Ruby on Rails

Web Development that doesn‘t hurt

Dezember 2008

www.xing.com/profile/**Christian_Feser**
www.xing.com/profile/**Michael_Kram**
www.xing.com/profile/**Jakob_Schroeter**
www.**Marc-Seeger**.de

Wissenskasten

Generatoren	Views	Businesslogik	Helper
ERB-Templates	Model		Assoziationen
Console		Forms	Routing
REST	Migrations	Partial	
Layout	Controller	Webservices	

agenda

- has_many
- Validatoren
- Testing
- Security
- Performance
- Deployment
- Diskussion

has_many

Wann wird es aufgerufen?

Once upon a time…

```
class Project < ActiveRecord::Base
    belongs_to                  :portfolio
    has_one                     :project_manager
    has_many                    :milestones
    has_and_belongs_to_many     :categories
 end
```

http://api.rubyonrails.com/classes/ActiveRecord/Associations/ClassMethods.html

Magic!

```
Project#portfolio
Project#portfolio=(portfolio)
Project#portfolio.nil?
Project#project_manager
Project#project_manager=(project_manager)
Project#project_manager.nil?,
Project#milestones.empty?
Project#milestones.size
Project#milestones
Project#milestones<<(milestone)
Project#milestones.delete(milestone)
Project#milestones.find(milestone_id)
Project#milestones.find(:all, options)
Project#milestones.build
Project#milestones.create
Project#categories.empty?
Project#categories.size
Project#categories
Project#categories<<(category1)
Project#categories.delete(category1)
```

Wait a second…

What?

„attribute functions“

When?

require(“my_model.rb“)

When II?

My Code

```
class Bla
  puts "LOADED BLA!"

  def some_function
  end

end
```

Testing it

```
PS D:\> irb
irb(main):001:0> require "test.rb"
LOADED BLA!
=> true
irb(main):002:0>
```

require? It wasn‘t me!

Rails::Initializer

```
def load_application_classes
  if configuration.cache_classes
    configuration.eager_load_paths.each do |load_path|
      matcher = /\A#{Regexp.escape(load_path)}(.*)\.rb\Z/
      Dir.glob("#{load_path}/**/*.rb").sort.each do |file|
        require_dependency file.sub(matcher, '\1')
      end
    end
  end
```

…or the lazy way…

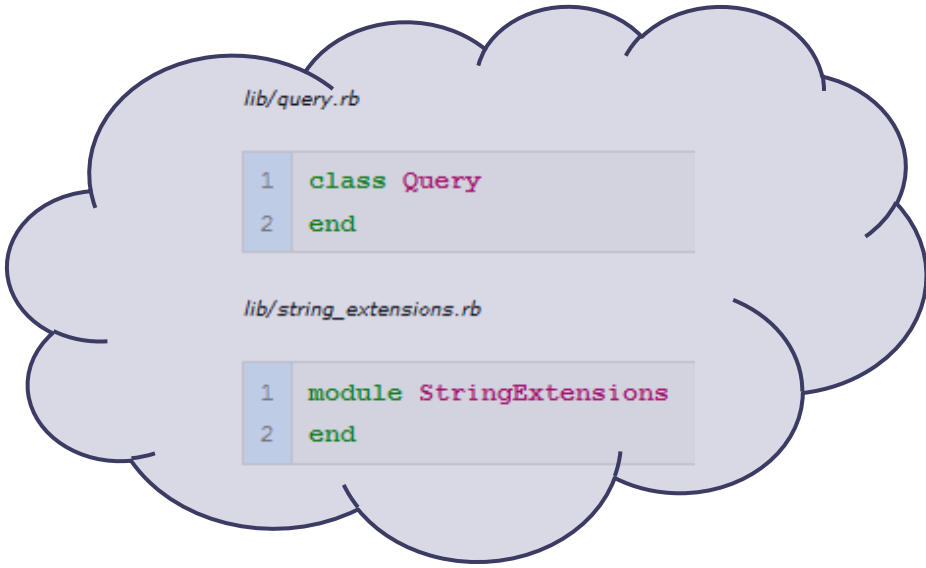
- require_association_class()

```
def has_many(association_id, options = {})
        [...]
        require_association_class(association_class_name)
        [...]
```

In general...

- This is what you can expect to be loaded:

•Models
•Views
•Controllers
•Helpers
•lib/



How? Meta-Programming!

look at input

manipulate code

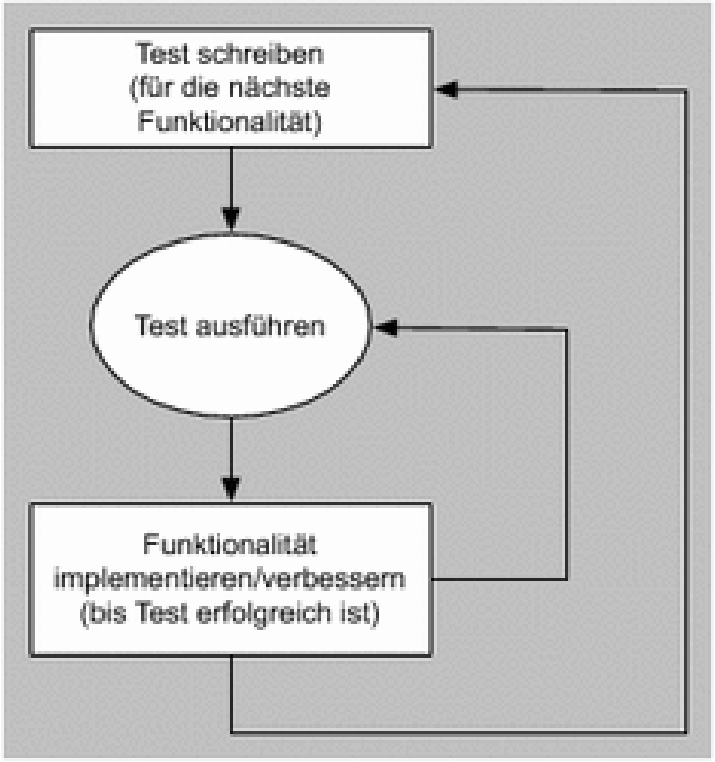

Pie I have eaten
Pie I have not yet eaten

Testing

Fast, Sexy and Svelte

Was ist TDD?

TDD - Test-Driven Development



Überblick

- **Unit-Test:**
  Hiermit werden hauptsächlich Models getestet.
- **Functional-Tests:**
  Setzt den Fokus auf das Testen von Controllern und Views.
- **Integration-Tests:**
  Dient zum Testen der Gesamtfunktionalität der Rails-Applikation.

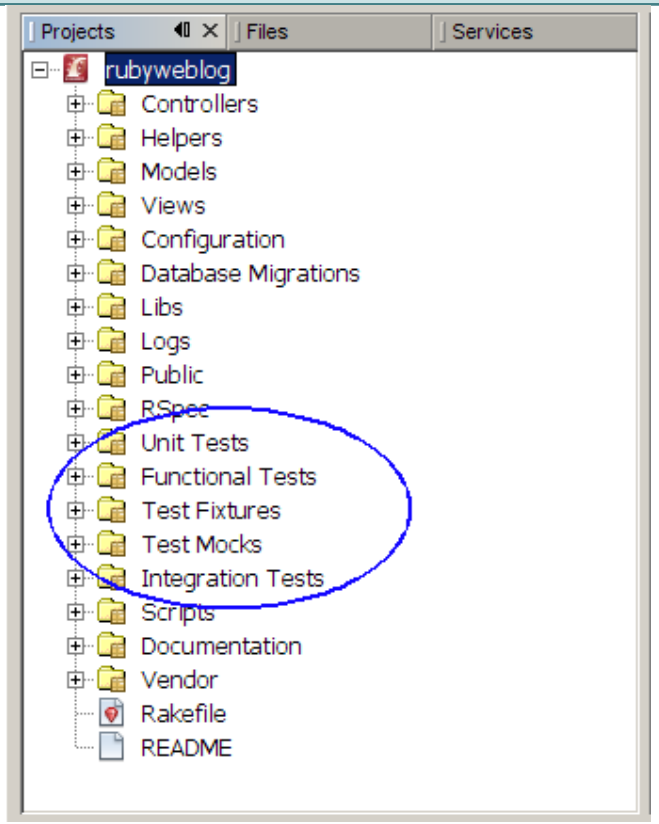
Was macht Rails für uns?

- Struktur anlegen:
  - Unit Tests (test/unit)
  - Functional Tests (test/functional)
  - Integration Tests (test/integration)
  - Test Fixtures (test/fixtures)

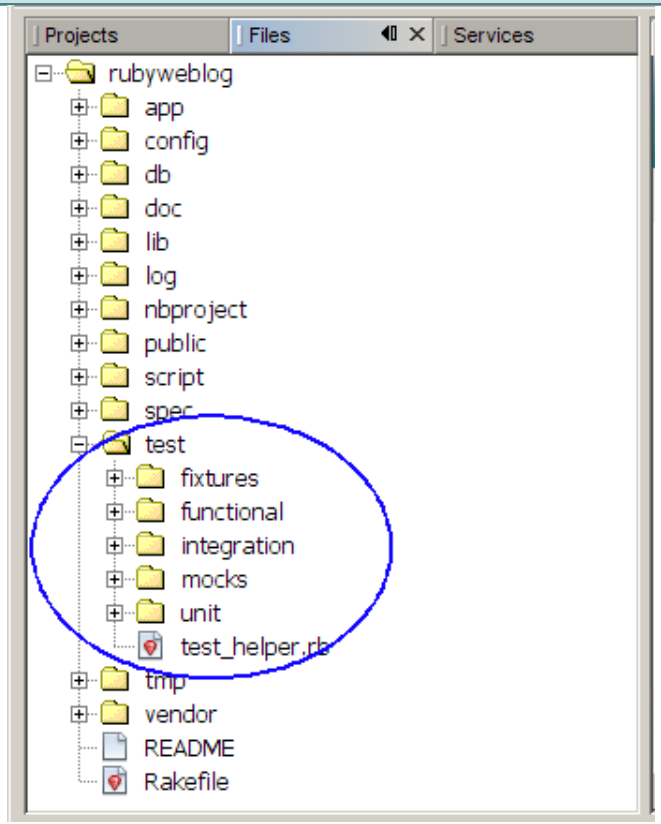
- Umgebung handeln:
  ➔ database.yml

Test - Verzeichnisse

Logisch

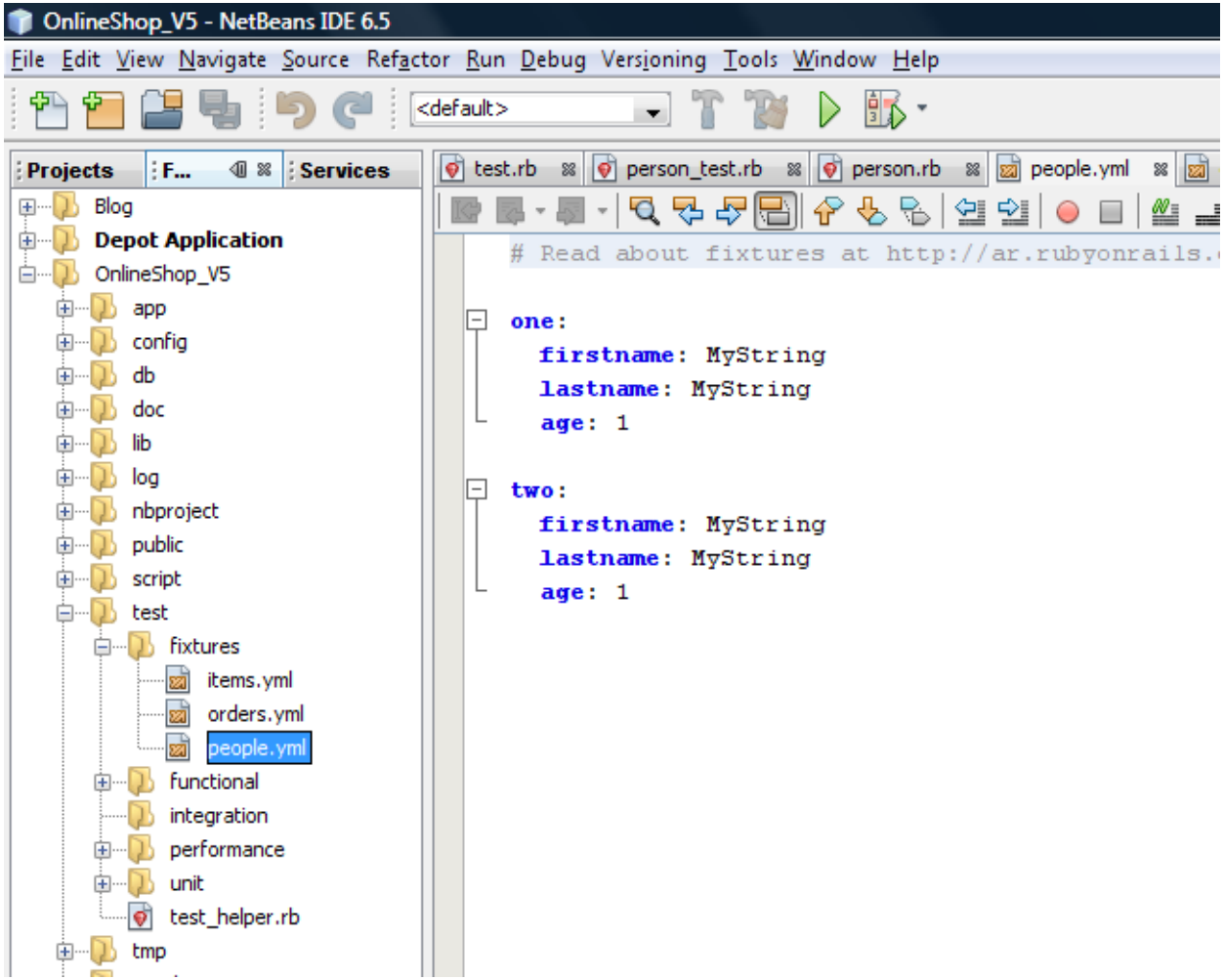


Physikalisch



Fixtures

- Testdaten, die Rails vor den Tests in die Modelle lädt



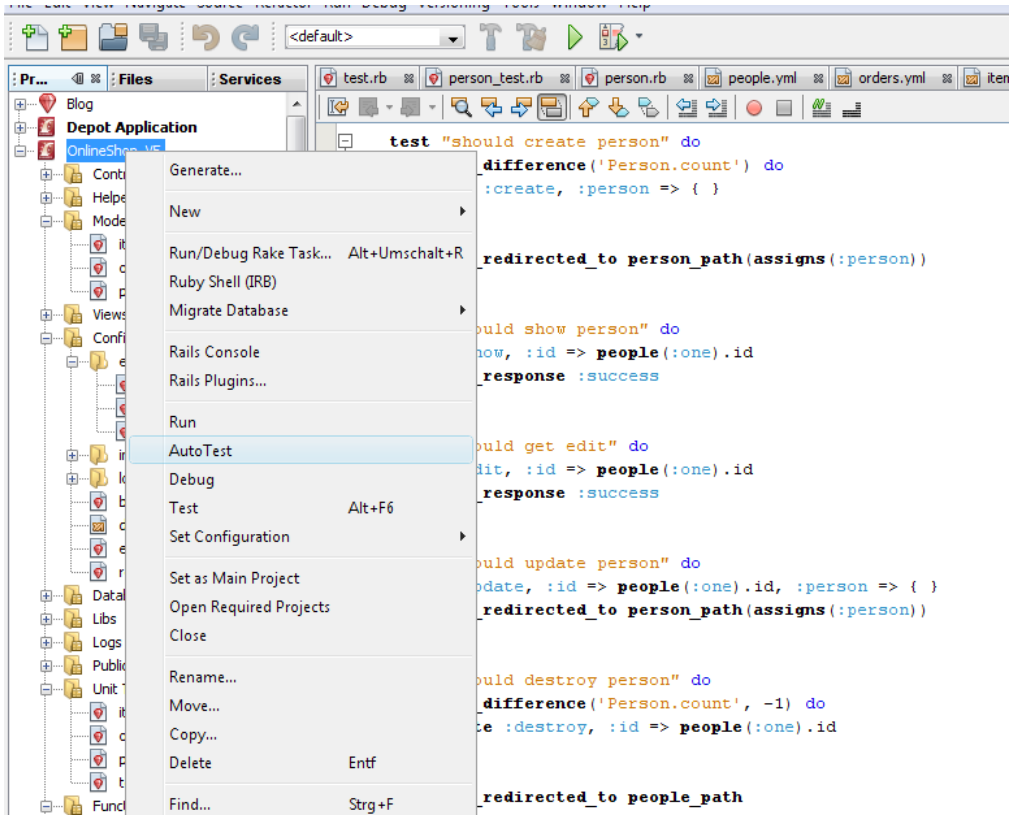
Unit Testing Class

- Subclass von *ActiveSupport::TestCase class*
  - *class PersonTest < ActiveSupport::TestCase*
- Benötigt test_helper
  - *require File.dirname(__FILE__) + '/../test_helper'*
- Test Methoden beginnen mit *test_*
  - *test_my_method*
- Mit Assertion Methoden Überprüfung auf *true*
  - assert_equal 3, Item.count

Unit Test - Demo

Autotest

ZenTest ➔ http://zentest.rubyforge.org/

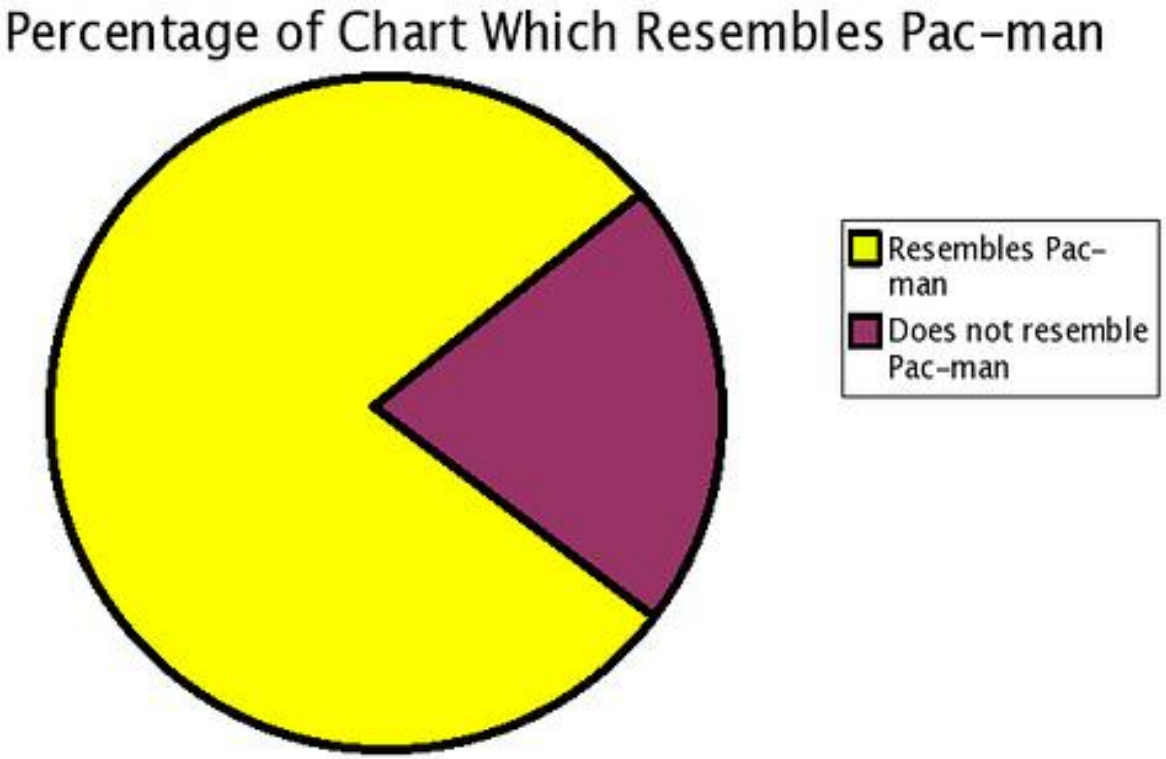


Functional-Test

Testen, ob...

- eine bestimmte Zeichenkette angezeigt wird.
- der Controller ein bestimmtes Template anzeigt.
- der Controller richtig weiterleitet.
- ob die Seite korrekt geladen worden ist.
- ob die richtigen Parameter übergeben worden sind.
- das Routing zu dem Controller korrekt funktioniert.
- ...

Functional-Test Demo


Percentage of Chart Which Resembles Pac-man
Resembles Pac-man
Does not resemble Pac-man

Integration-/Acceptance Test

1. Seite »flights/new« aufrufen.
2. Überprüfen, ob die Seite fehlerfrei aufgerufen werden konnte (HTTP-Status = 200).
3. Überprüfen, ob das Template »flights/new« geladen wurde.
4. Flight-Formulardaten an die Seite »/flights« schicken mit der HTML-Methode POST.
5. Der Weiterleitung folgen.
6. Überprüfen, ob die Seite ohne Fehler aufgerufen werden konnte (HTTP-Status = 200).
7. Überprüfen, ob das Template »flights/show« geladen wurde.

Integration-/Acceptance Test

1. Seite »flights/new« aufrufen.
2. Überprüfen, ob die Seite fehlerfrei aufgerufen werden konnte (HTTP-Status = 200).
3. Überprüfen, ob das Template »flights/new« geladen wurde.
4. Flight-Formulardaten an die Seite »/flights« schicken mit der HTML-Methode POST.
5. Der Weiterleitung folgen.
6. Überprüfen, ob die Seite ohne Fehler aufgerufen werden konnte (HTTP-Status = 200).
7. Überprüfen, ob das Template »flights/show« geladen wurde.

```
require "#{File.dirname(__FILE__)}/../test_helper"

class GeneralStoriesTest < ActionController::IntegrationTest
  fixtures :countries, :airports

  # Replace this with your real tests.
  def test_new_flight
    get '/flights/new'
    assert_equal 200, status
    assert_template 'flights/new'

    post '/flights', :flight => valid_flight_attributes
    follow_redirect!
    assert_equal 200, status
    assert_template 'flights/show'
  end
end
```

SeleniumHQ

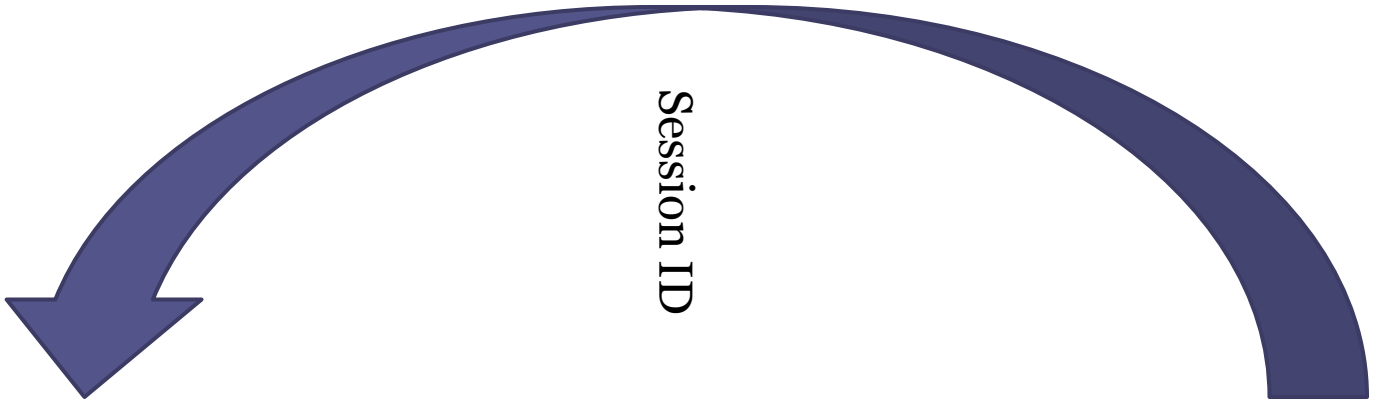
- DEMO

- http://seleniumhq.org/projects/on-rails/

Security

...on Rails

Session hijacking



CookieStore

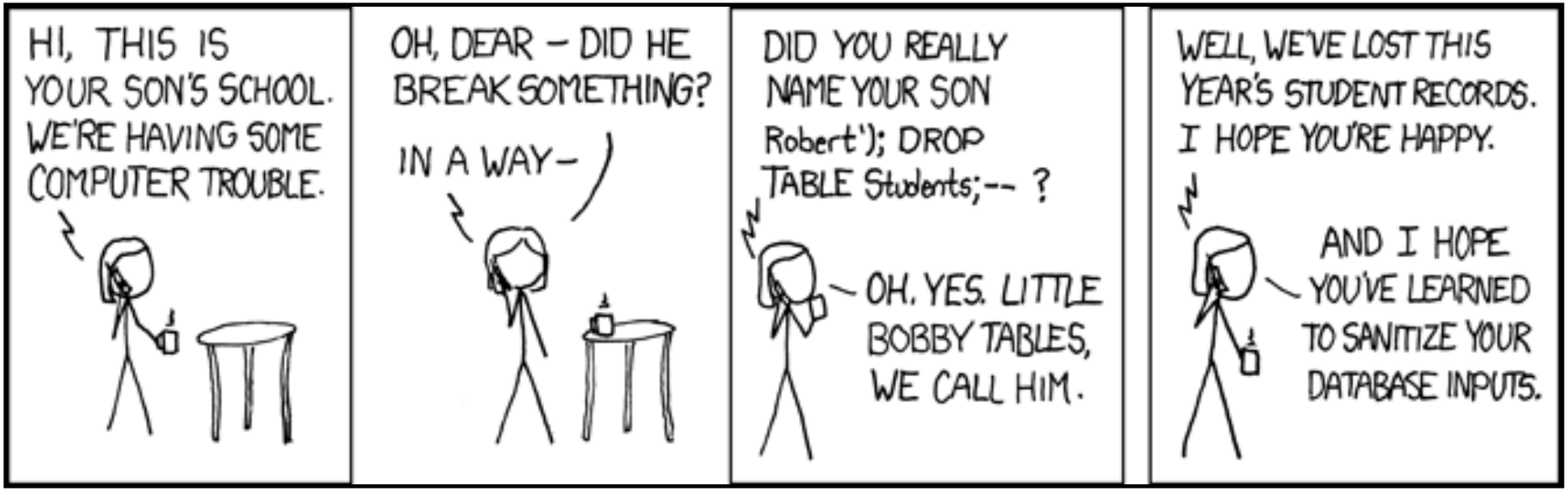
- Cookie
  - = Session ID + SHA512(Session ID+ ServerSideSecret)
  - → no tampering
- Encryption of cookie possible
  - → user can‘t see what you put in the cookie
- Especially for forms:
  - ActionController::RequestForgeryProtection

Example

```
class FooController < ApplicationController
    # uses the cookie session store (then you don't need a separate :secret)
    protect_from_forgery :except => :index

    # uses one of the other session stores that uses a session_id value.
    protect_from_forgery :secret => 'my-little-pony', :except => :index

    # you can disable csrf protection on controller-by-controller basis:
    skip_before_filter :verify_authenticity_token
  end
```

source: http://api.rubyonrails.com/classes/ActionController/RequestForgeryProtection/ClassMethods.html

SQL Injection



Source: http://xkcd.com/327/

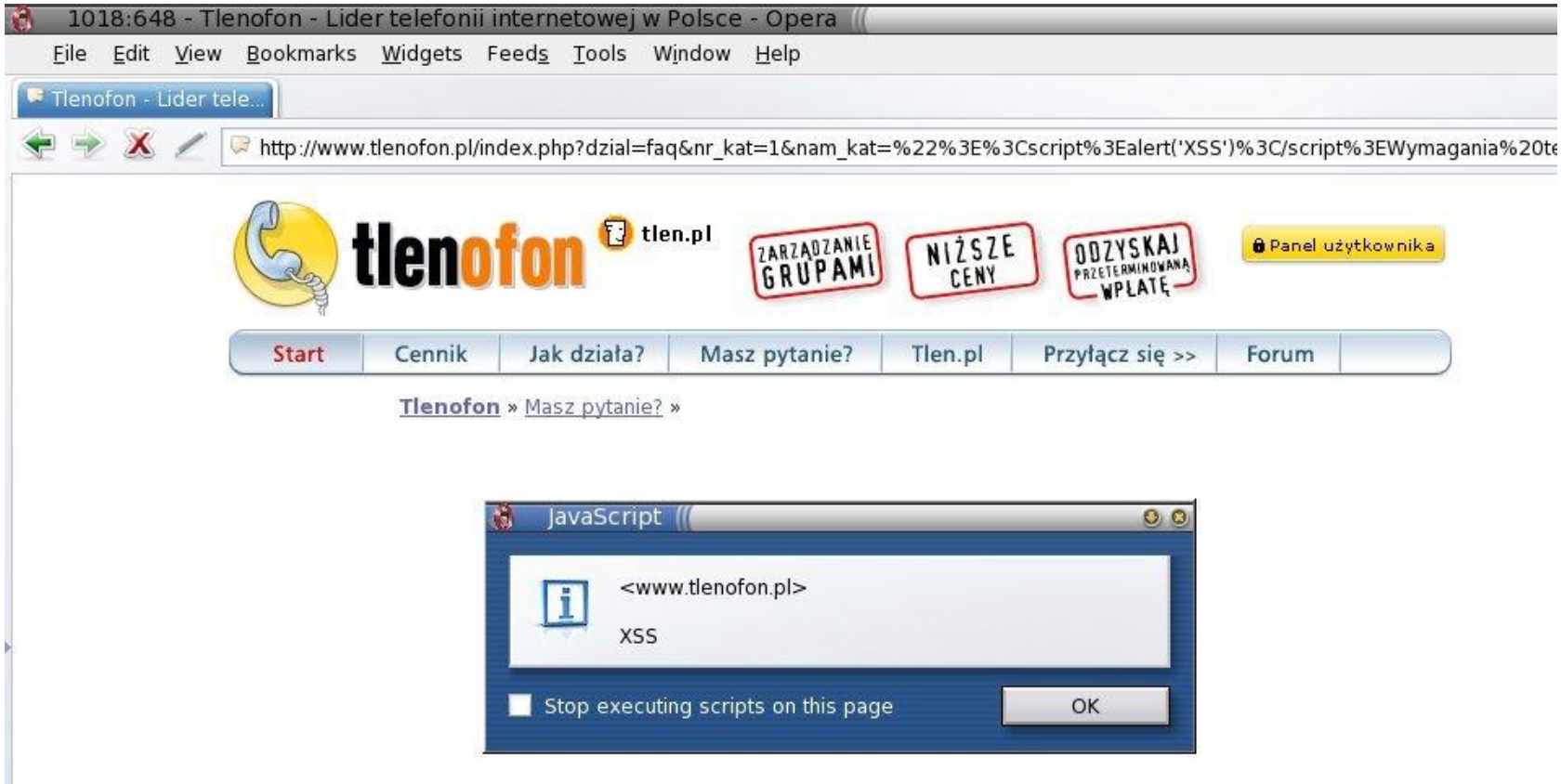
SQL Injection

automatically applies SQL Escaping
(' , " , NULL, ...)

Model.find(id)
Model.find_by_something(something)

XSS

```
<script type="text/javascript">alert("XSS");</script>
```



XSS

- html_escape()

```
<%= h foo.bar %>
```

- sanitize():

```
my_tags = ["a", "acronym", "b", "strong" "i", "em"]
my_atributes = ["href", "title"]
s = sanitize(user input, :tags => my tags, :attributes => my attributes))
```

- Safe ERB plugin
  - checks that the proper methods are used if a string is „tainted“ (read from I/O)

Performance

Rails on Speed

The known...

- Optimize your code
- Built-in Features verwenden
- Nur Daten laden, die auch verwendet werden

```
Order.find(:all, :include => [:person])
```

- Datenbank-Features nutzen
  - Stored procedures...

http://antoniocangiano.com/2007/02/10/top-10-ruby-on-rails-performance-tips/

Caching is good

- File-Caching für Controller und Views
  - Einstellung im Environment

Caching im View

- Page-Caching
  - Im Controller:

```
caches_page :index, :show

expire_page(orders_path)
```

- Action-Caching
  - Im Controller:

```
caches_action :index, :show

expire_action(orders_url)
```

Caching im View

- Fragment-Caching

```
<% cache(:action => "list") do %>
  …
<% end %>
```

```
expire_fragment(:controller => 'orders',
  :action => 'list')
```

More caching

- Im Model: cached_model
  - Plugin für Rails
  - überschreibt model.find-Methoden
  - spart DB-Queries
- Session Store

http://rubyfurnace.com/gems/cached_model

memcached

- generic, high-performance, distributed memory object caching system for speeding up dynamic web applications
- skalierbar, da globaler Cache
  - in Environment-Konfig:

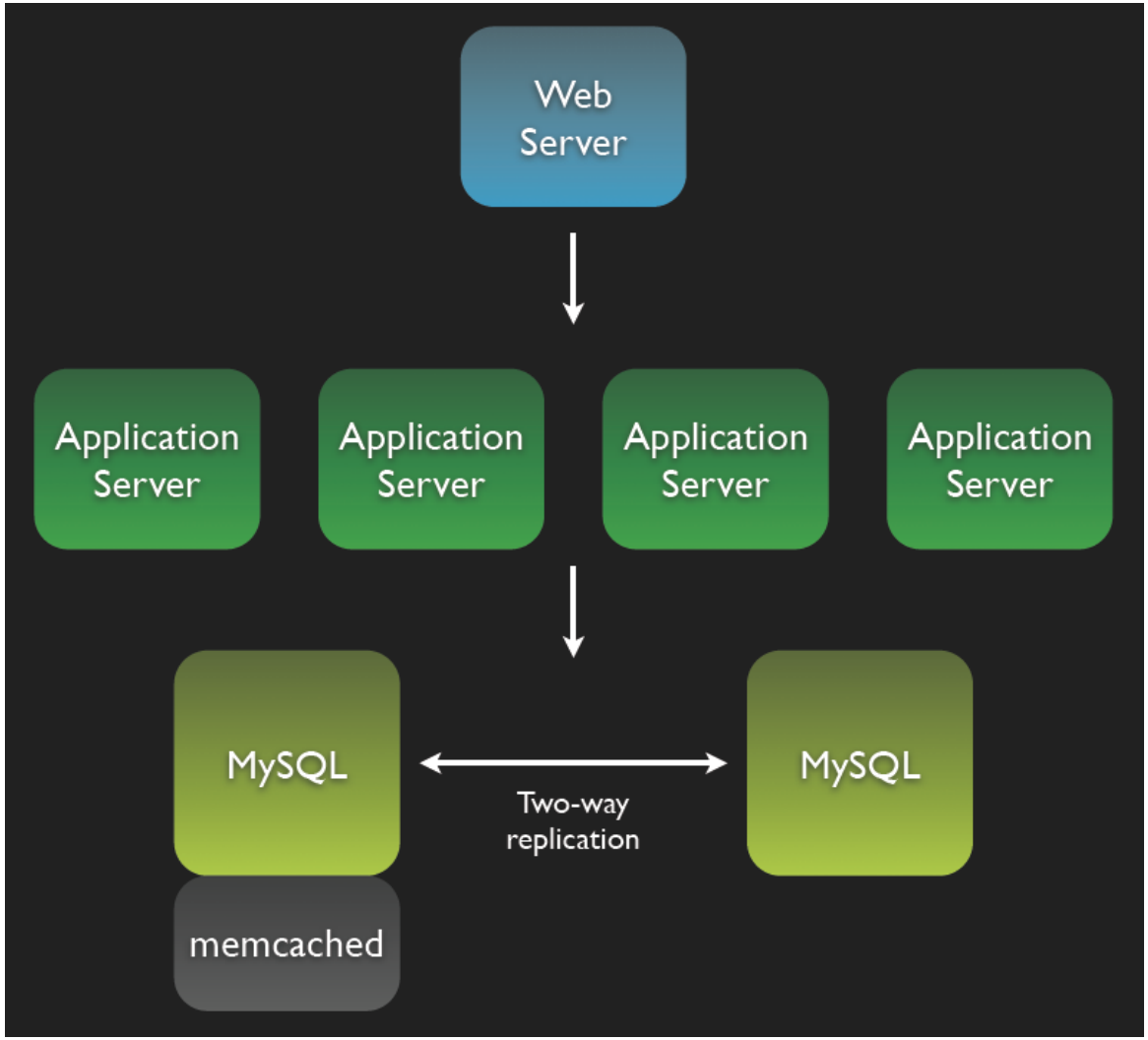
```
config.cache_store = :mem_cache_store
```

http://www.danga.com/memcached/

Rails scales

Really?

Yep, it does!



http://limited-overload.de/assets/2006/11/3/Rails-Konferenz_2006.pdf

Frontend tuning

- CSS- und JavaScript-Dateien zusammenfügen
  - javascript_include_tag
  - stylesheet_link_tag

```
javascript_include_tag "prototype",
 "cart", "checkout", :cache => "shop"
```

```
<script type="text/javascript"
 src="/javascripts/shop.js"></script>
```

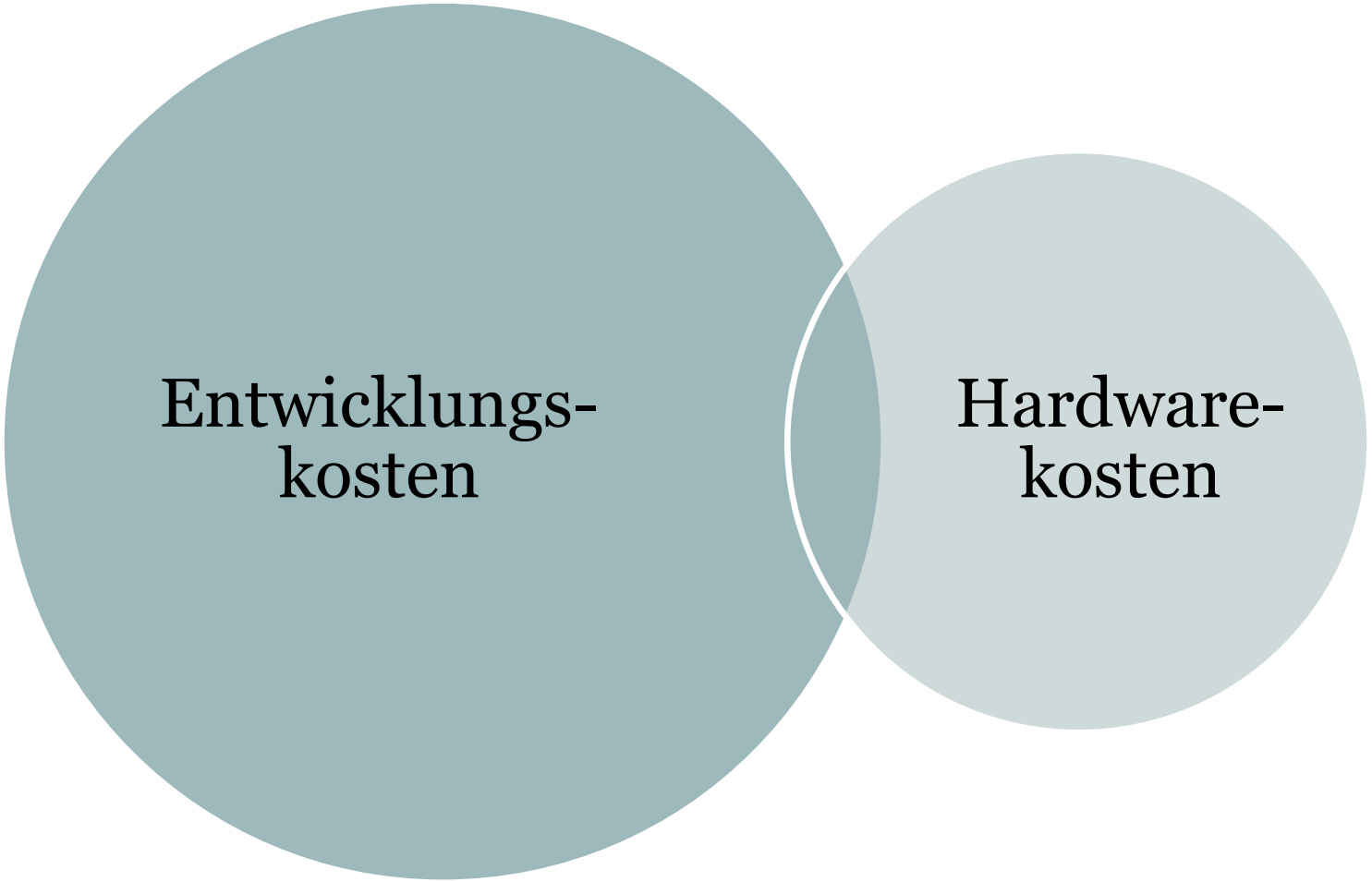
- Gzip-Komprimierung aktivieren

File-Requests auf mehrere (virtuelle) Server verteilen

- Environment-Einstellung für `javascript_include_tag`, `stylesheet_link_tag` und `image_tag`

```
config.action_controller.asset_host =
  „http://files%d.yourhost.de“
```

```
<img src="http://files0.yourhost.de
  /images/ruby.png„ />
<img src="http://files1.yourhost.de
  /images/rails.png„ />
```

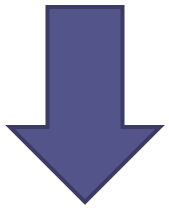

Entwicklungs-
kosten
Hardware-
kosten

Deployment

…get yourself some Rails!

In the beginning Part 1

1. cd /path/to/your/rails/app
2. mongrel_rails cluster::start



#curl -I 127.0.0.1:8000

→

HTTP/1.1 200 OK Connection: close Date: Tue, 11 Dec 2008 23:09:13 GMT Status: 200 OK Server: Mongrel 1.0.1 Content-Type: text/html Content-Length: 0

In the beginning Part 2

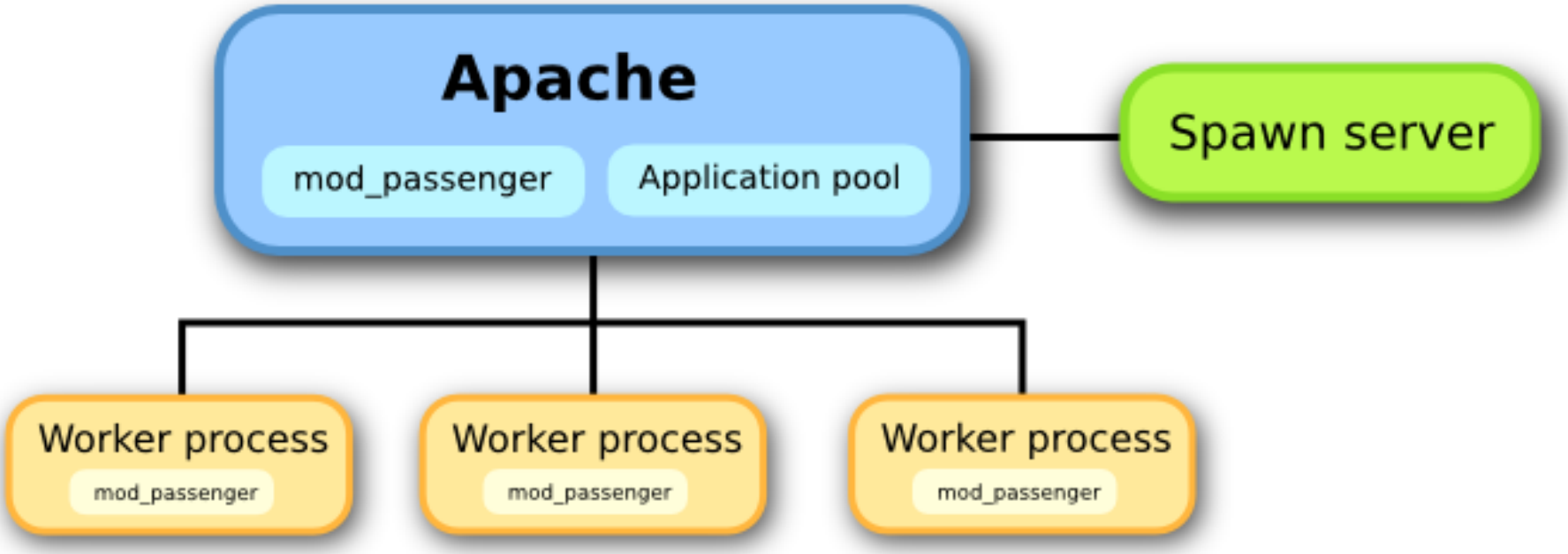
```
<VirtualHost MY-SERVERS-IP:80>
ServerName EXAMPLE.COM
ServerAlias *.EXAMPLE.COM
DocumentRoot /PATH/TO/YOUR/RAILS/APP

<Directory "/PATH/TO/YOUR/RAILS/APP/PUBLIC">

Options FollowSymLinks

AllowOverride None

Order allow,deny

Allow from all

</Directory>
```

```
<Proxy balancer://app1-mongrels>

BalancerMember http://MY-SERVERS-IP:8000

BalancerMember http://MY-SERVERS-IP:8001

BalancerMember http://MY-SERVERS-IP:8002

BalancerMember http://MY-SERVERS-IP:8003

</Proxy>
ProxyPass /images !

ProxyPass /stylesheets !

ProxyPass / balancer://app1-mongrels/

ProxyPassReverse / balancer://app1-mongrels/

ProxyPreserveHost On
</VirtualHost>
```

Today:


```
# gem install passenger
# passenger-install-apache2-module
```

Passenger (= mod_rails = mod_passenger)

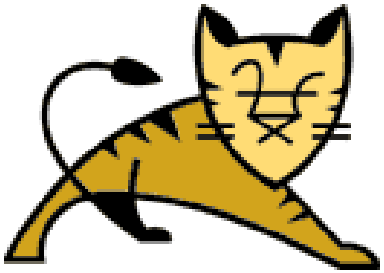


Configuration

```
<VirtualHost *:80>
    ServerName www.mycook.com
    DocumentRoot /webapps/mycook/public
</VirtualHost>
```

JRuby

Tomcat

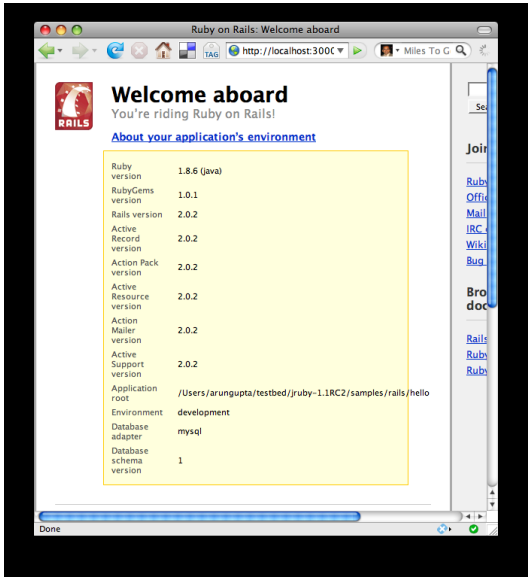


Glassfish

Glassfish

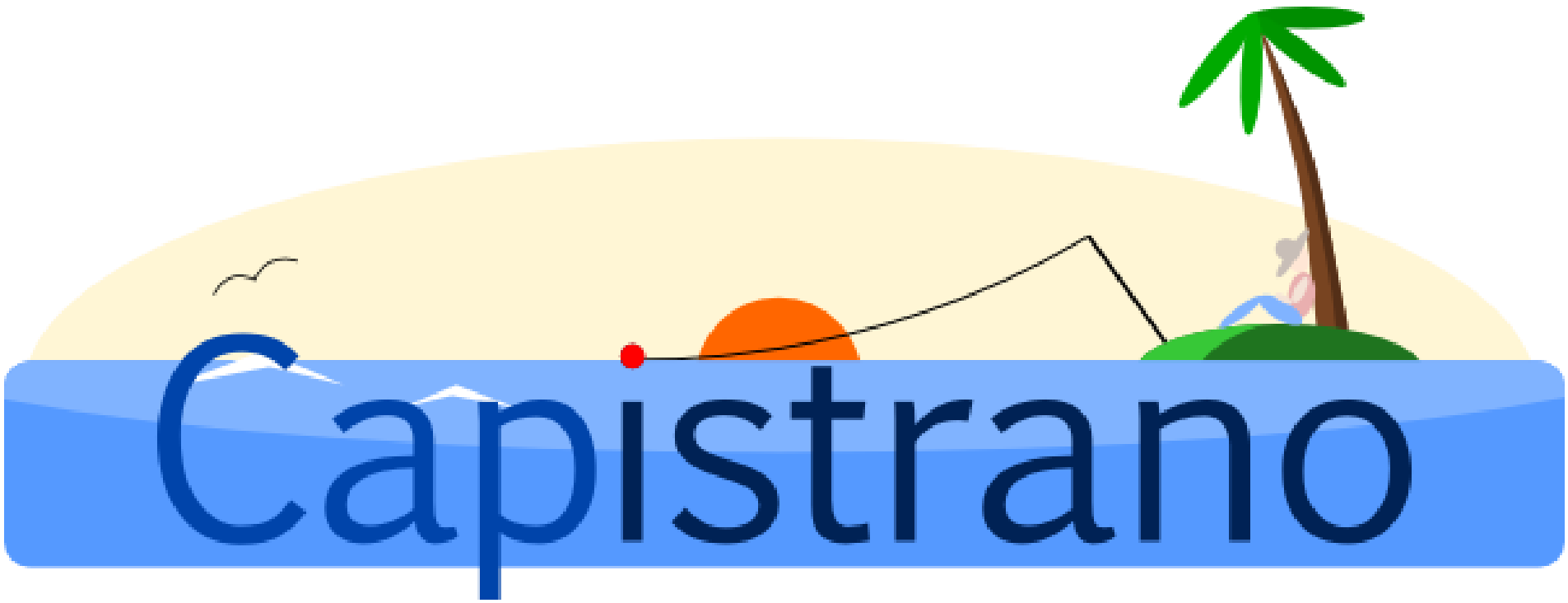
First: gem install glassfish

Then:
cd /my/rails/app
glassfish



Tomcat (or anything else…)

- gem install warbler
- cd /my/rails/app
- warble



Yes, it‘s the actual logo…

Diskussion

Praxistauglich

Enttäuscht

Stärken

Euer Eindruck?

Begeistert

Hype

Schwächen

Quellen

- Fischer, Jens-Christian: Professionelle Webentwicklung mit Ruby on Rails 2 (2008): mitp
- http://errtheblog.com/posts/25-memcaching-rails
- http://antoniocangiano.com/2007/02/10/top-10-ruby-on-rails-performance-tips/
- http://api.rubyonrails.org/
- Morsy, Hussein; Otto, Tanja: Ruby on Rails 2 – Das Entwickler-Handbuch (2008): Galileo Press
- Wartala, Ramon; Krutisch, Jan: Webanwendungen mit Ruby on Rails (2007): Addision-Wesley Verlag
- Sang Shin: *javapassion.com/rubyonrails/*
- http://wiki.rubyonrails.com/rails/pages/HowtosTesting
- Bruce, Tate; Curt Hibbs (2006): Durchstarten mit Ruby on Rails
- http://zentest.rubyforge.org/
- http://seleniumhq.org/projects/on-rails