



Ruby on Rails

Ein Web-Framework wird erwachsen

Patrick Lenz, limited overload GmbH
Open Source Meets Business 2007, Nürnberg

Was Rails ist.

Was Rails ist.

- “Full-Stack” Web Application Framework
- Vollständig in Ruby implementiert
- Objektorientiert
- Open Source
- Web 2.0 - klar!

Was Rails kann.

Was Rails kann.

- Produktivität steigern
- Kosten reduzieren
- Zeit sparen
- Qualität verbessern
- Glücklich machen

Mehr Details.

Mehr Details.

- Convention over Configuration
- Opinionated Software
- Beautiful Code
- Don't Repeat Yourself (DRY)
- Automated Testing
- Agile Development with Small Teams

Beautiful Code

Model

```
1 class Gallery < ActiveRecord::Base
2   has_many :gallery_photos
3   belongs_to :user
4   validates_presence_of :title
5 end
```

Controller

```
1 class GalleryPhotosController < ApplicationController
2   def show
3     @gallery = Gallery.find(params[:gallery_id])
4   end
5 end
```

View

```
1 <!-- show.rhtml -->
2 <h1><%= @gallery.title %></h1>
3
4 <div id="photos">
5   <%= render :partial => 'photo', :collection => @gallery_photos %>
6 </div>
7
8 <!-- _photo.rhtml -->
9 <%= image_tag photo.filename %>
10 <%= image_tag photo.filename %>
11 <!-- _photo.rhtml -->
```

AJAX

- Gleicher Aufwand ob Links/Forms mit oder ohne AJAX-Funktionalität erstellt werden sollen
- Sogar gleicher Aufwand wenn der gleiche Aufruf beides können soll

```
1 <!-- HTTP POST -->
2 <% form_for :comment, :url => comments_path do |f| %>
3   <%= f.text_area :body %>
4 <% end %>
5
6 <!-- AJAX Submit // HTTP POST Fallback -->
7 <% form_remote_for :comment, :url => comments_path do |f| %>
8   <%= f.text_area :body %>
9 <% end %>
```

AJAX

- Gleiche Controller Action für AJAX und nicht-AJAX Operationen, keine Logik-Wiederholung
- Komplexe DOM Operationen und Visual Effects in Ruby Code definieren (RJS Templates)

```
1 class CommentsController < ApplicationController
2   def create
3     @comment = Comment.create(params[:comment])
4
5     respond_to do |wants|
6       wants.html { redirect_to comment_path(@comment) }
7       wants.js
8     end
9   end
10 end
11
12 # create.rjs
13 page.insert_html :bottom, 'comments_list', :partial => @comment
14 page.visual_effect :highlight, dom_id(@comment)
```

Don't Repeat Yourself

- Definition des Datenbankschema bereits in der Datenbank - warum also in Rails wiederholen?

```
1 class Gallery < ActiveRecord::Base
2 end
3
4 # script/console
5
6 g = Gallery.new           # => #<Gallery:..>
7 g.title = '30up party at Cocoon Club Frankfurt' # => "30up party at Cocoon Club Frankfurt"
8 g.save                   # => true
9
10 g.to_yaml               # =>
11
12 --- !ruby/object:Gallery
13 attributes:
14   id: "2"
15   created_at: 2007-01-21 11:41:55
16   updated_at: 2007-01-21 11:41:55
17   title: 30UP Party
```

```
18 # => #<Gallery:0x12345678>
19 # => #<ActiveRecord::ConnectionAdapters::SQLite3Adapter:0x12345678>
```

```
20 # => #<ActiveRecord::ConnectionAdapters::SQLite3Adapter:0x12345678>
```

```
21 # => #<ActiveRecord::ConnectionAdapters::SQLite3Adapter:0x12345678>
```

Don't Repeat Others

- Plugin Repository mit ca. 500 freien Plugins
- Integration neuer Funktionalität in eigene Applikation mit minimalem Aufwand

```
1 # script/plugin install acts_as_taggable
2
3 class Gallery < ActiveRecord::Base
4   acts_as_taggable
5 end
6
7 # script/console
8
9 g = Gallery.find :first           # => #<Gallery:..>
10 g.tag_with 'party 30up'         # => ["party", "30up"]
11 g.tag_list                       # => "party 30up"
12
13 result = Gallery.find_tagged_with 'party' # => [#<Gallery:..>]
14 result.first.title              # => "30up party at Cocoon Club Frankfurt"
15 result.first.tag_list           # => "party 30up"
```

```
12 result.first.tag_list           # => "party 30up"
13 result.first.title              # => "30up party at Cocoon Club Frankfurt"
14 result.first.tag_list           # => "party 30up"
```

Automated Testing

```
1 class UserTest < Test::Unit::TestCase
2   fixtures :users
3
4   def test_should_create_user
5     assert create_user.valid?
6   end
7
8   def test_should_require_login
9     u = create_user(:login => nil)
10    assert u.errors.on(:login)
11  end
12
13  protected
14
15  def create_user(options = {})
16    User.create({
17      :login => 'quire',
18      :password => 'quire'
19    }.merge(options))
20  end
21 end
```

- Programmatisches Testen von Applikation und Logik
- Test- und Behavior Driven Development
- Geringe Fehleranfälligkeit bei Code-Erweiterung durch “Kollateralschäden”

Rails skaliert.

Rails skaliert.

1.

Web
Application
MySQL

Rails skaliert.

1.



2.



Rails skaliert.

1.



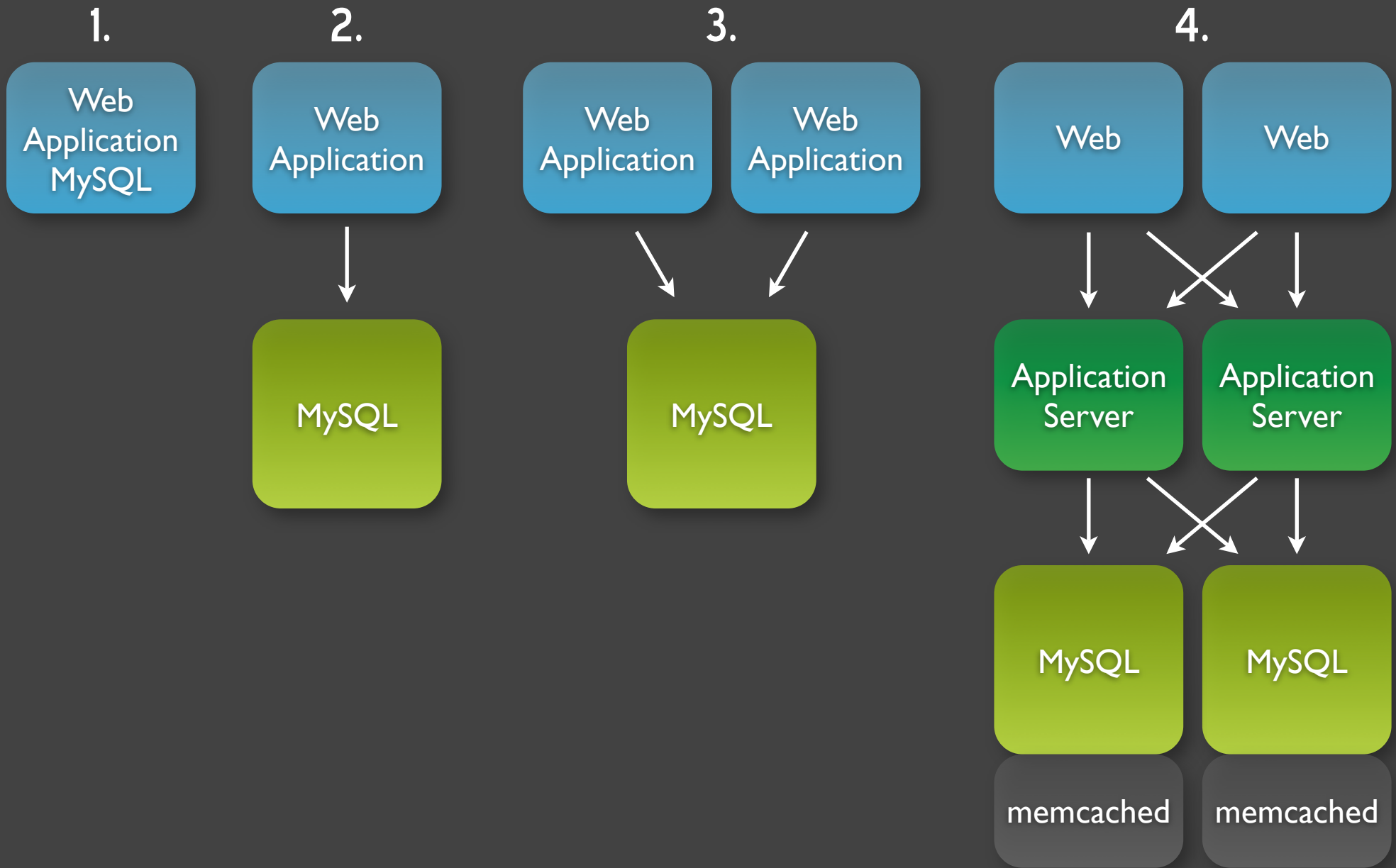
2.



3.



Rails skaliert.



Der Haken.

Der Haken.

- Ruby ist langsamer als andere interpretierte Sprachen
- Einfacher neue Anwendungen nach Rails Conventions zu entwickeln als Rails an bestehende Anwendungen und Datenbanken zu adaptieren
- Junges Framework, wenig kommerzieller Support (Aber extrem hilfsbereite Community.)
- Kleine Auswahl an Hosting Providern

Rails wird genutzt.

Rails wird genutzt.

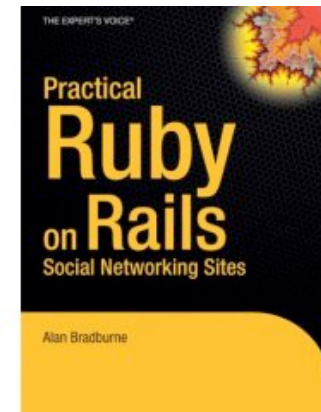
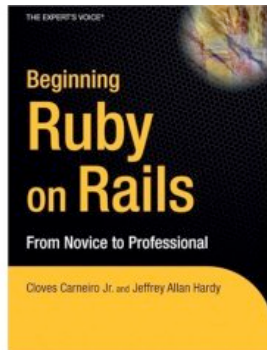
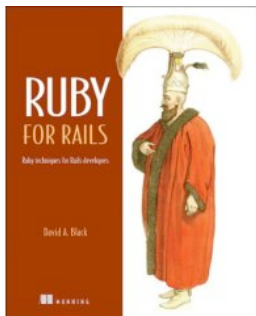
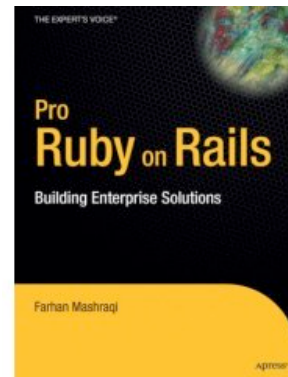
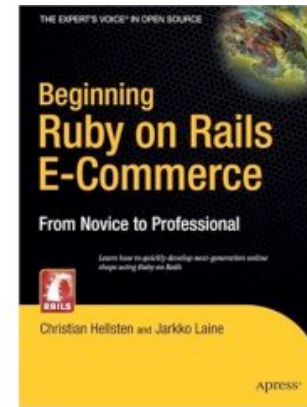
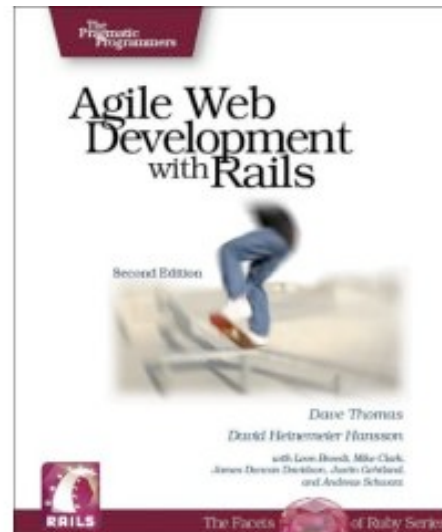
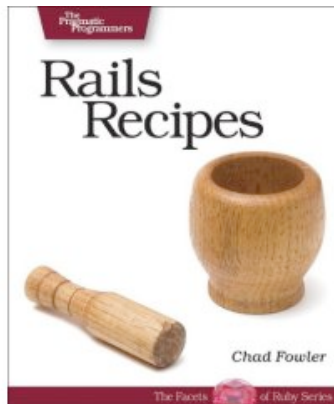
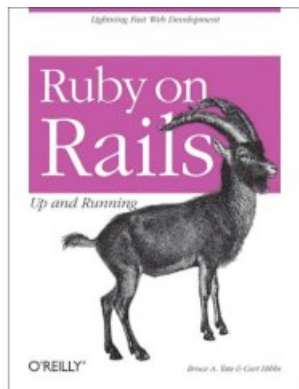
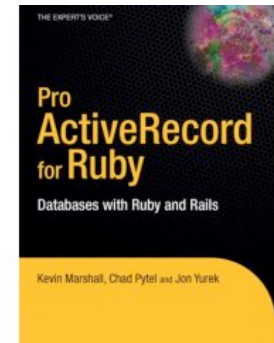
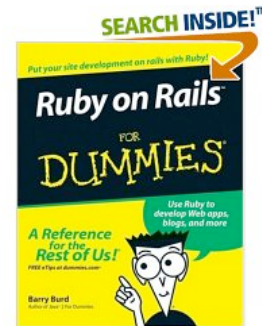
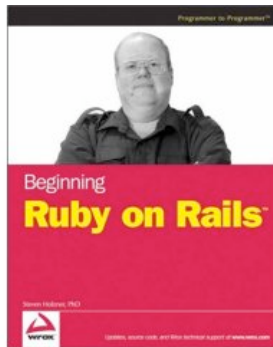
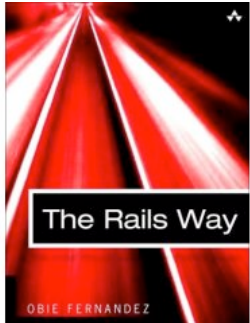


Von über 6.500 Entwicklern aus über 90 Ländern.

<u>Afghanistan (2)</u>	<u>Argentina (29)</u>	<u>Australia (99)</u>	<u>Austria (14)</u>	<u>Azerbaijan (4)</u>
<u>Bahrain (1)</u>	<u>Belarus (2)</u>	<u>Belgium (34)</u>	<u>Brazil (112)</u>	<u>Bulgaria (8)</u>
<u>Cambodia (1)</u>	<u>Canada (142)</u>	<u>Chile (5)</u>	<u>China (31)</u>	<u>Colombia (4)</u>
<u>Cook Islands (1)</u>	<u>Costa Rica (4)</u>	<u>Croatia (5)</u>	<u>Czech Republic (15)</u>	<u>Denmark (46)</u>
<u>Dominican Republic (1)</u>	<u>Ecuador (2)</u>	<u>Egypt (7)</u>	<u>El Salvador (1)</u>	<u>England (1)</u>
<u>Estonia (5)</u>	<u>Finland (18)</u>	<u>France (85)</u>	<u>Georgia (2)</u>	<u>Germany (128)</u>
<u>Gibraltar (1)</u>	<u>Greece (4)</u>	<u>Guatemala (3)</u>	<u>Hong Kong (2)</u>	<u>Hungary (11)</u>
<u>Iceland (1)</u>	<u>India (110)</u>	<u>Indonesia (7)</u>	<u>Iran (3)</u>	<u>Ireland (19)</u>
<u>Israel (19)</u>	<u>Italy (40)</u>	<u>Jamaica (1)</u>	<u>Japan (20)</u>	<u>Jordan (1)</u>
<u>Kazakhstan (3)</u>	<u>Korea (South) (9)</u>	<u>Kuwait (1)</u>	<u>Latvia (3)</u>	<u>Lithuania (7)</u>
<u>Luxembourg (2)</u>	<u>Macedonia (1)</u>	<u>Malawi (1)</u>	<u>Malaysia (9)</u>	<u>Maldives (2)</u>
<u>Mexico (10)</u>	<u>Moldova, Republic of (3)</u>	<u>Morocco (1)</u>	<u>Netherlands (85)</u>	<u>New Zealand (24)</u>
<u>Northern Ireland (4)</u>	<u>Norway (15)</u>	<u>Pakistan (3)</u>	<u>Peru (6)</u>	<u>Philippines (20)</u>
<u>Poland (58)</u>	<u>Portugal (29)</u>	<u>Romania (16)</u>	<u>Russia (42)</u>	<u>Scotland (4)</u>
<u>Serbia and Montenegro (2)</u>	<u>Singapore (11)</u>	<u>Slovakia (4)</u>	<u>Slovenia (6)</u>	<u>South Africa (17)</u>
<u>South Korea (1)</u>	<u>Spain (81)</u>	<u>Sri Lanka (2)</u>	<u>Sweden (54)</u>	<u>Switzerland (34)</u>
<u>Taiwan (3)</u>	<u>Tanzania (1)</u>	<u>Thailand (1)</u>	<u>Trinidad and Tobago (3)</u>	<u>Turkey (10)</u>
<u>Ukraine (32)</u>	<u>United Arab Emirates (2)</u>	<u>United Kingdom (189)</u>	<u>United States (996)</u>	<u>Uruguay (3)</u>
<u>Uzbekistan (3)</u>	<u>Venezuela (5)</u>	<u>Viet Nam (2)</u>		

Rails kann man lernen.

Rails kann man lernen.





Danke für's Zuhören.



<http://rubyonrails.org/>
<http://weblog.rubyonrails.org/>

patrick@limited-overload.de
<http://limited-overload.de/>
<http://poocs.net/>

