Internettechnologie: Practicum 1

Academiejaar 2014–2015

dr. ir. Ruben Verborgh

Practica Internettechnologie

- inleiding
- overzicht practica
- praktische afspraken

Practicum 1

- ontwerp en implementatie van web-API's
- programmeertaal Ruby
- Ruby on Rails framework
- Aptana Studio
- opgave

Practica Internettechnologie

3 practica

- inleidingsles
- oplossen in PC-klas Zuse
- groepen van 2 studenten

groepen

- inschrijven via Minerva vóór 19 februari 2015
- dezelfde groepen voor alle practica
- 1 resultaat per groep

Begeleiders

Hoofdbegeleider

Ruben Verborgh – *ruben.verborgh@ugent.be*

mail voor alle vragen en opmerkingen over practica

onderwerp: **[ITech]** ***groepsnummer*** **Korte samenvatting**

Medebegeleiders

Laurens De Vocht

Miel Vander Sande

Baptist Vandersmissen

Overzicht practica

Ontwerp en implementatie van web-API's

start: dinsdag 17 februari 2015

laatste indiening: donderdag 19 maart 2015 om 14u

Consumptie van web-API's op mobiele platformen

start: dinsdag 24 maart 2015

laatste indiening: donderdag 23 april 2015 om 14u

Het semantisch web

start: dinsdag 28 april 2015

laatste indiening: donderdag 14 mei 2015 om 14u

Praktische afspraken

indienen via Minerva Dropbox

zip-bestand met vermelding groepsnummer (zie opgaven)

versturen naar Peter Lambert en Ruben Verborgh
(niét naar iedereen)

vóór de deadline

te laat of verkeerde bestandsnaam is 0 voor dit practicum

beoordelingscriteria

stiptheid, correctheid, volledigheid, leesbaarheid

Praktische afspraken

kopieer geen oplossingen

overtredende groepen worden uitgenodigd voor gesprek

plagiaat wordt in de regel bestraft met 0

geen oplossingen op publieke computers

ook niet in de PC-klas

(on-)vrijwillige medewerking aan plagiaat

onregelmatigheden worden individueel behandeld

door practicumverantwoordelijke en/of lesgever

in overleg met de betrokken groepen

Practica Internettechnologie

- inleiding
- overzicht practica
- praktische afspraken

Practicum 1

- ontwerp en implementatie van Web API's
- programmeertaal Ruby
- Ruby on Rails framework
- Aptana Studio
- opgave

Web Application Programming Interface

Web-API's geven applicaties toegang tot functionaliteit

toegang voor mensen en machines

verschillend van gewone API's door het Web

Web-API's volgens de REST architecturale stijl

“Wat is er uniek aan het Web?”

weinig goede voorbeelden, veel tegenvoorbeelden

De REST architecturale stijl

Lijst van voorwaarden waaraan API's moeten voldoen

Meest karakteristieke voorwaarden: uniforme interface

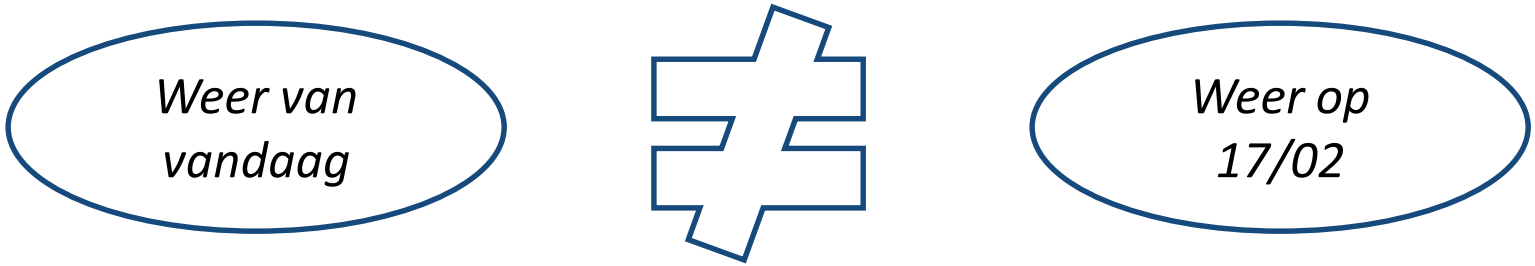
- identificeren van resources
- interactie via representaties
- zelfbeschrijvende boodschappen
- hypermedia voor overgangen

Identificatie van resources

Web API's bestaan uit resources

Resources zijn conceptuele, constante relaties waarvan de waarde varieert in tijd.

Resources op het Web hebben een HTTP(S) URL



Identificatie van resources

niet REST

http://example.org/api/
showWeather.php

Wat is dit?
Kan ik een bladwijzer maken?
Kan ik dit delen?

REST

http://example.org/weather/
ghent/today

Wat is dit?
Kan ik een bladwijzer maken?
Kan ik dit delen?

Interactie via representaties

Iedere bezoeker interageert met *dezelfde* resources

Het weer van vandaag blijft het weer van vandaag,

of dit nu door een mens of een machine gelezen wordt.

Bezoekers onderhandelen over een representatie

“Ik ben een mens, graag HTML in het Nederlands.”

“Ik ben een machine, graag XML.”

HTTP Content negotiation

Interactie via representaties

niet REST

http://example.org/weather/ghent/today **voor HTML**

http://api.example.org/api/weather/ghent/today ?format=xml **voor XML**

Kan ik een bladwijzer maken?
Kan ik dit delen?

REST

http://example.org/weather/ghent/today
voor HTML en XML

Kan ik een bladwijzer maken?
Kan ik dit delen?

Zelfbeschrijvende boodschappen

Iedere boodschap (request / response) staat op zichzelf

geen andere boodschappen of context nodig
om de boodschap te kunnen begrijpen

De uniforme interface van HTTP

Correct ontwerp van resources

Zelfbeschrijvende boodschappen

Statelessness: elke boodschap is onafhankelijk van de vorige

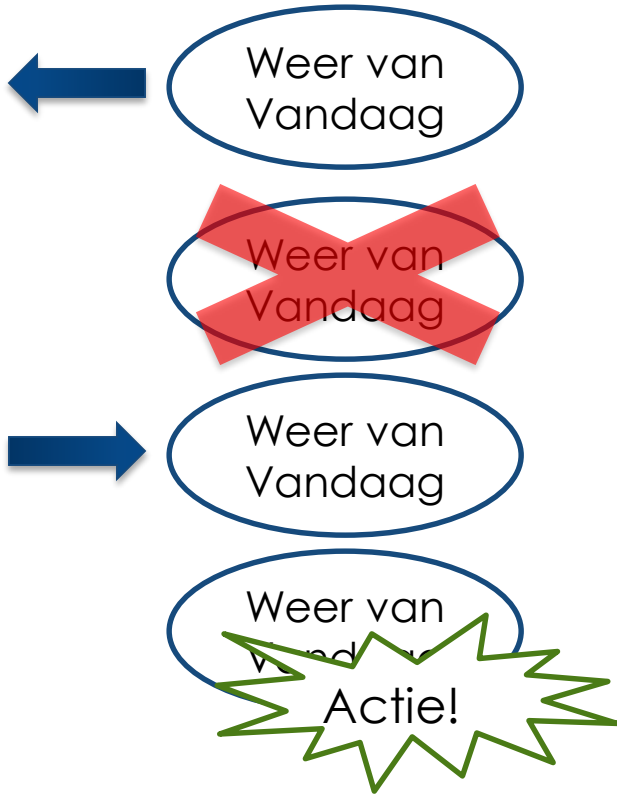
Zelf beschrijvende boodschappen: GET, DELETE, PUT, POST

GET **http://weather.com/ghent/today**

DELETE **http://weather.com/ghent/today**

PUT **http://weather.com/ghent/today**

POST **http://weather.com/ghent/today**



Zelfbeschrijvende boodschappen

niet REST

/weather?q=belgium

/page/2

/page/3

Wat is dit?
Kan ik een bladwijzer maken?
Kan ik dit delen?

REST

/weather?q=belgium

/weather?q=Belgium&p=2

/weather?q=Belgium&p=3

Wat is dit?
Kan ik een bladwijzer maken?
Kan ik dit delen?

Hypermedia voor overgangen

Navigeren tussen resources via links (statelessness)

HTML: <form> en <anchor>

Volgende actie door het activeren van links

Ook mogelijk in niet-HTML-representaties

Hypermedia voor overgangen

niet REST

```
<weather>
  <country id="belgium" />
  <country id="france" />
</weather>
```

Wat kan ik hiermee doen?
Hoe kan ik dit doen?

REST

```
<weather>
  <country
    id="http://example.org/
        weather/belgium" />
  <country
    id="http://example.org/
        weather/france" />
</weather>
```

Wat kan ik hiermee doen?
Hoe kan ik dit doen?

Het representatieformaat JSON

JavaScript Object Notation

komt voort uit de taal JavaScript, maar breder gebruikt

stelt basisstructuren uit software voor

Een JSON-object heeft deze waarden als bouwblokken:

string

integer

double

array

map *(van string naar om het even welke waarde)*

Het representatieformaat JSON

```
{
    "weather": {
        "title": "Today",
        "temp": 20.5,
        "wind": 5
    },
    "related": [
        { "title": "Yesterday","href": "/weather/ghent/yesterday" },
        { "title": "Tomorrow","href": "/weather/ghent/tomorrow" }
    ]
}
```

Introductie tot Ruby

Ruby is een dynamische scriptingtaal
met een nadruk op eenvoud en productiviteit

Elegante syntax: leesbaar en eenvoudig te schrijven

Ideaal voor webtoepassingen

Talen

Ruby, PHP	Java, C++, C#
Dynamische typering a = 14 a = “Dit is een String”	**Statische typering** int a = 14; String b = “Dit is een String”;

Geïnterpreteerd
 	
$ ruby -e 'puts "hello\n”
of
$ irb
	Gecompileerd
	
	$ javac example.java
	$ java example

Snellere aanpassingen Optimaal voor webomgevingen	**Betere prestatie**

Dit is *geen* practicum over Ruby

Dit practicum gaat over REST web-API's,
de taal Ruby is slechts een middel.

Ruby leren is niet moeilijk en haalbaar in 1 à 2 sessies:

http://tryruby.org/

https://rubymonk.com/

Probeer zelfstandig een antwoord te vinden op Ruby-vragen.

Als je een Ruby-specifieke vraag hebt, dien je altijd te tonen dat je reeds hebt gezocht op fora zoals Stackoverflow.com.

Eenvoudige klasse

Klasse Cat met attributen: *name*, *color*, *type*

```
class Cat
  attr_accessor :name, :type, :color

  def initialize(name, type, color)
    @name = name
    @type = type
    @color = color
   end
end
```

Instantie van een klasse

Instanties maken van Cat:

gc = Cat.new("GC", "short hair", "black")
lc = Cat.new("LC", “long hair", "gray")

Methode toevoegen

```
def describe
   @name + " is a " + @color + " " + @type + " cat"
end
```

Concatenatie elimineren

```
def describe
    "#{@name} is a #{@color} #{@type} cat”
end
```

Methode aanroepen

```
gc = Cat.new("GC", "short hair", "black")

puts gc.describe
```

Access Control

Methoden in een klasse zijn standaard public

Private: enkel gekend voor het individueel object

Protected: kunnen enkel aangeroepen worden door onderdelen van de klasse waarin ze gedefinieerd zijn

Variabelen

In Ruby is alles een object; variabelen zijn referenties

Geen primitieve types (int, float, boolean,…)

```
a = "mijn waarde"
b = a
a[0] = "n"  verandert a en b
```

Arrays

Array door toewijzing

```
my_array = [ "one", "two", 3, 4 ]
```

Refereren naar de array:

```
puts "my_array[0] is: #{my_array[0]}\n”
```

Hashes

Associatieve array (“Map” in Java)

Hash door toewijzing:

```
my_hash = { 'tree' => 'pine', 'bird' => 'mocking'}
puts "\n"
puts "my_hash['tree'] is: #{my_hash['tree']}\n"
puts "my_hash['bird'] is: #{my_hash['bird']}\n"
```

Ruby on Rails

Webraamwerk voor Ruby

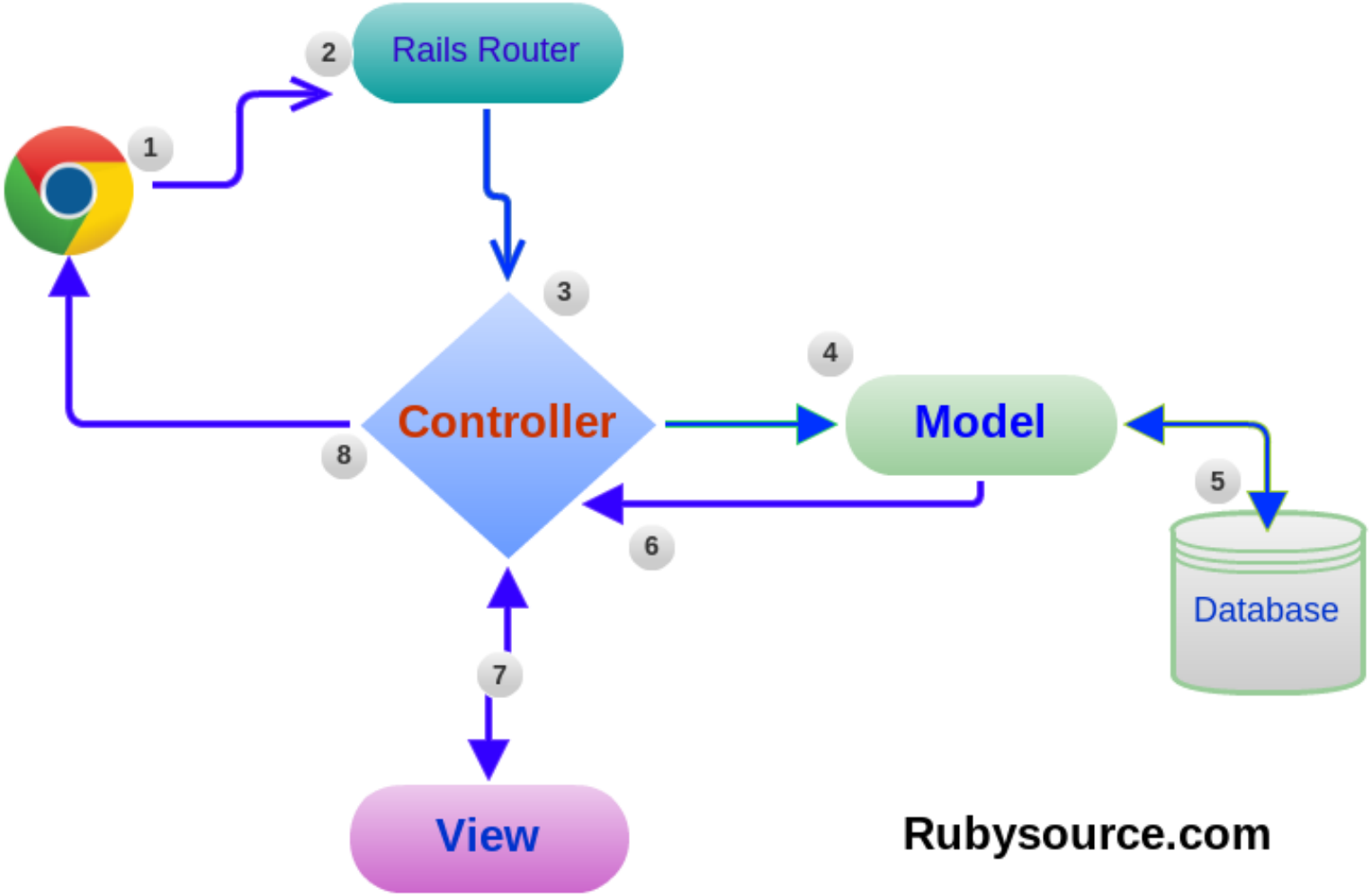
Ontworpen om Webapplicaties gemakkelijker te ontwikkelen, te implementeren en te onderhouden

Ontwerp met Model-View-Controller

Gebaseerd op het DRY-concept

- Don’t Repeat Yourself (DRY)
- Elk stuk kennis moet slechts worden uitgedrukt op één plaats

Model-View-Controller (MVC)



Mogelijke routes voor Users API

HTTP request	URI	Action	Purpose
GET	/users	`index`	page to list all users
GET	/users/1	`show`	page to show user with id `1`
GET	/users/new	`new`	page to make a new user
POST	/users	`create`	create a new user
GET	/users/1/edit	`edit`	page to edit user with id `1`
PUT	/users/1	`update`	update user with id `1`
DELETE	/users/1	`destroy`	delete user with id `1`

Model

Via ActiveRecord

volgt min of meer automatisch uit databank

View

View maakt (delen van) representaties (vb. HTML)

Kan dynamische inhoud bevatten

.html.erb bestanden

Rails ondersteunt page templates en helpers

Controller

Logisch centrum van de applicatie

Klasse die initieel wordt gegenereerd door rails

- Methoden mappen naar acties
- Zijn logisch verbonden met views

Controller-variabelen zijn beschikbaar voor views

- Controller vult variabele in
- View gebruikt dan de variabele

Aanpak Rails

Rails heeft hulpscripts

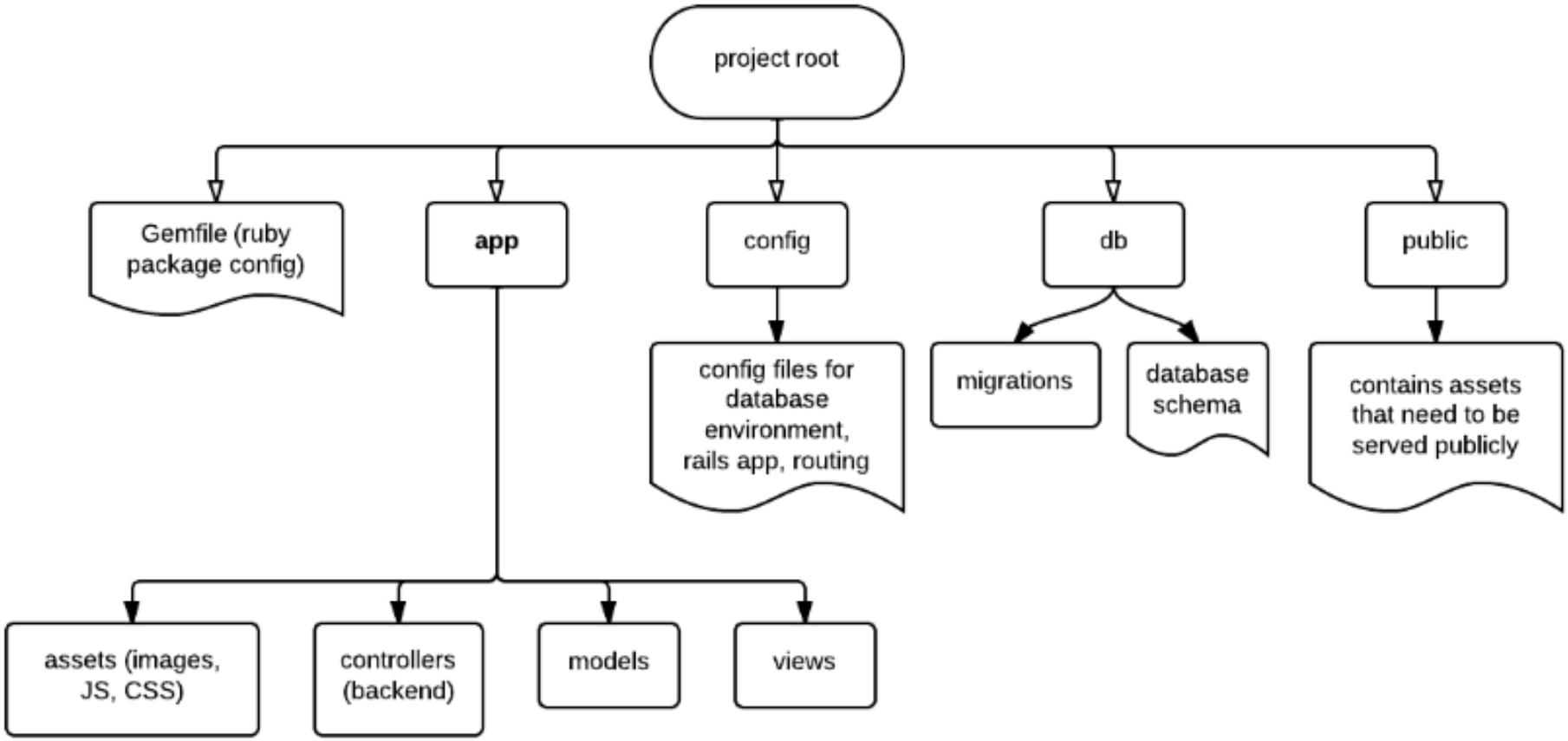
- voer de scripts uit
- pas bestanden aan waar nodig

Doel van Rails

- zorg dat er altijd iets werkt
- incrementeel ontwikkelen

Ruby-on-Rails Project maken

$ rails my_app



Hello World

$ rails HelloWorld

$ ruby script/generate controller Say

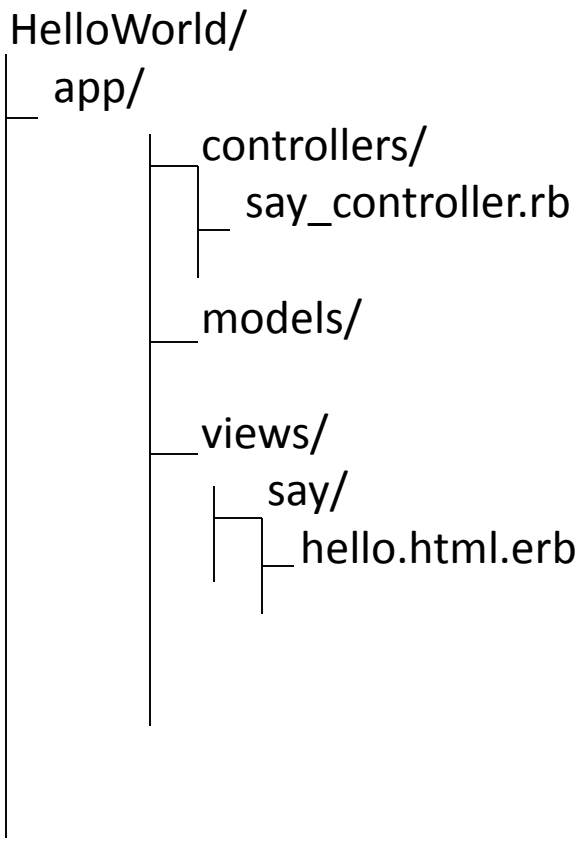
Kijkje naar de Controller

app/controllers/say_controller.rb

```
class SayController < ApplicationController
end
```

Zeer minimaal

Voeg een actie toe




```
class SayController < ApplicationController
  def hello
  end
end
```

```
<html>
<head><title>Hello World</title></head>
<body>
    <h1>Hello World</h1>
</body>
</html>
```

Rails naamconventies

Controller

URL	http://……/say/hello
Bestand	app/controllers/say_controller.rb
Klasse	SayController
Methode	hello
Layout	app/views/layouts/say.html.erb

View

URL	http://……/say/hello
Bestand	app/views/say/hello.html.erb
Helper	module SayHelper
Bestand	app/helpers/say_helper.rb

Rails dynamische pagina's

Dynamische content in .html.erb bestanden

- Expressies ter evaluatie
  - `<%= Time.now %>`
  - `<%= 1.hour.from_now  %>`
- Ruby-code
  - `<% 3.times do |count| %>`
  - `<%= count %>: Hello<br>`
  - `<% end -%>`

Rails dynamische pagina's

Controller-variabelen in views

```
class SayController < ApplicationController
  def hello
    @time = Time.now
  end
end
```

```
<html>
<head><title>Hello World</title></head>
<body>
    <h1>Hello World</h1>
    It is now <%= @time %>
</body>
</html>
```

Bestanden in een directory

Bestanden uit een map oplijsten

Index actie toevoegen aan say_controller.rb

```
def index
  @files = Dir.glob('*')
End
```

Add the index view file – index.html.erb

```
<% for file in @files -%>
file: <%= file %>
<% end -%>
```

Ruby On Rails ontwikkelen

Via de console

Zeer eenvoudig en snel

```
$ gem install rails
$ rails new app
$ rails server
```

In eigen texteditor (vim, notepad++)

Via een IDE: Aptana Studio 3

Aptana Studio 3

Beschikbaar via Athena

http://athena.ugent.be

verleen toegang tot lokale schijven met Citrix

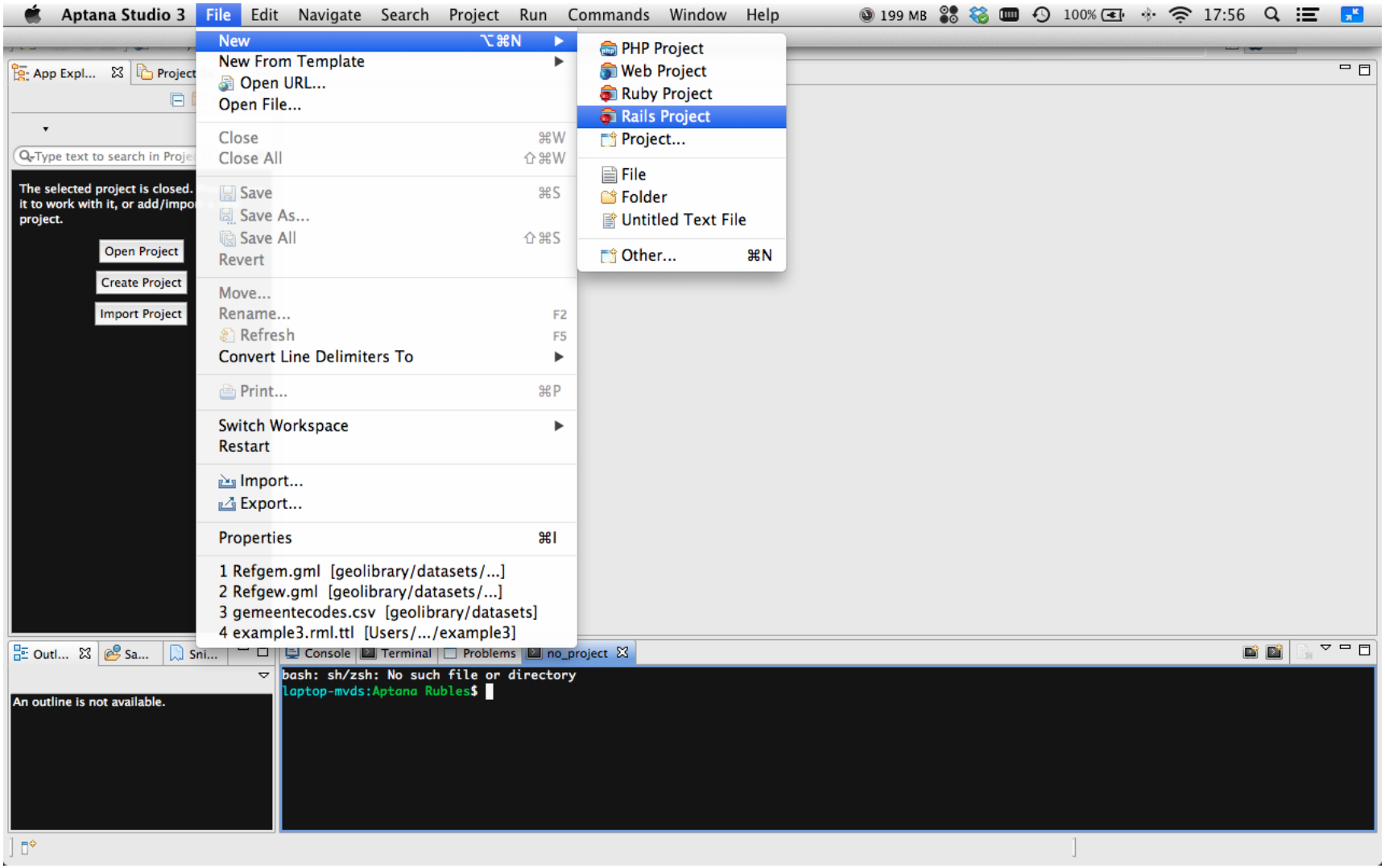
Plaats de projectfolder op “H:”

gekoppeld aan UGent-account

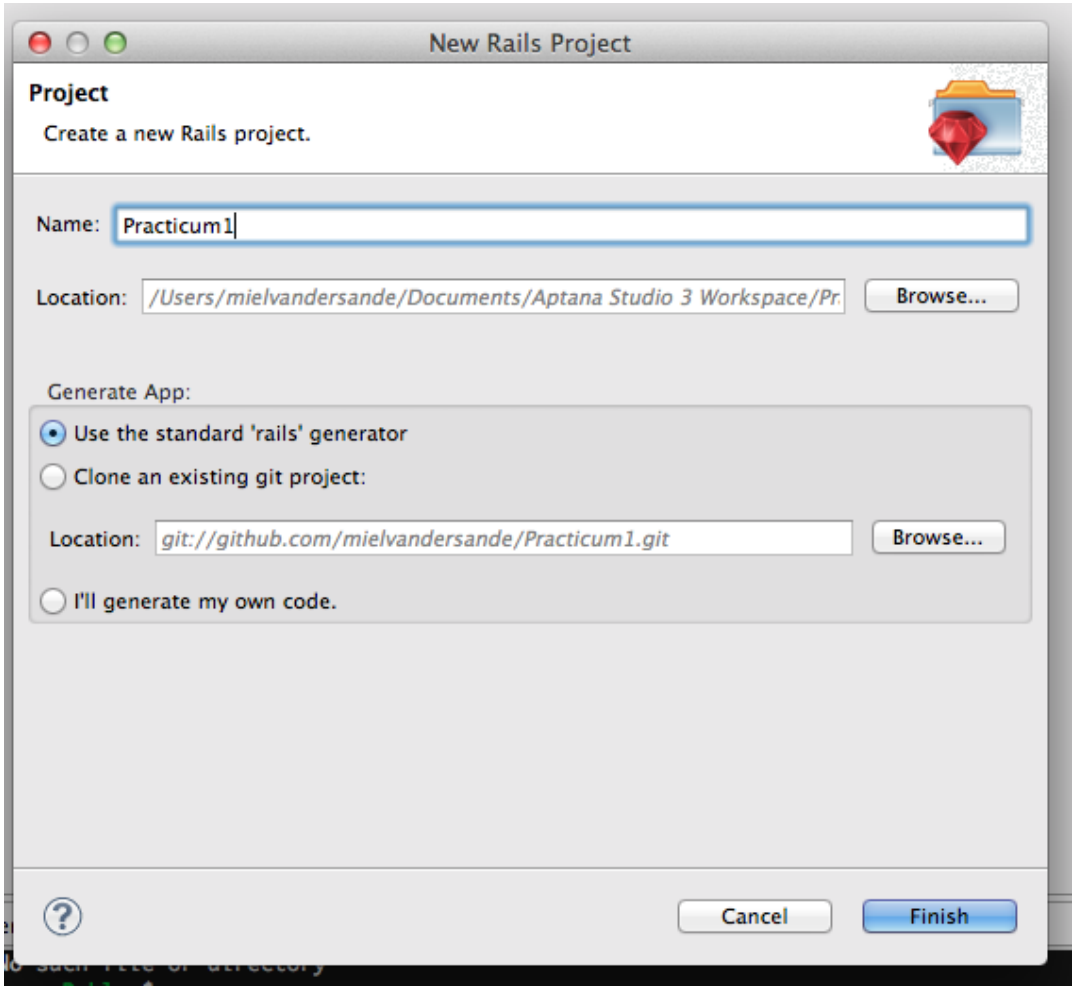
bereikbaar via Citrix-omgeving (Athena)

mount eventueel deze drive op lokale PC

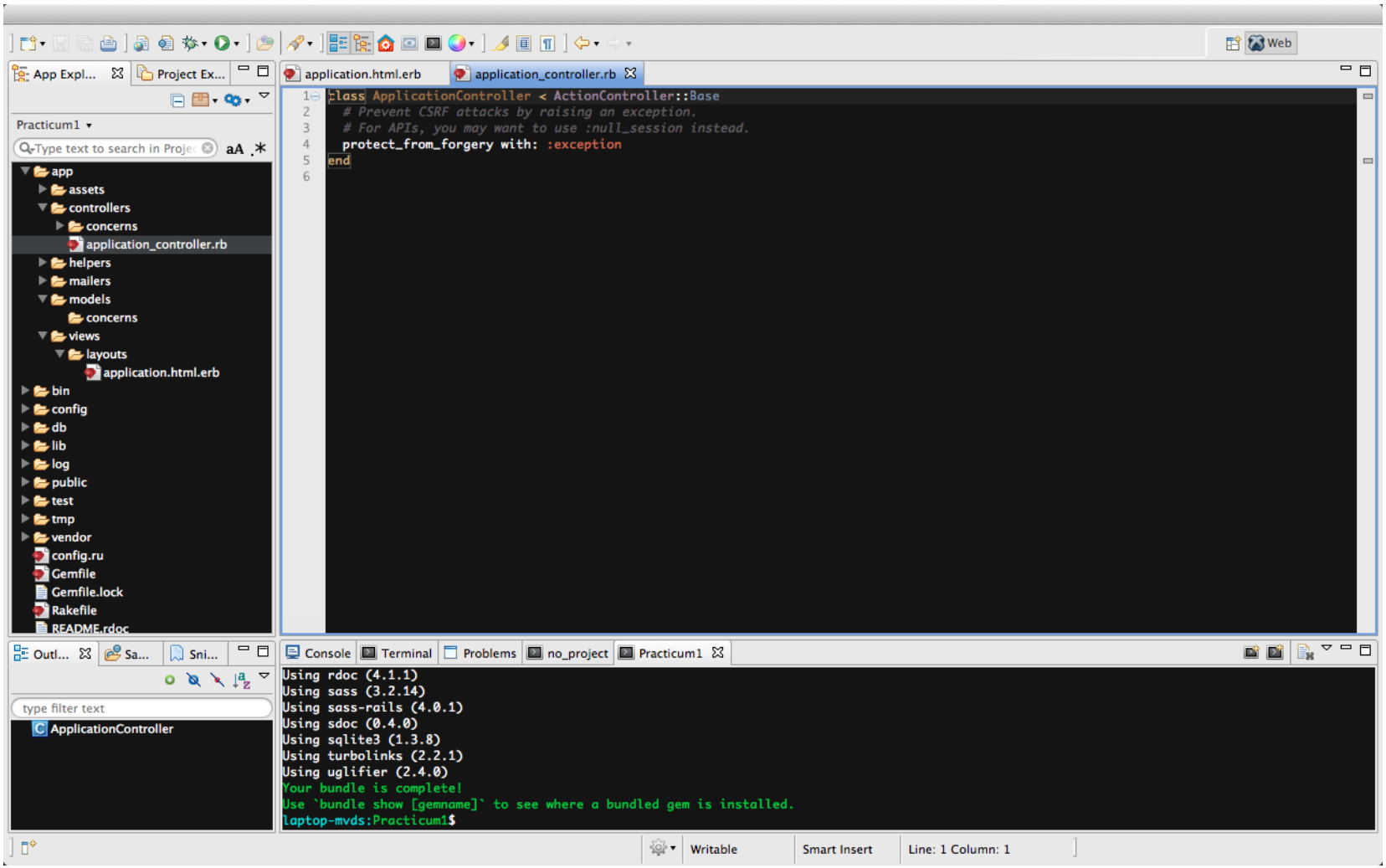
Maak een nieuw project aan



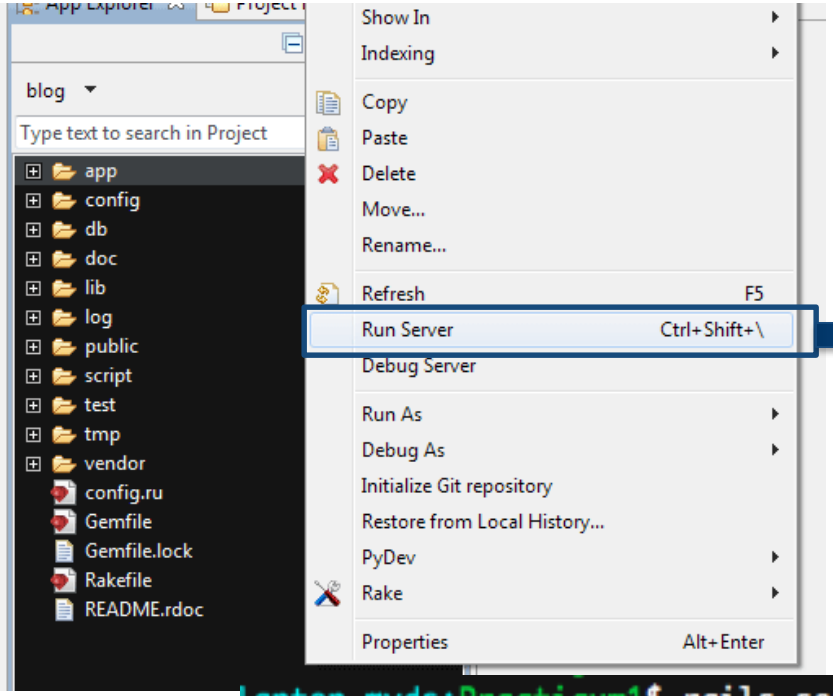
Maak een nieuw project aan



Projectstructuur is aangemaakt



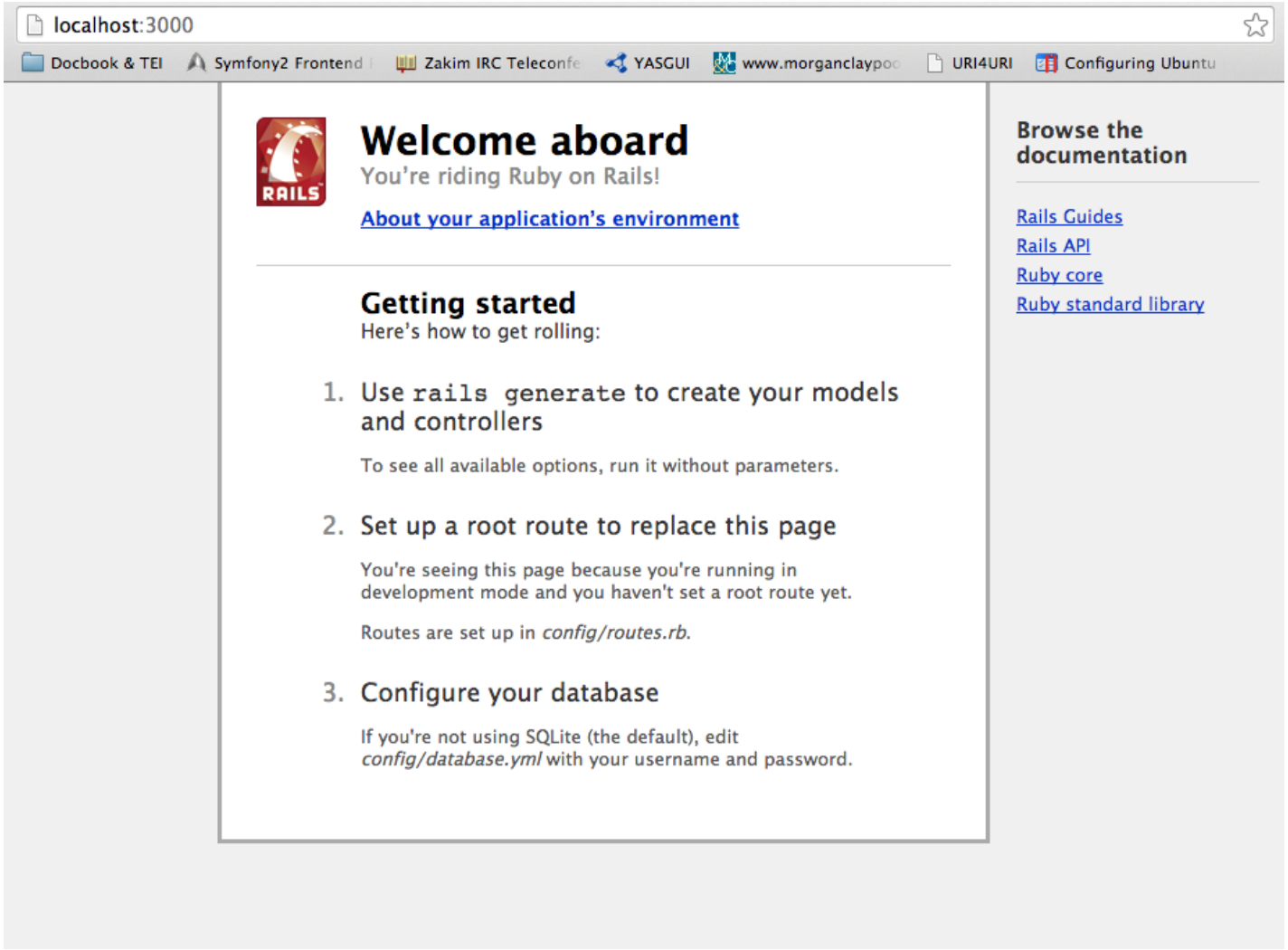
Start de server



Opent een console venster
$ rails server

```
laptop-mvds:Practicum1$ rails server
=> Booting WEBrick
=> Rails 4.0.2 application starting in development on http://0.0.0.0:3000
=> Run `rails server -h` for more startup options
=> Ctrl-C to shutdown server
[2014-02-12 18:09:34] INFO  WEBrick 1.3.1
[2014-02-12 18:09:34] INFO  ruby 2.0.0 (2013-06-27) [universal.x86_64-darwin13]
[2014-02-12 18:09:34] INFO  WEBrick::HTTPServer#start: pid=7656 port=3000
```

Open http://localhost:3000 in de browser



Per Athena-machine is er maar één poort 3000

iedere groep start server op andere poort

Poort = 3000 + groepsnummer

rails server -p 3015

Opgave practicum 1

Opgave

Kennismaking met Ruby on Rails + 3 oefeningen

Minerva > Documenten > Practica > Practicum 1

Indienen via Minerva "Dropbox"

Alle 3 oefeningen *afzonderlijk* indienen

Versturen enkel naar Peter Lambert en Ruben Verborgh

Bestandsnaam: itech-p1-g<groep>-o<opgave>.zip

Tips

lees de opgave

gebruik de help-functie van de ontwikkelomgeving

druk “F1” na het selecteren van een control, veld, …

gebruik de online documentatie

https://www.ruby-lang.org/en/documentation/

http://rubyonrails.org/documentation

http://stackoverflow.com/

gebruik de debugger