RAILS BASICS

INSTALL RAILS

```
$ gem install rails
```

Quit and restart your terminal when it's done (⌘ + Q)!

On Ubuntu, run as well:

```
$ sudo apt-get install nodejs
```

HISTORY

Created in 2003 by David Heinemeier Hansson, while working on Basecamp.

Extracted Ruby on Rails and released it as open source code in July of 2004

3 PRINCIPLES

- Ruby
- MVC
- Programmer happiness

RELEASES

```
1.0   December 13, 2005
1.2   January 19, 2007

2.0   December 7, 2007
2.1   June 1, 2008
2.2   November 21, 2008
2.3   March 16, 2009

3.0   August 29, 2010
3.1   August 31, 2011
3.2   January 20, 2012

4.0   June 25, 2013
4.1   April 8, 2014
4.2   December 19, 2014
```

PHILOSOPHY

- Convention over Configuration
- Don't Repeat Yourself

(both mean write less code)

GOING STRONG

More than 3400 contributors.

Code is on GitHub at rails/rails

COMMUNITY

+100k gems at rubygems.org. How do I know which one to use?

- Ask teachers
- Browse the Ruby Toolbox

HOW EVERY NEW RAILS PROJECT STARTS

CREATE A NEW RAILS APP

First, go to your personal code folder:

```
$ cd ~/code/$GITHUB_USERNAME
```

Then create a new rails app

```
$ rails new lacuillere -T
```

This creates a new folder
`~/code/$GITHUB_USERNAME/lacuillere`.

SET UP GIT

```
$ cd lacuillere
$ pwd
# => ~/code/$GITHUB_USERNAME/lacuillere
$ git init
$ git add .
$ git commit -m "Starting awesome development with Rails :)"
```

PUSH PROJECT TO GITHUB

```
$ git remote -v
# => No remotes yet! Cannot push!
```

Install the `hub` binary, with `brew install hub` (Mac) or `gem install hub` (Linux). Then, just run:

```
$ hub create
```

```
$ git remote -v
# => An `origin` remote is now set!
$ git push -u origin master  # Push the generated rails app t
$ hub browse                 # Will open your browser to the
```

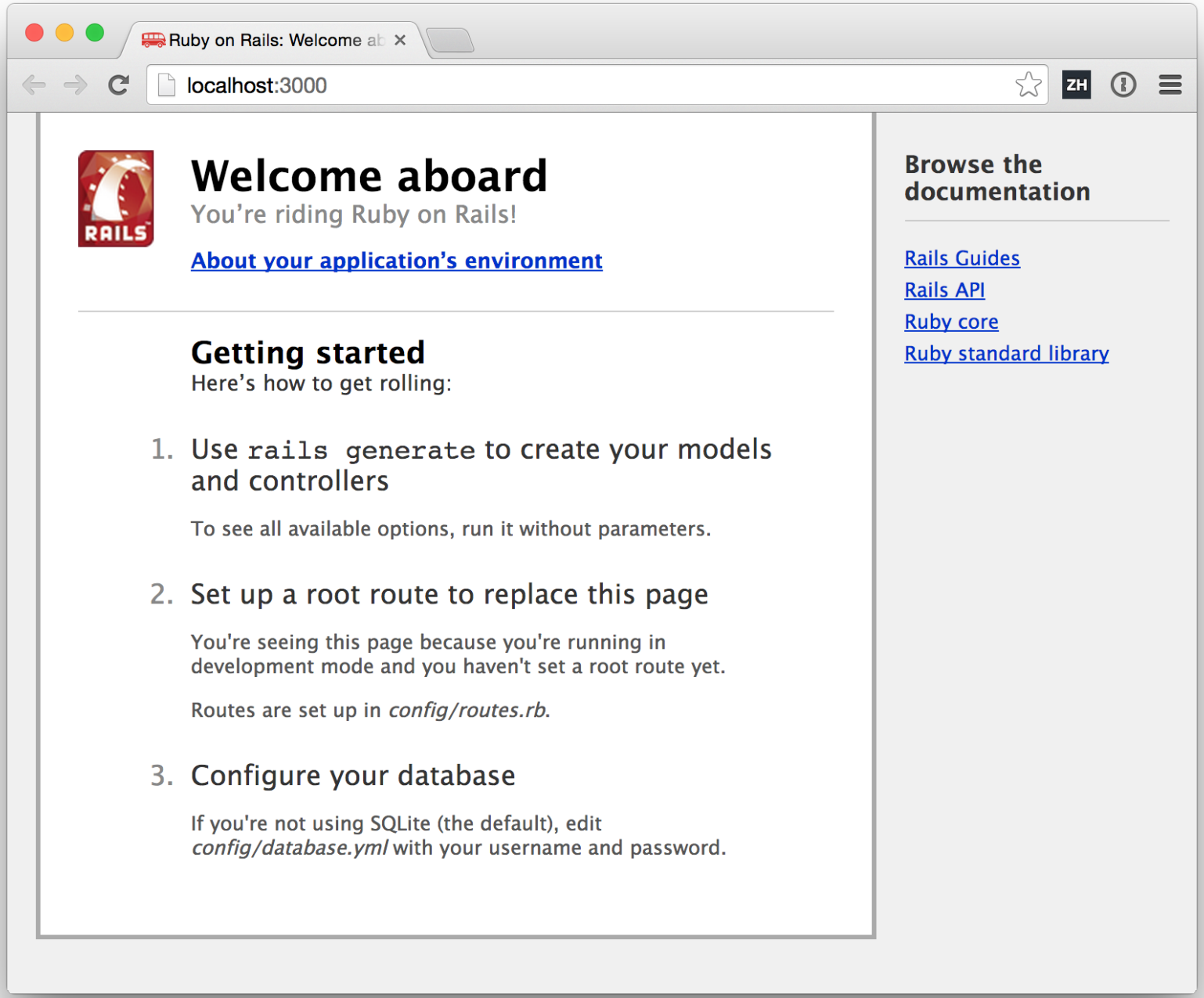
LAUNCH THE RAILS SERVER

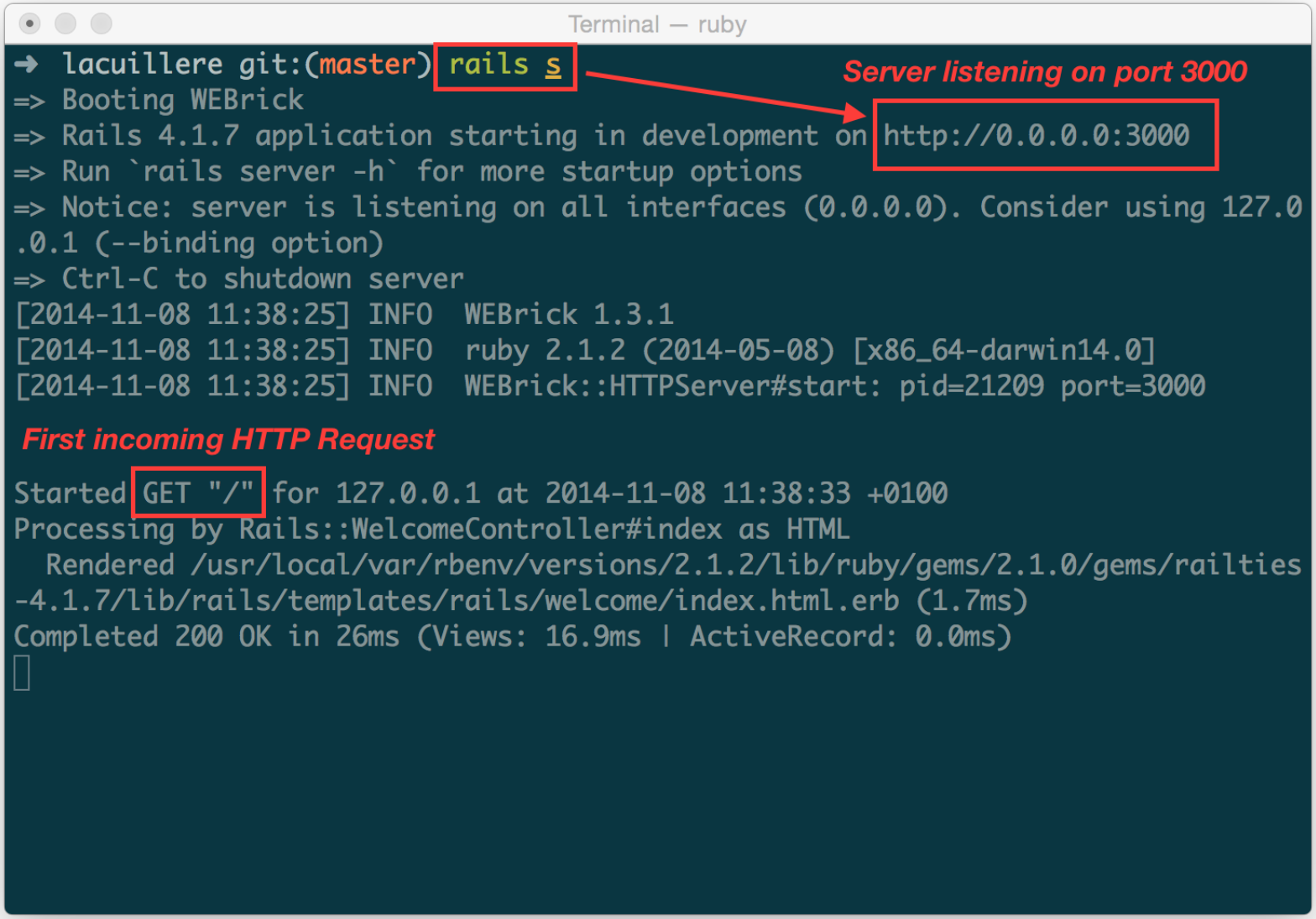
Open a **new tab** in your terminal:

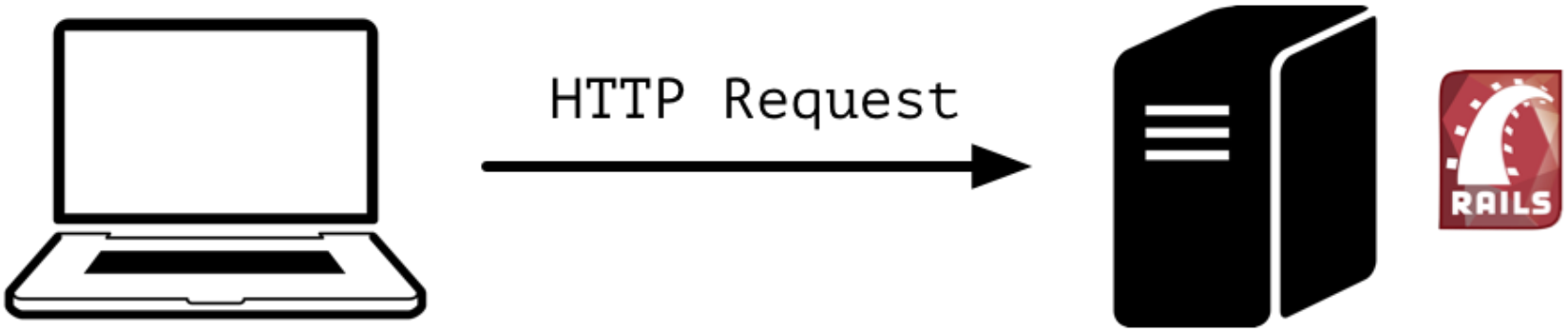
```
$ cd ~/code/$GITHUB_USERNAME/lacuillere # if not already ther
$ rails s
```

Keep this tab **opened**!

Open your terminal and go to http://localhost:3000


Ruby on Rails: Welcome ab
localhost:3000
Welcome aboard
You're riding Ruby on Rails!
About your application's environment
Getting started
Here's how to get rolling:
1. Use rails generate to create your models and controllers
To see all available options, run it without parameters.
2. Set up a root route to replace this page
You're seeing this page because you're running in development mode and you haven't set a root route yet.
Routes are set up in config/routes.rb.
3. Configure your database
If you're not using SQLite (the default), edit config/database.yml with your username and password.
Browse the documentation
Rails Guides
Rails API
Ruby core
Ruby standard library


Terminal — ruby
➜ lacuillere git:(master) rails s
Server listening on port 3000
=> Booting WEBrick
=> Rails 4.1.7 application starting in development on http://0.0.0.0:3000
=> Run `rails server -h` for more startup options
=> Notice: server is listening on all interfaces (0.0.0.0). Consider using 127.0
.0.1 (--binding option)
=> Ctrl-C to shutdown server
[2014-11-08 11:38:25] INFO WEBrick 1.3.1
[2014-11-08 11:38:25] INFO ruby 2.1.2 (2014-05-08) [x86_64-darwin14.0]
[2014-11-08 11:38:25] INFO WEBrick::HTTPServer#start: pid=21209 port=3000
First incoming HTTP Request
Started GET "/" for 127.0.0.1 at 2014-11-08 11:38:33 +0100
Processing by Rails::WelcomeController#index as HTML
Rendered /usr/local/var/rbenv/versions/2.1.2/lib/ruby/gems/2.1.0/gems/railties
-4.1.7/lib/rails/templates/rails/welcome/index.html.erb (1.7ms)
Completed 200 OK in 26ms (Views: 16.9ms | ActiveRecord: 0.0ms)




```
GET http://localhost:3000/
```

verb scheme host port path

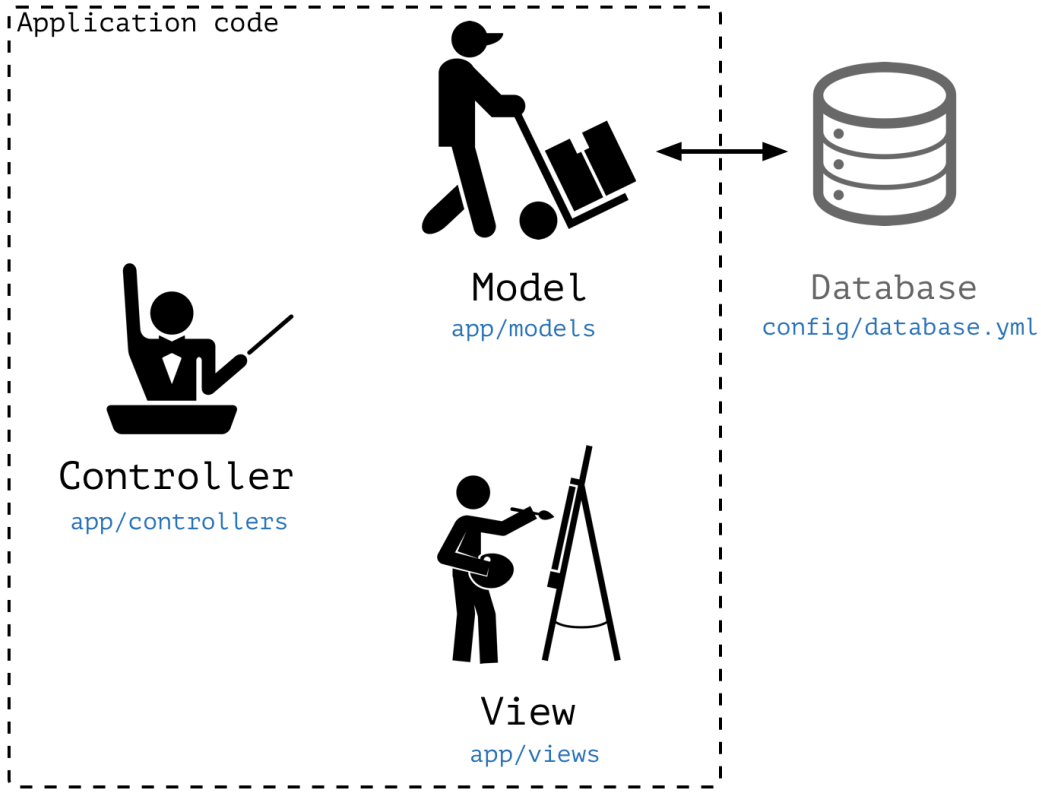
RAILS ARCHITECTURE

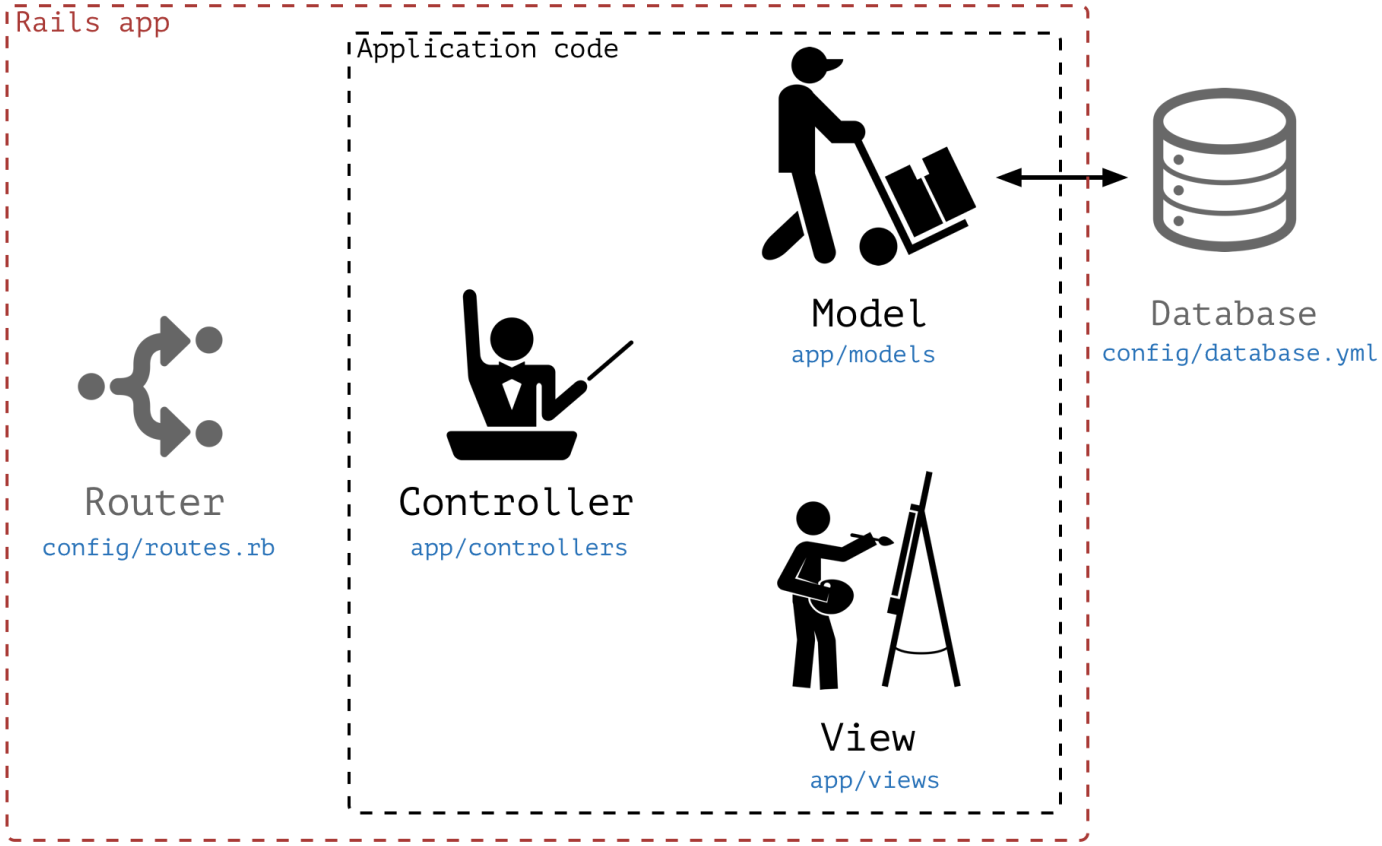
OPEN THE PROJECT IN SUBLIME TEXT:

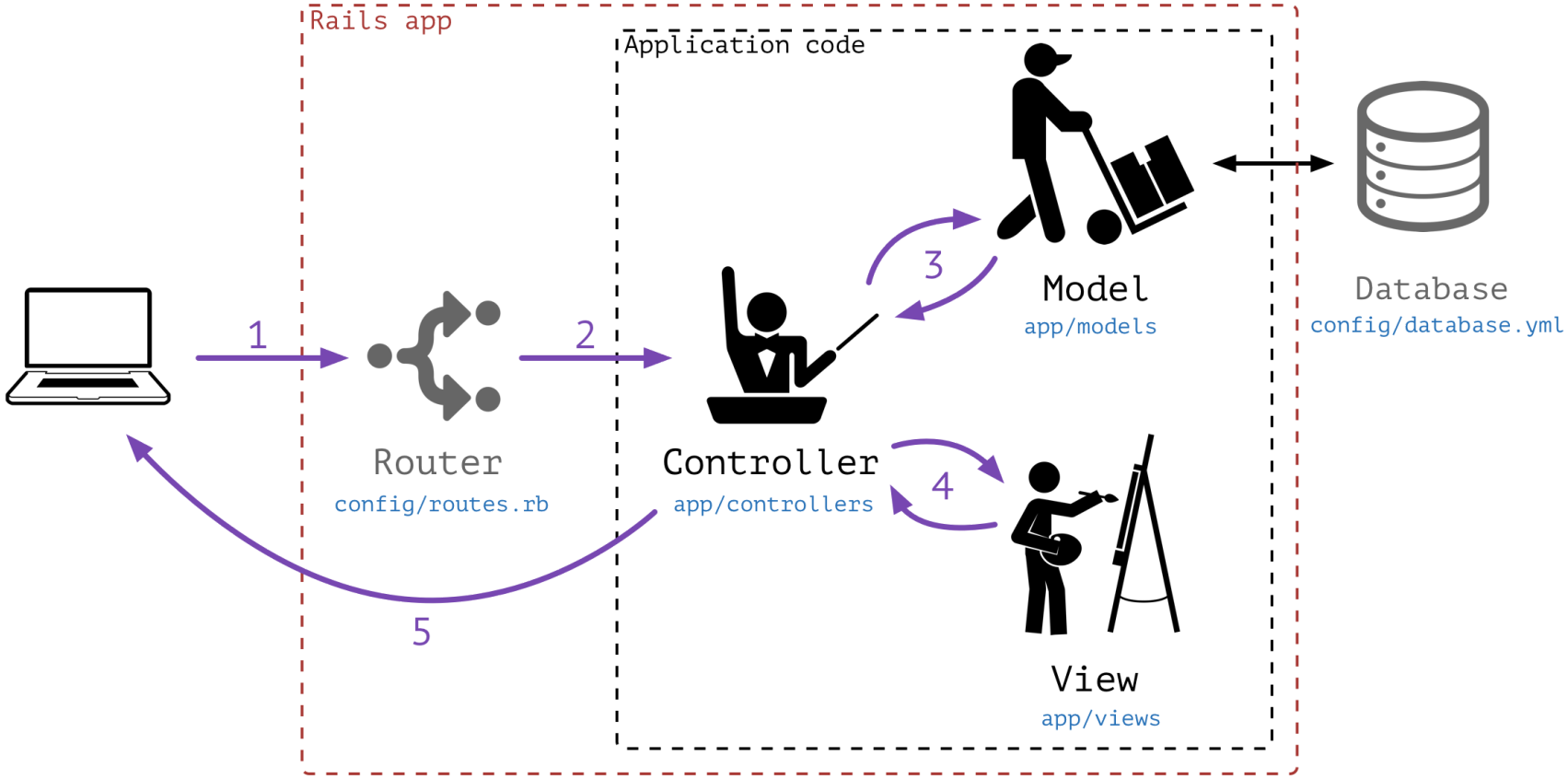
```
$ pwd
# => ~/code/$GITHUB_USERNAME/lacuillere
$ stt
```

```
.
├── app
│   ├── controllers
│   │   └── application_controller.rb
│   ├── models
│   └── views
│       └── layouts
│           └── application.html.erb
└── config
    ├── database.yml
    └── routes.rb
```

MVC REVISION


Application code
Model
app/models
Database
config/database.yml
Controller
app/controllers
View
app/views


Rails app
Application code
Model
app/models
Database
config/database.yml
Router
config/routes.rb
Controller
app/controllers
View
app/views


Rails app
Application code
Model
app/models
Database
config/database.yml
1
2
3
4
5
Router
config/routes.rb
Controller
app/controllers
View
app/views

CONTROLLER

Let's add basic pages to our app (contact, about).

We need a new controller, which we'll generate:

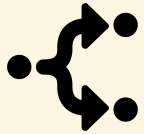
```
$ rails generate controller pages contact about
#      create  app/controllers/pages_controller.rb
#       route  get 'pages/about'
#       route  get 'pages/contact'
#      invoke  erb
#      create    app/views/pages
#      create    app/views/pages/contact.html.erb
#      create    app/views/pages/about.html.erb
```

I can now navigate to:

- http://localhost:3000/pages/contact
- http://localhost:3000/pages/about

The generator created 2 routes, you can find them in `config/routes.rb`.

```ruby
# config/routes.rb
Rails.application.routes.draw do
  get 'pages/contact'
  get 'pages/about'
end
```

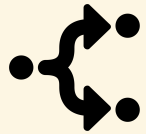


```ruby
# app/controllers/pages_controller.rb
class PagesController < ApplicationController
  def contact
  end

  def about
  end
end
```

```
.
└── app
    └── views
        └── pages
            ├── about.html.erb
            └── contact.html.erb
```

CUSTOMIZING ROUTES

```
# config/routes.rb
Rails.application.routes.draw do
  get 'about', to: 'pages#about'
  get 'contact', to: 'pages#contact'

  # Generic syntax:
  # verb 'path', to: 'controller#action' (action is an instar
end
```

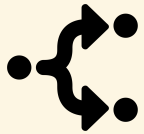


We ditched the `/pages` from the URL path:

- http://localhost:3000/about
- http://localhost:3000/contact

ROOT PATH

```ruby
# config/routes.rb
Rails.application.routes.draw do
  # [...]
  root to: 'pages#home'
end
```



```ruby
# app/controllers/pages_controller.rb
class PagesController < ApplicationController
  def home
  end
  # [...]
end
```

```
.
└── app
    └── views
        └── pages
            └── home.html.erb
```

CONVENTION OVER CONFIGURATION

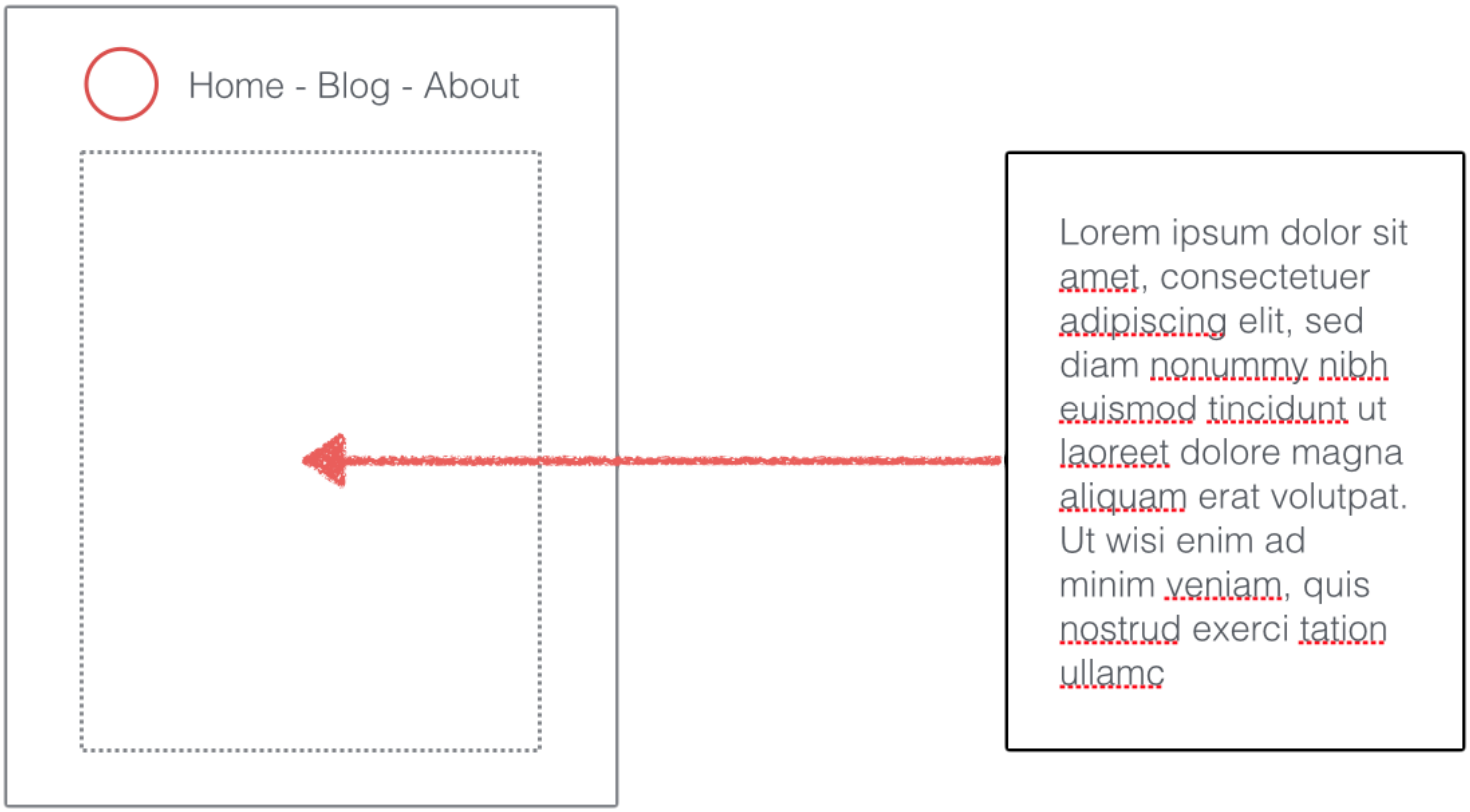
AND IF YOU GET LOST

```
$ rake routes
```

```
 Prefix Verb URI Pattern            Controller#Action
   root GET  /                      pages#home
  about GET  /about(.:format)       pages#about
contact GET  /contact(.:format)     pages#contact
```

VIEW

```
.
└── app
    └── views
        ├── layouts
        │   └── application.html.erb
        └── pages
            ├── about.html.erb
            ├── contact.html.erb
            └── home.html.erb
```


Home - Blog - About
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamc
Layout
View

```
<!-- app/views/layouts/application.html.erb -->

<!DOCTYPE html>
<html>
<head>
  <title>Lacuillere</title>
  <%= stylesheet_link_tag    'application', media: 'all', 'da
  <%= javascript_include_tag 'application', 'data-turbolinks-
  <%= csrf_meta_tags %>
</head>
<body>

<%= yield %>

</body>
</html>
```

A TYPICAL GENERATED VIEW

```
<!-- app/views/pages/home.html.erb -->
<h1>Pages#home</h1>
<p>Find me in app/views/pages/home.html.erb</p>
```

View is inserted in its layout at the line:

```
<%= yield =>
```

ERB

View files are `.html.erb` (".erb" stands for "**e**mbedded **r**u**b**y").

We will mix Ruby inside HTML.

SYNTAX

- You can write standard HTML
- You can execute ruby code inside `<% %>`
- You can execute ruby code and add it to the HTML with `<%= %>` (~ `puts`)

```
<h1>A 100%-HTML greetings!</h1>

<% name = "boris" %>

<p>Hello <%= name.capitalize %>!</p>
```

```
<h1>A 100%-HTML greetings!</h1>

<p>Hello Boris!</p>
```

GET THE DATE

```
<p>Today we are <%= Time.now.strftime("%d/%m/%Y") %><p>
```

```
<p>Today we are 8/11/2014</p>
```

LOOP

```
<% ingredients = [ 'milk', 'sugar', 'bread' ] %>

<h2>Shopping list</h2>

<ol>
<% ingredients.each do |ingredient| %>
  <li><%= ingredient %></li>
<% end %>
</ol>
```

```
<h2>Shopping list</h2>

<ol>
  <li>milk</li>
  <li>sugar</li>
  <li>bread</li>
</ol>
```

CONTROLLER <=> VIEW

CONTROLLER INSTANCE VARIABLES...

```ruby
# app/controllers/pages_controller.rb
class PagesController < ApplicationController
  def home
    @last_year = Time.now - 1.year
  end
end
```

...are accessible by the associated action view.

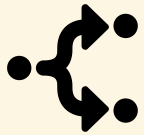
```erb
<!-- app/views/pages/home.html.erb -->
<h1>
  Last year was exactly <%= @last_year %>
</h1>
```

WHERE'S THE MODEL?

Let's anticipate tomorrow's lecture.

```ruby
# app/models/post.rb
class Post < ActiveRecord::Base
end
```

```ruby
# config/routes.rb
Rails.application.routes.draw do
  get '/posts', to: 'posts#index'
end
```



```ruby
# app/controllers/posts_controller.rb
class PostsController < ApplicationController
  def index
    @posts = Post.all
  end
end
```

```erb
<!-- app/views/posts/index.html.erb -->
<% @posts.each do |post| %>
  <h2><%= post.title %></h2>
  <p><%= post.content %></p>
<% end %>
```

Controller instance variables will often be model instances or array of model instances.

PARAMS

PARAMS COMING FROM THE QUERY STRING

```
<!-- app/views/pages/home.html.erb -->
<form action="/" method="get">
  <input type="text" name="search_term">
  <input type="submit">
</form>
```

Clicking on the submit button, the browser will make the following request:

```
GET /?search_term=something_you_typed
```

The controller can then retrieve this parameter passed in the **query string**.

```
# app/controllers/pages_controller.rb
class PagesController
  def home
    @search_term = params[:search_term]
  end
```

QUERY STRING

Everything between the ? and the # in the URL.

```
GET /some_path?first_name=alan&last_name=turing#some-facultat
```

```
# `params` is the following hash:
{
  first_name: "alan",
  last_name: "turing"
}
```

PARAMS COMING FROM THE REQUEST BODY

When do we have a request body?

POST

```
<!-- app/views/pages/home.html.erb -->
<form action="/search" method="post">
  <input type="text" name="search_term">
  <input type="submit">
</form>
```

Clicking on the submit button, the browser will make the following request:

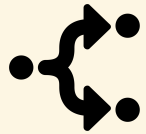
```
Header: POST /

Body: search_term=something_you_typed
```

The controller can then retrieve this parameter passed in the **request body**.

```
# app/controllers/pages_controller.rb
class PagesController
  def search
    @search_term = params[:search_term]
  end
end
```

Wait, what was the route?

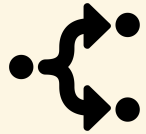
```
# config/routes.rb
Rails.application.routes.draw do
  post 'search', to: 'pages#search'
end
```



PARAMS COMING FROM THE URL PATH

Anticipation on tomorrow's lecture

```ruby
# config/routes.rb
Rails.application.routes.draw do
  get 'posts/:id', to: 'posts#show'
end
```



When the browser navigates to the following URL:

```
GET /posts/23
```

The controller can then retrieve this parameter passed in the **path**.

```ruby
# app/controllers/posts_controller.rb
class PostsController < ApplicationController
  def show
    id = params[:id]
    @post = Post.find(id)
  end
end
```

SUMMARY

The `params` hash is populated from 3 sources:

- The URL **query string** arguments
- The **body** of a `POST` request
- The URL **path** of parametric routes

LINKS

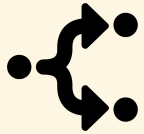
```
<a href="ANCHOR_URL">ANCHOR_TEXT</a>
```

USE LINK_TO

```
<%= link_to ANCHOR_TEXT, ANCHOR_URL %>
```

ANCHOR_URL will use a **path helper** based on the route name.

```ruby
# config/routes.rb
Rails.application.routes.draw do
  get 'about',   to: 'pages#about'
  get 'contact', to: 'pages#contact'
end
```

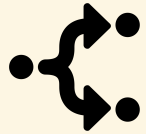


```
$ rake routes
 Prefix Verb URI Pattern         Controller#Action
  about GET  /about(.:format)    pages#about
contact GET  /contact(.:format) pages#contact
```

You can use in the view:

```ruby
<%= link_to "some anchor text", about_path %>
```

```ruby
# config/routes.rb
Rails.application.routes.draw do
  get 'bonjour', to: 'pages#welcome', as: :welcome
end
```



```
$ rake routes
 Prefix Verb URI Pattern            Controller#Action
welcome GET  /bonjour(.:format) pages#welcome
```

You can use in the view:

```ruby
<%= link_to "some anchor text", welcome_path %>
```

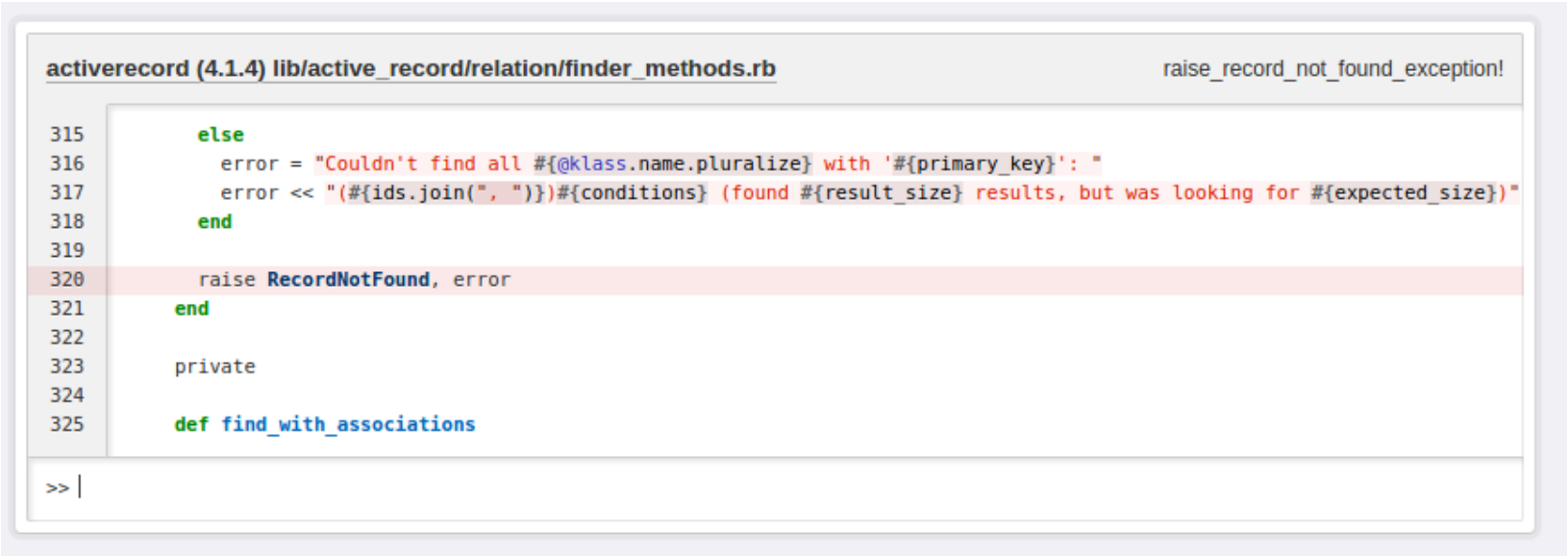
USEFUL TOOLS

BETTER ERRORS GEM

```
# Gemfile
group :development do
  gem "better_errors"
  gem "binding_of_caller"
end
```

Then run in your terminal

```
$ bundle install
```



Will basically give you an IRB in your browser each time you get an error.

Your turn! Dive into the **Routing/Controller/View** with the exercises.

Tomorrow, we'll add the **Model** layer with the well-known **ActiveRecord**!