

Ruby on Rails

Alternative zur Web-Entwicklung mit Java?

Thomas Baustert

Ralf Wirdemann

www.b-simple.de

Überblick

- Was ist Ruby on Rails?
 - Weblog Demo
- Rails Komponenten
 - Controller, Modelle, Views
 - Demos
- Unsere Erfahrungen mit Rails
- Was können wir als Java Entwickler lernen?

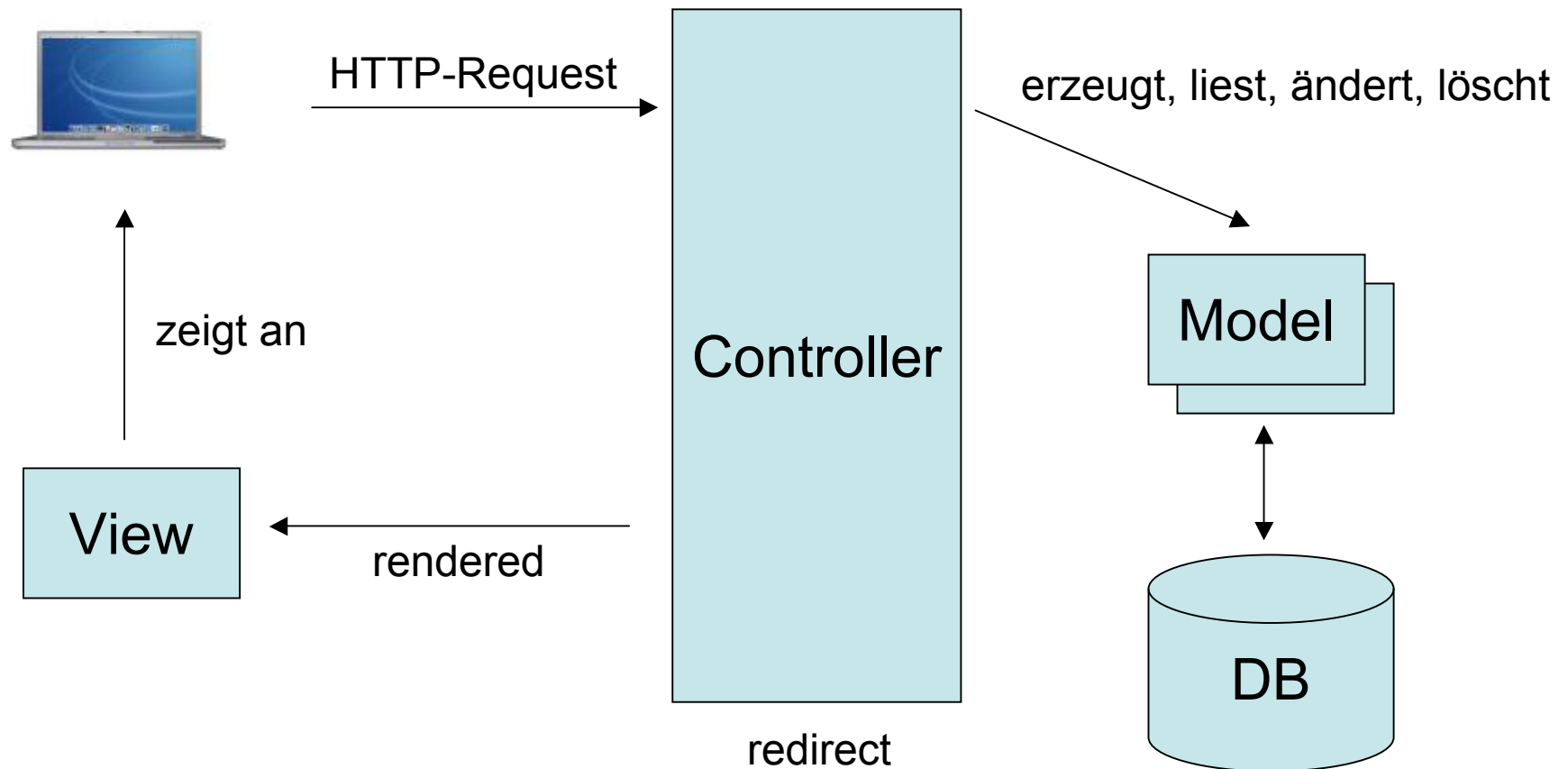
Ruby on Rails

- Ruby on Rails ist ein Framework für die Entwicklung von Web-Applikationen in Ruby:
 - Ruby
 - MVC Architektur
 - DRY (don't repeat yourself) Prinzip
 - Konvention statt Konfiguration
 - Extrahiert
 - Hohe Testbarkeit in MVC
 - Open Source, David Heinemeier Hansson

Demo: Weblog #1



Rails Architektur



Controller

```
class PostController < ApplicationController
  def show
    ...
  end

  def list
    ...
  end
end
```

- steuern Kontrollfluss
- bearbeiten HTTP-Requests
- gruppieren Actions
- beliebig viele Controller
- seine Konfiguration (<http://localhost/post/show>)

Controller – View Daten

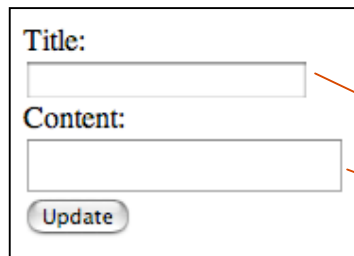
```
class PostController
  < ApplicationController

  def show
    @post = Post.new
    @post.title = "JFS"
    @post.content = "... "
  end
end
```

```
<html>
  <h1>Weblog Posting</h1>
  Title:
  <%= @post.title %><br>
  Content:<br>
  <%= @post.content %><br>
</html>
```

- Actions rendern Views
- Datenaustausch über Instanzvariablen
- Implizites View-Rendering

Controller - Request Parameter



Title:

Content:

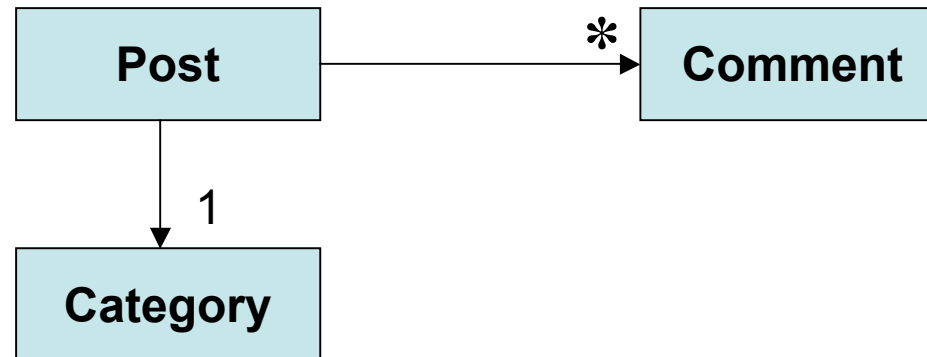
```
class PostController < ApplicationController  
  
  def update  
    title = @params["title"]  
    content = @params["content"]  
    ...  
    render "list"  
  end  
  
end
```

- @params Hash liefert Request-Parameter
- explizites View Rendering

Demo: Hello World

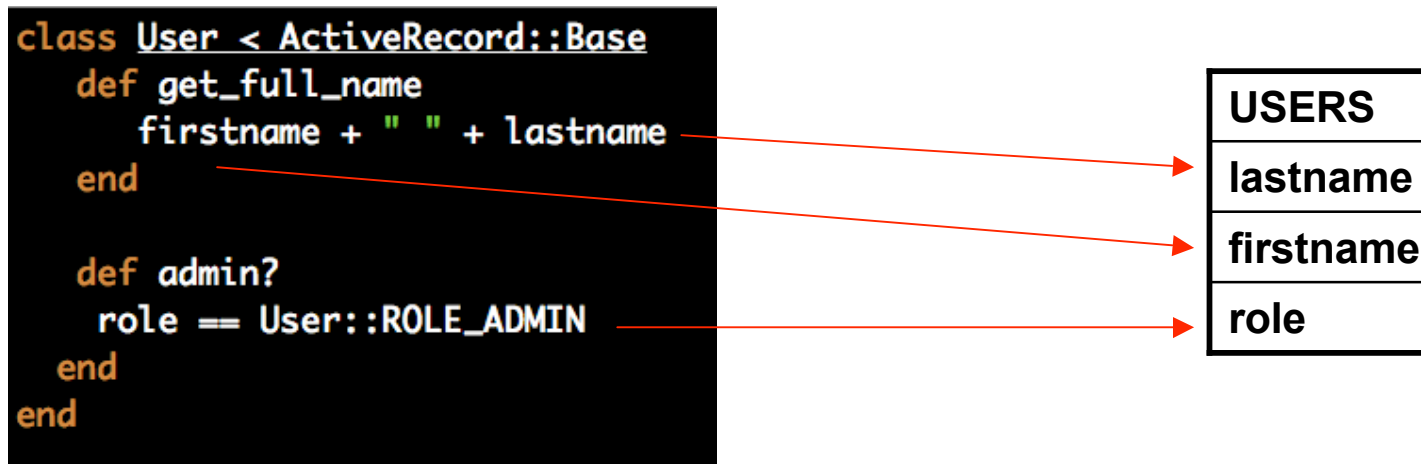


Rails Modelle



- Domainobjekte
- OR-Mapping
- wie POJOs (Hibernate) oder Entity Beans (EJB 2.1)
- Relationen
- Vererbung

OR Mapping

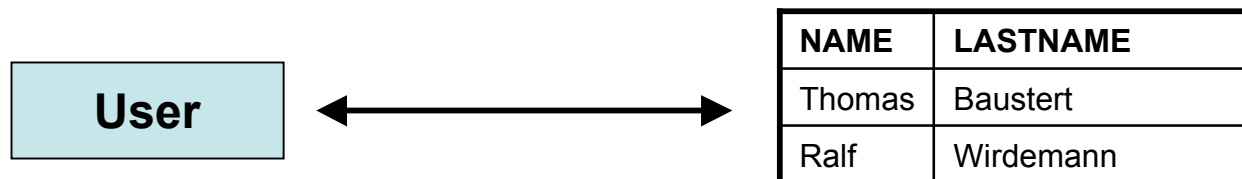


- erben von ActiveRecord::Base
- Klasse = Tabelle, Instanzen = Einträge
- Tabellename wird „geraten“
- Rails generiert Attribute, Getter und Setter (DRY)

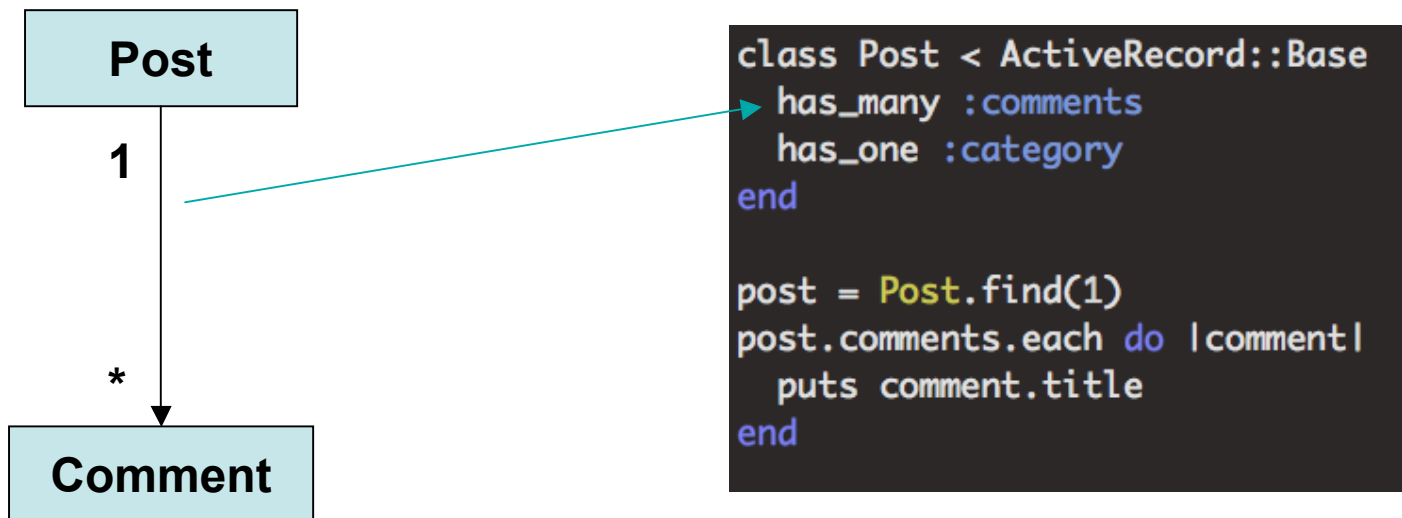
CRUD (create, read, update, delete)

```
class User < ActiveRecord::Base
end

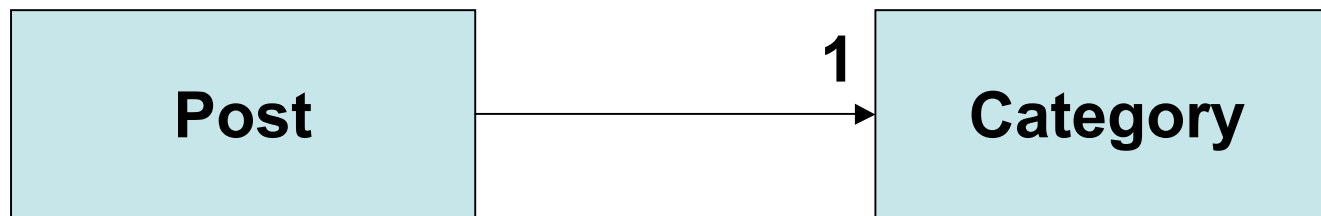
user = User.new(:name => „Thomas“)
user.save // INSERT
user.lastname = „Baustert“
user.save // UPDATE
user.destroy // DELETE
```



Modellassoziationen



Demo: Weblog #2 - Model



POSTS	
id	int
title	varchar
content	text
category_id	int

A red arrow points from the **category_id** field in the **POSTS** table to the **id** field in the **CATEGORIES** table, indicating a foreign key relationship.

CATEGORIES	
id	int
name	Varchar

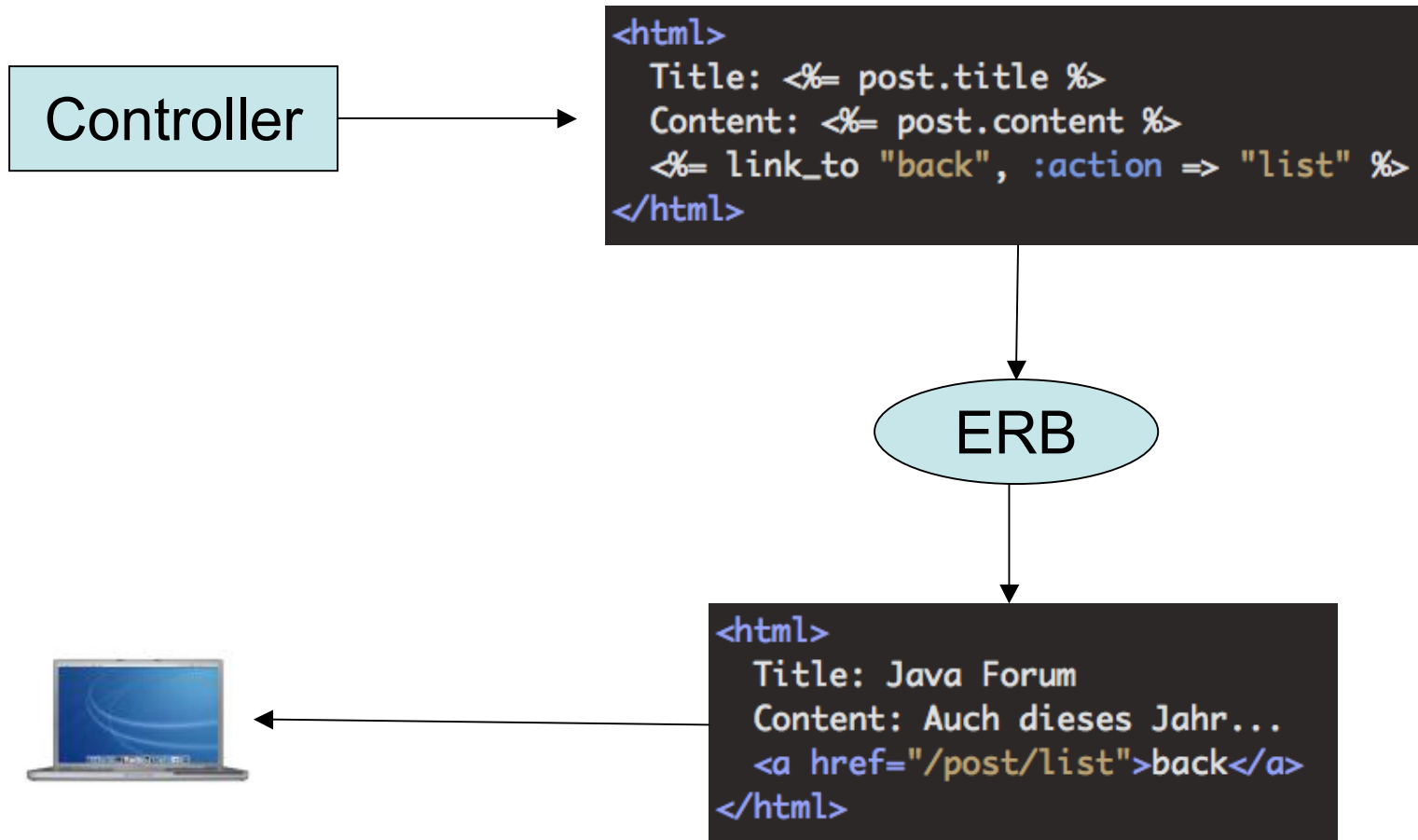
Rails Views

```
<html>
  <h1>Weblog Posting</h1>
  Title: <%= @post.title %><br>
  Content:<br>
  <%= @post.content %><br>

  <% @post.comments.each do |comment| %>
    ==> Re: <%= comment.title %>
  <% end %>
</html>
```

- Präsentation von Modellen in HTML
- eingebetteter Ruby-Code (keine neue Sprache)
- Helper Methoden und Form-Tags
- Daten über Instanzvariablen

RHTML



Unit Testen

- Dynamische Sprachen und Testen
 - zusätzliche Fehlerquelle
 - automatisiertes Testen wird „notwendiger“
 - Unit Tests finden Typfehler „nebenbei“
- Test::Unit
- Rails erweitert Test::Unit
 - spezielle Assertions
 - assert_response, assert_template
 - assert_not_nil assigns(:post)
 - assert_tag
- Test von Modellen, Controller und View

Und sonst?

- AJAX (Asynchronous JavaScript And XML)
 - Prototype Library (Calls, DOM, Visual Effects)
 - Hilfsmethoden, z.B. `link_to_remote()`
- Web Services (ActionWebService)
 - SOAP, XML-RPC
 - API Definition
- Mail-API (ActionMailer)
 - Mail Templates via RHTML Templates
 - Mails senden und empfangen
- Active Support, Filter, Validierung, Pagination, usw.

Rails Philosophie

- Konventionen über Konfiguration
 - so gut wie keine Konfigurationsdateien
 - viel Reflection und „Qualified Guessing“
- DRY
- Unmittelbares Feedback
- Weniger Code
- Wirkt homogen
- Best Practices durch Aufforderung
 - automatisch generierte Unit-Tests

Projekt

- 2 Personen, 3 Monate, nur in der Freizeit
- 16 Modelle, 13 Controller
- 300 Tests mit mehr als 1500 Assertions
- extrem geringe Fehlerrate

Andere

- Basecamp (www.basecamphq.com)
- Backpackit (www.backpackit.com)
- BellyButton (www.bellybutton.de)
- SnowDevil (www.snowdevil.ca)

Unsere Erfahrungen mit Rails

- Spaß bei der Arbeit
- Wir haben Ruby und Rails gleichzeitig gelernt
- Produktiv
- Testgetrieben
- Ruby und Rails Community hilfreich
- Wir sind so begeistert, dass wir
 - weiter mit Rails arbeiten möchten
 - ein deutschsprachiges Rails-Buch schreiben
- TDD und dynamische Sprache = Top Team
- Rails in „Werkzeugkoffer“

Was können wir als Java-Entwickler lernen?

- Konvention statt Konfiguration
- Keep DRY
- TDD auch in Java
- Codegenerierung
- Sei einfach!

Alternative zu Java?

- Für einen Großteil von Web-Anwendungen
- Domain-Modell entspricht DB-Modell
- Anbindung externer Systeme (SAP?)
- Fehlende Tools, APIs (z.B. PDF-Server)
- Anwendung basiert auf Java, Integration möglich oder nur Neuimplementierung?

Ressourcen

- www.rubyonrails.org
- www.b-simple.de
 - J2EE/Spring/Rails
 - Projekte
 - Coaching
 - Workshop
 - Schulungen