

```
<%= link_to 'Zeige', :action => 'show', :id => address %>
```

ruft beim Anklicken im gleichen Controller die Aktion **show** auf und übergibt den **aktuellen Datensatz** als **id**

weitere interessante Parameter:

- `:confirm => 'Sicher?'`
- `:method => :post`

Blättern

verwaltet den sichtbaren Teil

```
<%= link_to 'Next page',  
{ :page => @address_pages.current.next }  
if @address_pages.current.next %>
```

keine Aktion => Selbstaufruf (gleiche Aktion)

if ... ist eine Bedingung für die Anzeige des Links

Was ist gut an RoR?

- Sehr schnelle Entwicklung von datenbankbasierten Web-Anwendungen
- Konsequente Implementierung des MVC-Pattern
- zahlreiche Bibliotheken und Tutorials zu fast jedem Aspekt der RoR-Entwicklung.

Was ist nicht so schön?

- Eleganz nur wenn man sich an die Konventionen halten kann, problematisch bei bestehenden Tabellenstrukturen (Views?)
- Um nichttriviale Projekte mit RoR zu realisieren, muss man auf jeden Fall Ruby lernen; Grails als Alternative für "Javanesen"

Das Letzte ...

- Updates unter

<http://www.dcs-fuerth.de/linuxtag2007>

- Fragen, Probleme?

peter.dickten@dcf-fuerth.de

JOB, PRAKTIKUM ODER DIPLOMARBEIT in den
Bereichen Ruby on Rails, J2EE oder .NET gesucht?

jobs@dcf-fuerth.de