

RAILS FOR NOOBS

A 3 part, 6 hour tutorial on ruby on rails,
in Portuguese

by João Girão

NOTES

Portuguese	English	German
Parem lá com o barulho que estou a tentar dizer coisas importantes	Attention, please	Guten Tag
Sim, o meu primo que é polícia casou com a Maria lá da terra	The answer is correct	Danke
Vocês não são engenheiros não são nada!	The answer is wrong	Bitte schön
Infelizmente esqueci-me do gás ligado em casa. Volto já.	Lesson is over	Auf Wiedersehen

TODAY'S STUFF

- A short, yet comprehensive, introduction to the Ruby programming language
- Introducing Rails
- Challenge Accepted!
- Models
- Some References
- Q&A and Feedback



<https://github.com/jgirao/rails-tutorial>

FORMAT

- Chapter Start
 - Goals
 - Content
 - Recap
 - Clap, because you deserve it!
- Next Chapter (*Lather, Rinse, Repeat*)



Taking her out for a
night on the town



INTRODUCING

Part I



GOALS



- Install ruby & rvm
- Launch IRB
- Methods and Classes
- Looping
- Hashes, hashes, ha.. you get the point
- Symbols



INSTALL RVM & RUBY

Install RVM:

```
$ bash <<(curl -s https://rvm.beginrescueend.com/install/rvm)  
  
$ echo '[[ -s "$HOME/.rvm/scripts/rvm" ]] && . "$HOME/.rvm/scripts/rvm" # Load RVM function' >>  
~/.bash_profile
```

Install Ruby 1.9.2:

```
$ rvm install 1.9.2 # Dont forget to reload .bash_profile first
```

Make 1.9.2 your default:

```
$ rvm list  
$ rvm use 1.9.2 --default  
$ ruby -v
```

Reference: <http://beginrescueend.com/rvm/install/>

LAUNCH IRB

Launch IRB:

```
$ irb
```

Simple math:

```
> puts "Hello World"  
Hello World  
=> nil  
> (3+2)**5  
=> 3125  
> Math.sqrt(25)  
=> 5.0  
  
> a=3  
=> 3  
> a+2  
=> 5  
> 3.times do  
>   a+=1  
?> end  
=> 3  
> a  
=> 6
```

METHODS & CLASSES

Creating a method:

```
> def say_name(name = "John Doe")
?>   puts "His name is #{name}"
?> end
=> nil
> say_name
His name is John Doe
=> nil
> say_name "Peter Jenkins"
His name is Peter Jenkins
=> nil
```

Running a .rb:



fib.rb

```
$ ruby fib.rb
0
1
15
```

or

```
$ ./fib.rb #!/usr/bin/env ruby
```

METHODS & CLASSES

Creating our first class:



person.rb

```
class Person
  attr_accessor :name
  attr_accessor :age
  # ...
end
```

```
> require "./person"
=> true
> p1 = Person.new
```

```
def initialize(name = "John Doe", age = 18)
  @name = name
  @age = age
end

def print
  puts "My name is #{@name} and I'm #{@age} years old"
end
```

```
> p1.print
My name is John Doe and I'm 18 years old
=> nil
```

```
> p2 = Person.new("Peter Jackson", 21)
> p2.print
My name is Peter Jackson and I'm 21 years old
=> nil
```

METHODS & CLASSES

Playing with accessors:

```
class Person
  attr_accessor :name
  attr_accessor :age
  # ...
end
```

```
> p2.name
=> "Peter Jackson"
> p2.name = "Mary Poppins"
=> "Mary Poppins"
> p2.print
My name is Mary Poppins and I'm 21 years old
=> nil
```

So... accessors

Create the @var and the corresponding accessor functions

```
attr_reader :var
attr_writer :var
attr_accessor :var
```

A lot is not covered here about classes, e.g.
private, protected, inheritance, extending, etc...
Some things we will see in rails

LOOPING

for blocks

```
> for i in 1..3 do puts i end  
1  
2  
3  
=> 1..3
```

iterating an array with each

```
> [1, 2, 3, 4].each do |x| puts x end  
1  
2  
3  
4  
=> [1, 2, 3, 4]
```

Others

```
expr while condition  
expr until condition  
begin  
  exprs...  
end while condition
```

next, in for loops is the *continue* in other languages
redo, resets this iteration of the loop without checking conditions

HASHES

the traditional way

```
> h = Hash.new  
=> {}  
> h[ "name" ] = "Peter Jackson"  
=> "Peter Jackson"  
> h[ "age" ] = 22  
=> 22  
> h  
=> {"name"=>"Peter Jackson", "age"=>22}
```

using literals

```
> h = { "name" => "Mary Poppins", "age" => 33}  
=> {"name"=>"Mary Poppins", "age"=>33}  
> h[ "name" ]  
=> "Mary Poppins"
```

from hash to array to string

```
> a = h.map do |k,v| [k, v] end  
=> [[ "name", "Mary Poppins" ], [ "age", 33 ]]  
> a.join ", "  
=> "name, Mary Poppins, age, 33"
```

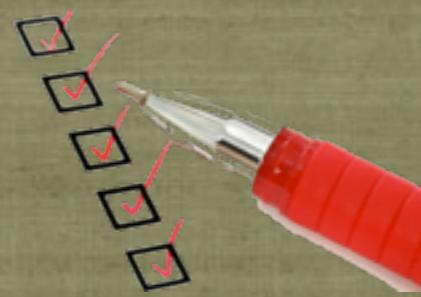
SYMBOLS

"Simply, a symbol is something that you use to represent names and strings. What this boils down to is a way to efficiently have descriptive names while saving the space one would use to generate a string for each naming instance." -- Gluttonous on Ruby Symbols

applied to hashes

```
> h1 = { :name => "Peter Jackson", :age => 22}
=> { :name=>"Peter Jackson", :age=>22}
> h1[:name]
=> "Peter Jackson"
> h2 = { :name => "Mary Poppins", :age => 33}
=> { :name=>"Mary Poppins", :age=>33}
> h1.keys[0].object_id
=> 20488
> h2.keys[0].object_id
=> 20488
> puts h2.keys[0]
name
=> nil
```

RECAP



```
$ rvm install 1.9.2  
$ rvm use 1.9.2 --default  
$ irb
```

```
> puts "Hello World"
```

```
> say_name "Peter Jenkins"  
His name is Peter Jenkins
```

```
$ ./fib.rb
```

```
class Person  
  attr_accessor :name
```

```
> require "./person"  
> p1 = Person.new
```

```
> for i in 1..3 do puts i end
```

```
> [1, 2, 3, 4].each do |x| puts x end
```

```
> h = {"name" => "Mary Poppins", "age" => 33}  
> (h.map do |k,v| [k, v] end).join ", "
```

```
> h1 = { :sym => 1}  
> h2 = { :sym => 2}  
> h1.keys[0].object_id == h2.keys[0].object_id  
=> true
```



CLAP, YOU MADE IT





Chuck Norris



INTRODUCING RAILS

Part 2

GOALS



- Install Rails 3.0
- Files & Folders
- Model View Controller
- Routing
- Interactions between Rails components (Rack & Metal)



INSTALL RAILS3.0

Create a gemset in rvm:

```
$ rvm use 1.9.2@rails3 --create
```

Install Rails 3.0:

```
$ gem install rails  
$ rails -v  
Rails 3.0.6
```

First project



bare project

```
$ mkdir -p ~/rails  
$ cd ~/rails  
$ rails new bare
```

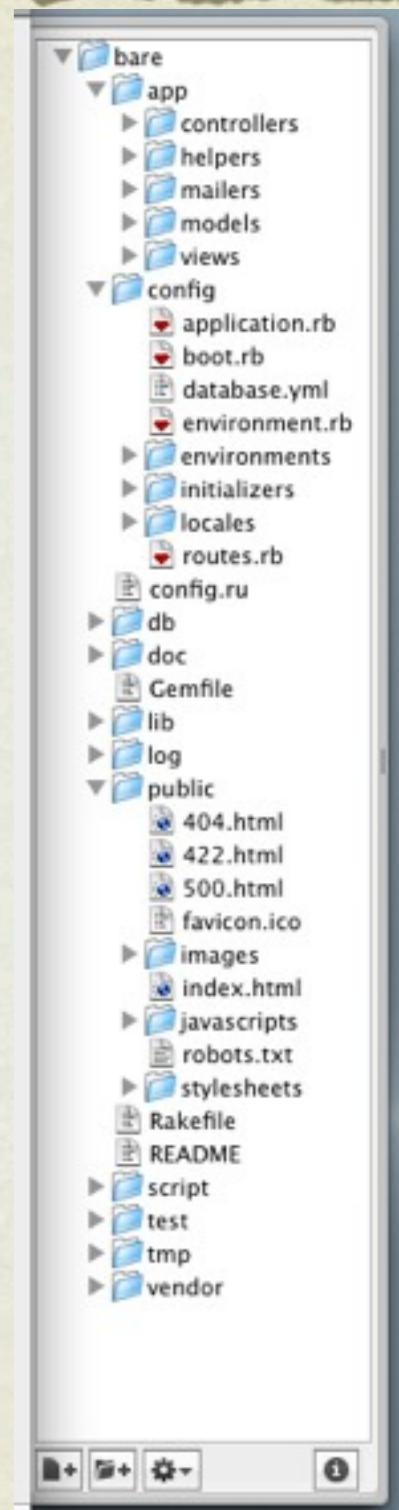
```
$ cd bare  
$ bundle install # run whenever we change Gemfile  
$ rake db:migrate # run whenever we change migrations  
$ rails s # stop and run if we change configs (especially initializers)
```

Go to <http://127.0.0.1:3000>

Reference: http://edgeguides.rubyonrails.org/3_0_release_notes.html

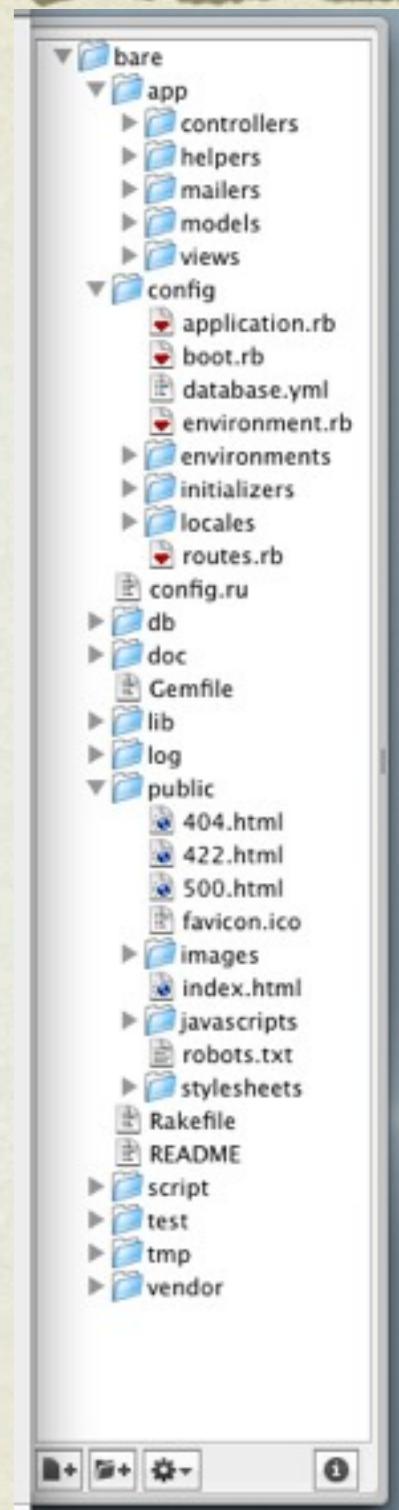
FOLDERS & FILES

- App is where your main body of code goes
 - MVC stuff (in each of the respective folders)
 - helpers methods is code available in views to keep your code clean and dry
 - mailers are a special type of views (e.g. for email)
- config is where your app params and init stuff goes
 - application: general setup stuff
 - database: your DBs
 - environment: development, test, production
 - routes: where your URLs end up



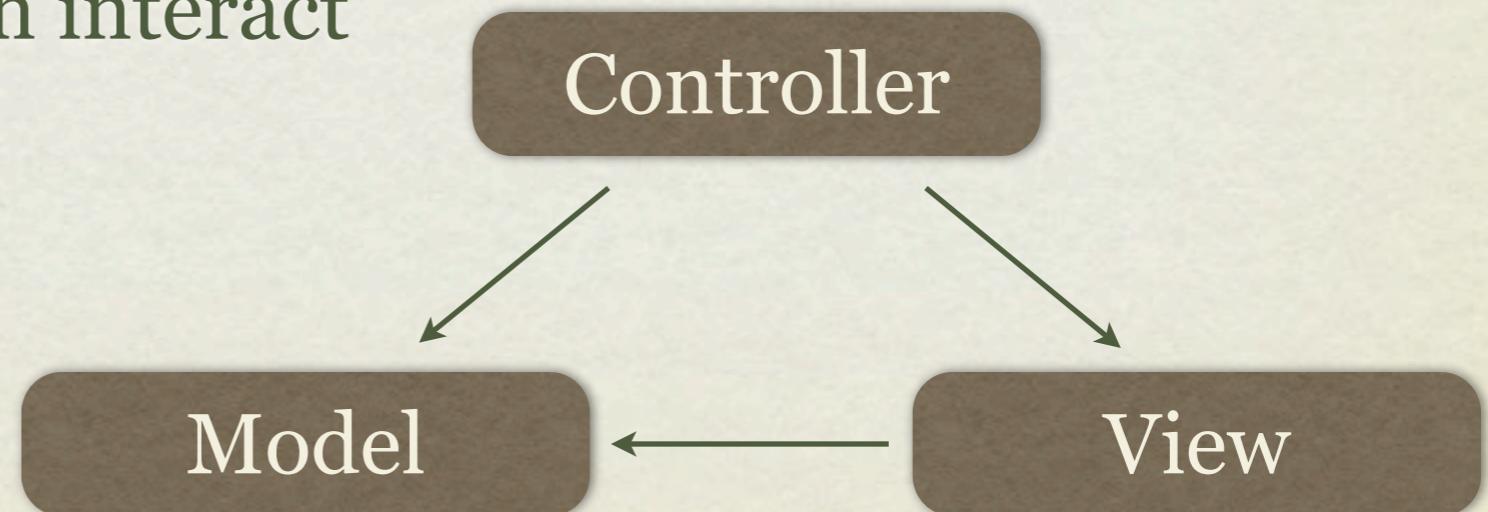
FOLDERS & FILES

- Gemfile: all your dependencies (gems)
- DB
 - where your migrations and, if you choose to use sqlite, db files go
 - also some seeding data might go here
- public
 - your static content goes here
 - your javascript and css as well
- vendor
 - plugins and other code which is not directly of



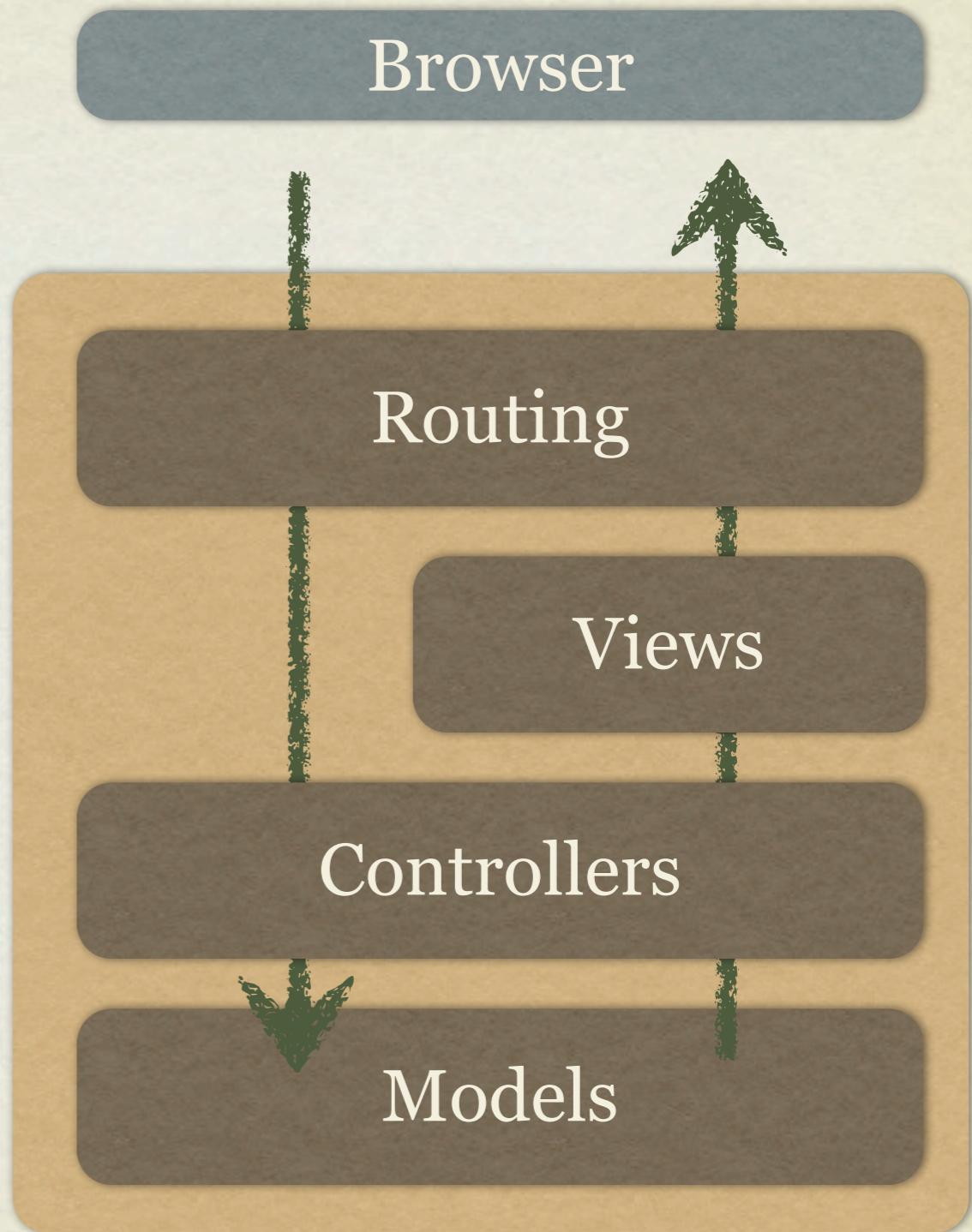
MVC

- Keep the code clean and domain independent
- The idea is that you can truly replace one part with minor modifications to another
- Model: describes your data
- Controller: is the logic behind your app
- View: the frontend side, how does it look for the user and how you can interact



ROUTING IN RAILS

- Routing decides which controller should receive the request
- The controller sets up the models (accesses data) and calls the view
- The view renders the data and gives control back to the routing, which returns the results to the browser



RACK & METAL

- Rack middleware are the pieces that constitute rails
- One calls the next, in a chain. The request is passed by from mw to mw and the response is built on the way back
- Metal is a subset of Rack which allows the coder to wrap around certain functionality (e.g. for performance)

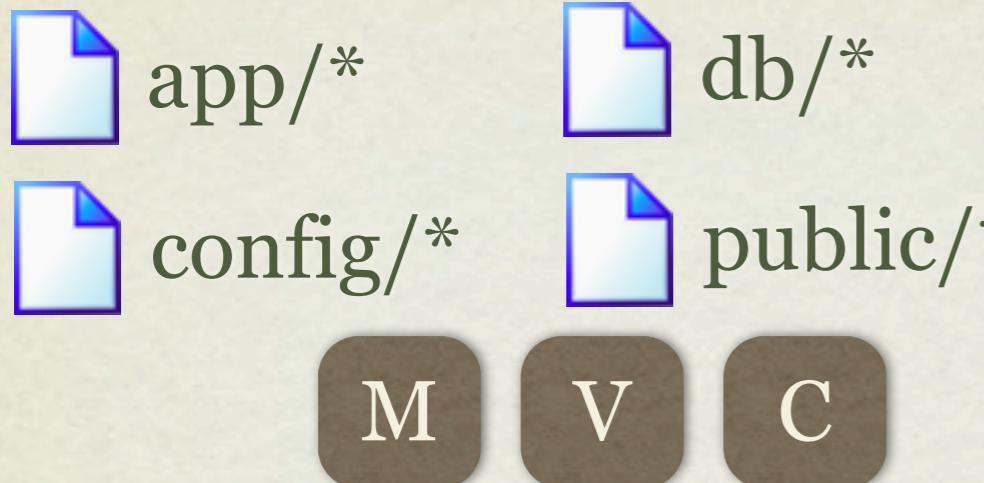
```
$ rake middleware
use ActionDispatch::Static
use Rack::Lock
use ActiveSupport::Cache::Strategy::LocalCache
use Rack::Runtime
use Rails::Rack::Logger
use ActionDispatch::ShowExceptions
use ActionDispatch::RemoteIp
use Rack::Sendfile
use ActionDispatch::Callbacks
use ActiveRecord::ConnectionAdapters::ConnectionManagement
use ActiveRecord::QueryCache
use ActionDispatch::Cookies
use ActionDispatch::Session::CookieStore
use ActionDispatch::Flash
use ActionDispatch::ParamsParser
use Rack::MethodOverride
use ActionDispatch::Head
use ActionDispatch::BestStandardsSupport
run Bare::Application.routes
```

http://guides.rubyonrails.org/rails_on_rack.html

RECAP

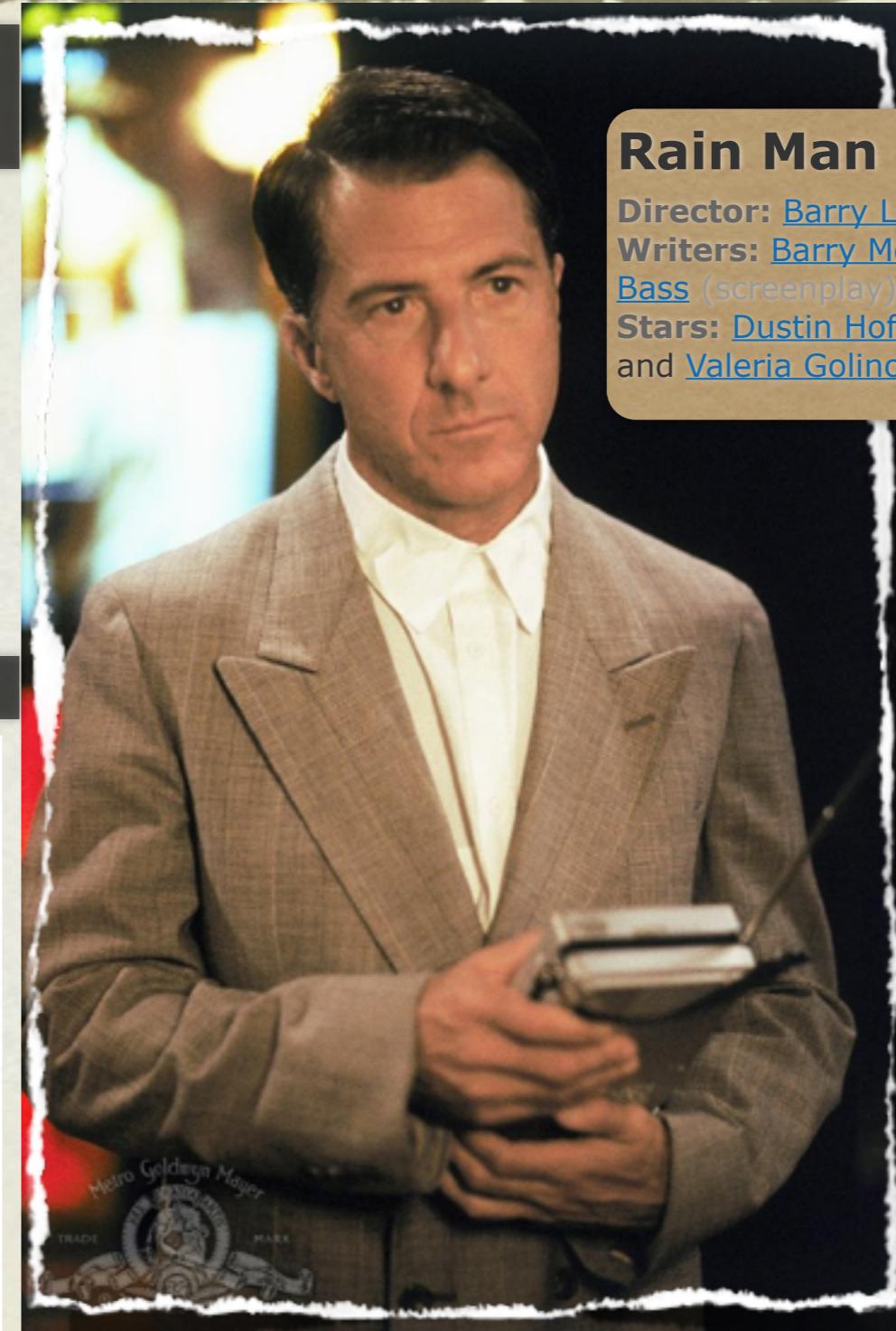


```
$ gem install rails  
$ rails new projectname  
$ rails s
```



```
$ rake middleware
```

Middleware	Purpose
Rack::Lock	Sets env["rack.multithread"] flag to true and wraps the application within a Mutex.
ActionController::Failsafe	Returns HTTP Status 500 to the client if an exception gets raised while dispatching.
ActiveRecord::QueryCache	Enable the Active Record query cache.
ActionController::Session::CookieStore	Uses the cookie based session store.
ActionController::Session::MemCacheStore	Uses the memcached based session store.
ActiveRecord::SessionStore	Uses the database based session store.
Rack::MethodOverride	Sets HTTP method based on _method parameter or env["HTTP_X_HTTP_METHOD_OVERRIDE"].
Rack::Head	Discards the response body if the client sends a HEAD request.



Rain Man ([1988](#))

Director: [Barry Levinson](#)
Writers: [Barry Morrow](#) (story), [Ronald Bass](#) (screenplay), and 1 more credit »
Stars: [Dustin Hoffman](#), [Tom Cruise](#) and [Valeria Golino](#)

CLAP, YOU MADE IT



```
3.times do
  3.times do puts "*knock*"
  puts "Penny!"
end
```



too far?



```
def setup
  200.times do |i|
    container.dispatcher.begin_invoke do |s,e|
      my_square(15)
    end
  end
end

def draw
  container.children.each do |obj|
    obj.tag.update(obj) if obj.tag
  end if @bounce_on
end

def my_square(length)
  size = rand(dimensions.max / length)
  container.children.add(square(
    :size => size,
    :fill => rand_brush,
    :stroke => rand_brush,
    :stroke_thickness => rand(dimensions.max / (length * 7)) + 4,
    :left => rand(dimensions.first - size),
    :top  => rand(dimensions.last - size),
    :tag => Bouncer.new(rand(10)-5, rand(10)-5, container)
  ))
end
```

runs only on page-load

runs for each frame

adds a square to the sketch

CHALLENGE ACCEPTED!

A PUBLICATION DB

- <http://info.office.hd>
- Publication Process
- OMG... no wonder ctrl+c
ctrl+v is so popular

My favorite part:

“... and then a courier pigeon will take your paper to Hogwarts where a team of trained goblins will spellcheck each letter individually ...”



NLE needs YOU!

IF YOU BUILD IT...

... they will come.

- DB Tables + Models
- Automatic Processes
- Authentication (LDAP)
- Email / Integration



Field of Dreams ([1989](#))

Director: [Phil Alden Robinson](#)

Writers: [W.P. Kinsella](#) (book), [Phil Alden Robinson](#)(screenplay)

Stars: [Kevin Costner](#), [James Earl Jones](#) and [Ray Liotta](#)





MODELS

Part 3

GOALS



- Models & Migrations
- Associations
- Add validations
- Add functions
- Callbacks



MODEL & MIGRATION

Let's use a generator



publicationdb

```
$ rails g scaffold paper title:string abstract:text submitted:date accepted:date rejected:boolean  
$ rake db:migrate
```

Can you guess what 'rake db:drop' does?

Looking at the migration



db/migrate/20110531022112_create_papers.rb

```
class CreatePapers < ActiveRecord::Migration
  def self.up
    create_table :papers do |t|
      t.string :title, :null => false
      t.text :abstract
      t.date :submitted
      t.date :accepted
      t.boolean :rejected, :default => false

      t.timestamps
    end
  end

  def self.down
    drop_table :papers
  end
end
```

MODEL & MIGRATION

Playing around with the model



app/models/paper.rb

```
$ rails c
```

```
> p1 = Paper.new # this will give an error... remember, title cant be null
> p1 = Paper.new :title => "500 Famous tweets"
=> #<Paper id: nil, title: "500 Famous tweets", abstract: nil, submitted: nil, accepted: nil,
rejected: false, created_at: nil, updated_at: nil>
> p1.save
=> true
> p2 = Paper.create :title => "First days of Linkedin stock", :abstract => "This paper describes
the chaos and bla bla bla...."
=> #<Paper id: 2, title: "First days of Linkedin stock", abstract: "This paper describes the chaos
and bla bla bla....", submitted: nil, accepted: nil, rejected: false, created_at: nil, updated_at:
nil>
> Paper.all # Paper.first
#...
> Paper.limit(1)
> Paper.find_by_title("First days of Linkedin stock")
#...
> Paper.where(:rejected => false)
#...
> Paper.where(:rejected => true)
=> []
> Paper.where(:rejected => false).limit(1)
#...
```

where
select
group
order
limit
offset
joins
includes
lock
readonly
from
having

ASSOCIATIONS

Looking at the model



app/models/paper.rb

```
class Paper < ActiveRecord::Base
  attr_accessible :title, :abstract, :submitted, :accepted, :rejected
end
```

Let's add authors



app/models/author.rb



app/models/publication.rb

```
$ rails g scaffold author name:string email:string
$ rails g model publication author_id:integer paper_id:integer
$ rake db:migrate
```

Authors can have many publications and publications can have many authors

```
class Paper < ActiveRecord::Base
  attr_accessible :title, :abstract, :submitted, :accepted, :rejected
  has_many :publication
  has_many :authors, :through => :publication
end
```

```
class Author < ActiveRecord::Base
  has_many :publication
  has_many :papers, :through => :publication
end
```

```
class Publication < ActiveRecord::Base
  belongs_to :author
  belongs_to :paper
end
```

ASSOCIATIONS

Again, play some more

```
> a = Author.create(:name => "Stephen King")
=> #<Author id: 2, name: "Stephen King", email: nil, ...>
> a.papers << p1
> a.papers
=> [#<Paper id: 1, title: "Cinderella part III", abstract: nil, submitted: nil, accepted: nil,
rejected: false, ...>]
> p1.authors
=> [#<Author id: 2, name: "Stephen King", email: nil, ...>]

> p1.authors.limit 1 # still just a query (lazy loading)
...
> p1.authors.delete a
> p1.authors
=> []
```

Can we limit so that the same author can't co-author the same publication with himself?

VALIDATIONS

Adding some validations to prevent multiple “same” authors

```
class Publication < ActiveRecord::Base
  belongs_to :author
  belongs_to :paper

  validates_uniqueness_of :author_id, :scope => :paper_id
end
```

```
> a.papers.delete_all
> a.papers << p
#...
> a.papers << p # validation error
```

```
validates :email, :presence => true,
          :length => { :minimum => 3, :maximum => 254 },
          :uniqueness => true,
          :format => { :with => /^[^@\s]+@[?:[-a-z0-9]+\.\.]+[a-z]{2,}$/i }
```

Validations run at the moment of saving the model

validates_acceptance_of
validates_associated
validates_confirmation_of
validates_each
validates_exclusion_of
validates_format_of

validates_inclusion_of
validates_length_of
validates_numericality_of
validates_presence_of
validates_size_of
validates_uniqueness_of

FUNCTIONS

Lets add a gender column



db/migrate/20110531033256_add_gender_to_author.rb

```
$ rails g migration add_gender_to_author male:boolean  
$ rake db:migrate
```

```
class AddGenderToAuthor < ActiveRecord::Migration  
  def self.up  
    add_column :authors, :male, :boolean  
  end  
  def self.down  
    remove_column :authors, :male  
  end  
end
```

Simple enough .. but we want to be able to say “male” or “female”

```
attr_accessor :gender  
validates :gender, :inclusion => { :in => %w{male female}, :message => "is not male or female"}  
before_save :set_male_or_female  
  
protected  
  
def set_male_or_female  
  self.male = (@gender == "male")  
end
```

CALLBACKS

Callbacks

- (-) save
- (-) valid
- (1) before_validation
- (-) validate
- (2) after_validation
- (3) before_save
- (4) before_create
- (-) create
- (5) after_create
- (6) after_save
- (7) after_commit

RECAP



```
$ rails g scaffold paper title:string  
$ rake db:migrate
```

```
class CreatePapers < ActiveRecord::Migration
```

```
class Paper < ActiveRecord::Base
```

```
> Paper.create :attr => "value"
```

```
> p = Paper.first
```

```
> p = Paper.find(1)
```

```
> a = Author.create(:name => "Stephen King")  
> a.papers << p  
> a.papers.create(...)
```

```
validates_uniqueness_of :author_id, :scope => :paper_id
```

```
validates :email, :presence => true
```

```
validates :gender, :inclusion => { :in => %w{male  
female}, :message => "is not male or female"}  
before_save :set_male_or_female
```



Mars Attacks! (1996)

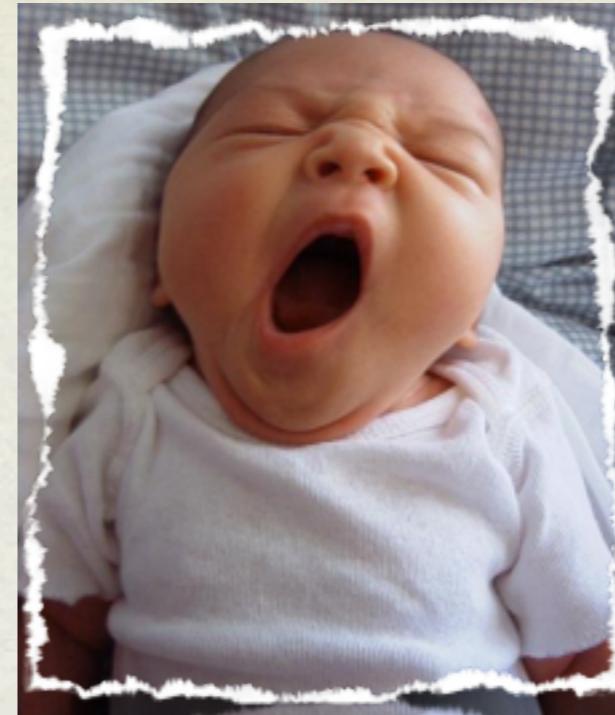


Director: [Tim Burton](#)

Writers: [Len Brown](#) (trading card series), [Woody Gelman](#)(trading card series), [and 5 more credits »](#)

Stars: [Jack Nicholson](#), [Pierce Brosnan](#) and [Sarah Jessica Parker](#)

CLAP, YOU MADE IT



You know who
you are!

By now I estimate we are at least 30 mins late...

... you! asking all the questions, it's your fault ...

...we could all be home right now.



To whom it may concern

Quis ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exercitationem ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilis et vero eros et accumsan et iusto odio dignissim qui blanditiis praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilis. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exercitationem ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilis et vero eros et accumsan et iusto odio dignissim qui blanditiis praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilis. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilis et vero eros et accumsan et iusto odio dignissim qui blanditiis praesent luptatum zzril delenit augue duis dolore te feugiat nulla facilis. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

REFERENCES

and homework

SUPERDUPER!

- Ruby in Twenty Minutes ([ruby-lang.org](http://www.ruby-lang.org)):
 - <http://www.ruby-lang.org/en/documentation/quickstart/>
- Rails for Zombies (envylabs):
 - <http://railsforzombies.org/>
- Railscasts (Ryan Bates):
 - <http://www.railscasts.com>
- The Ruby Toolbox:
 - <http://www.ruby-toolbox.com/>
- Heroku:
 - <http://www.heroku.com/>



I'm kidding...

<http://enishi.herokuapp.com>

Q&A *und el feedbaco*





This picture, shamelessly taken from the template...

NEXT TIME
Controllers, Controllers, Controllers