Advanced Web Programming

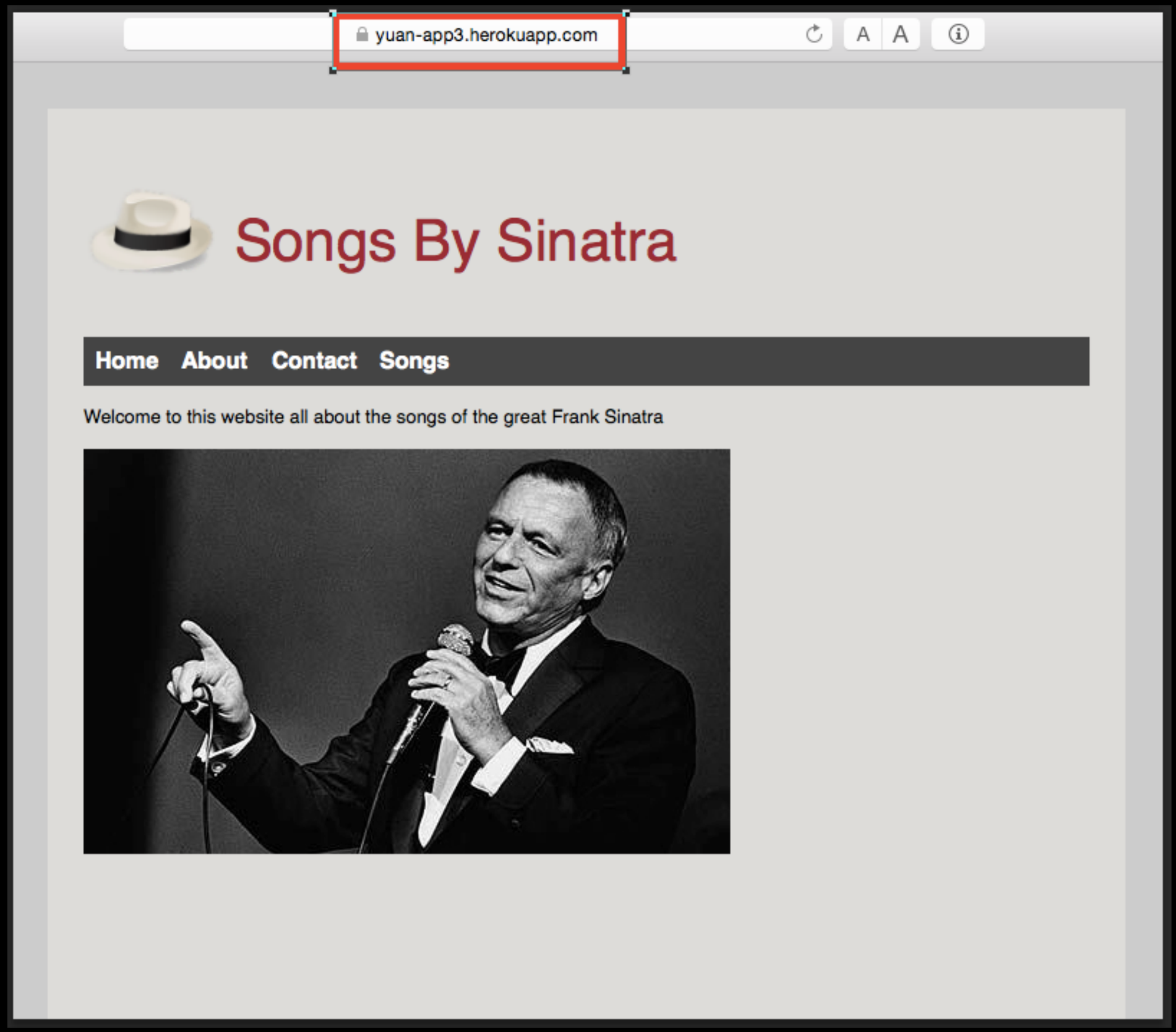
Yuan Wang

2018 Fall

Lecture 14

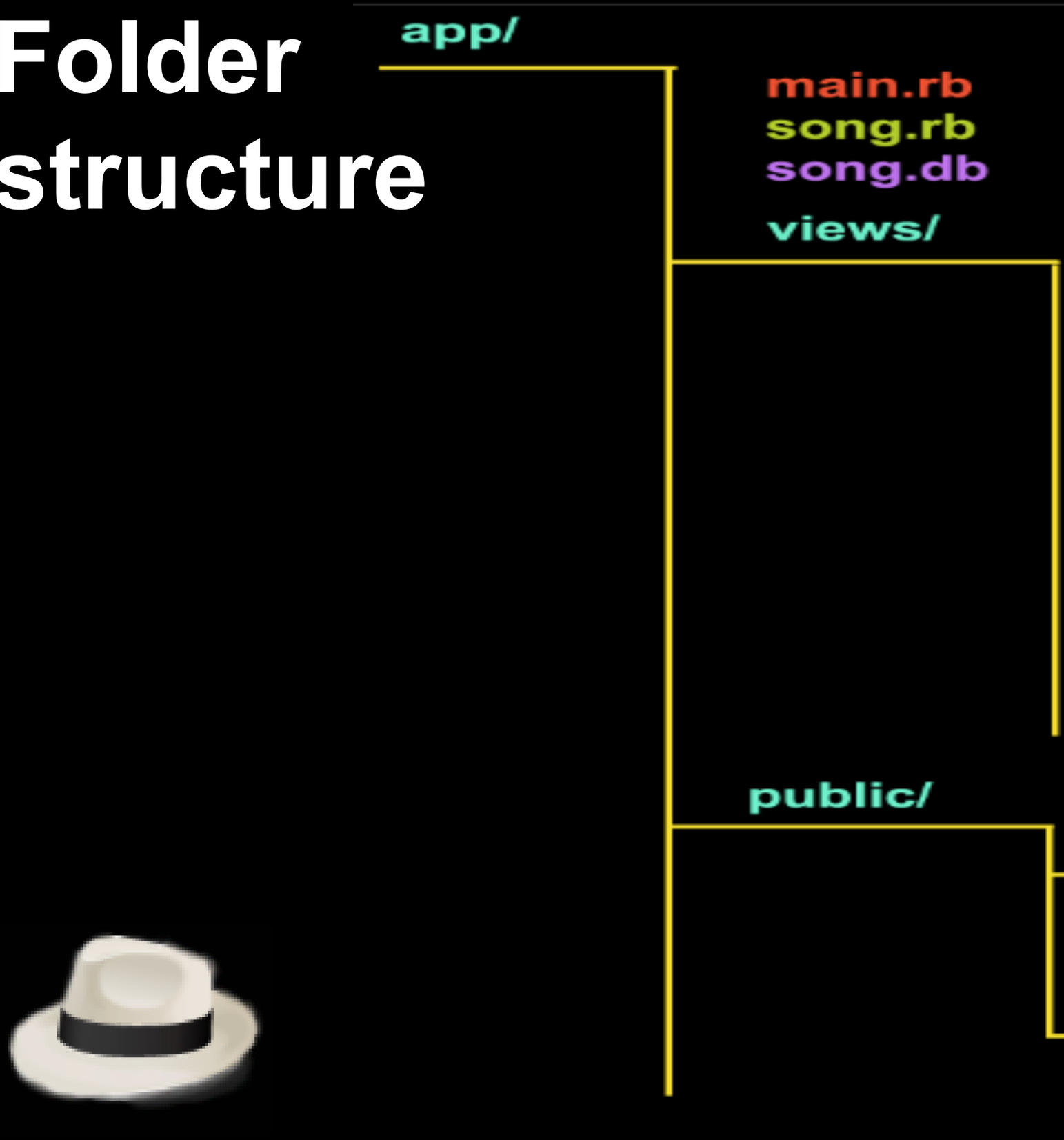
Ruby - Sinatra - RECAP

Application on Heroku



Ruby - Sinatra - RECAP

Folder structure



Today's topic:

Ruby on Rails

Most famous web application development framework

Ruby - Ruby on Rails

David Heinemeier Hansson - Creator

39 years old

Ruby - Ruby on Rails

David Heinemeier Hansson

selling pirated-CDs when he was 14

2001, while working on web based project management system tool in PHP, developed a web framework in Ruby

2004, release the framework as open source project “Ruby on Rails”

2005, “Hacker of the Year” award for creation of Ruby on Rails.

(while he was still in Copenhagen Business School)

Ruby - Ruby on Rails

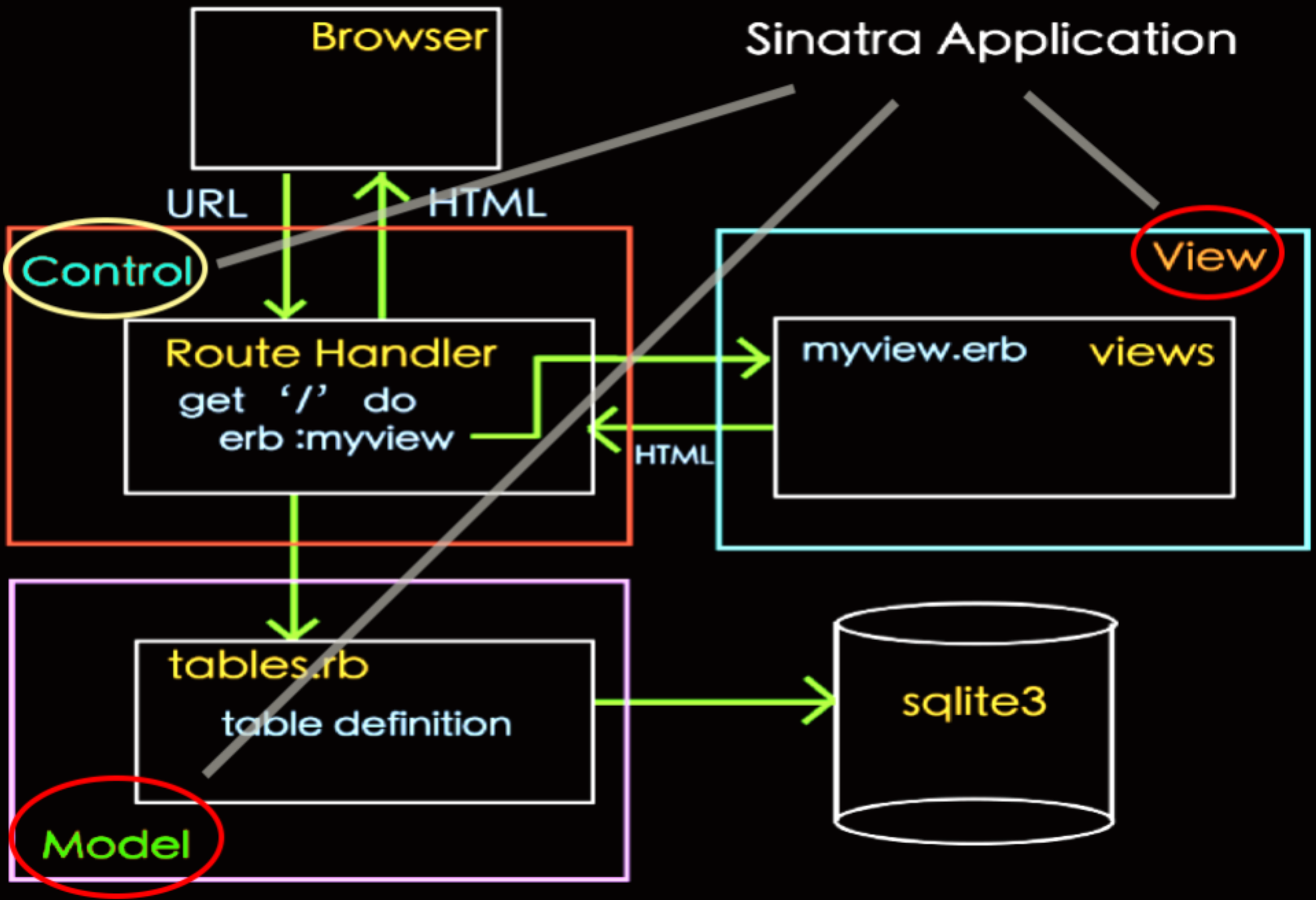
David Heinemeier Hansson
2009, Began racing sports cars

2012, SPEED magazine's Rookie of the Year award

His advice:

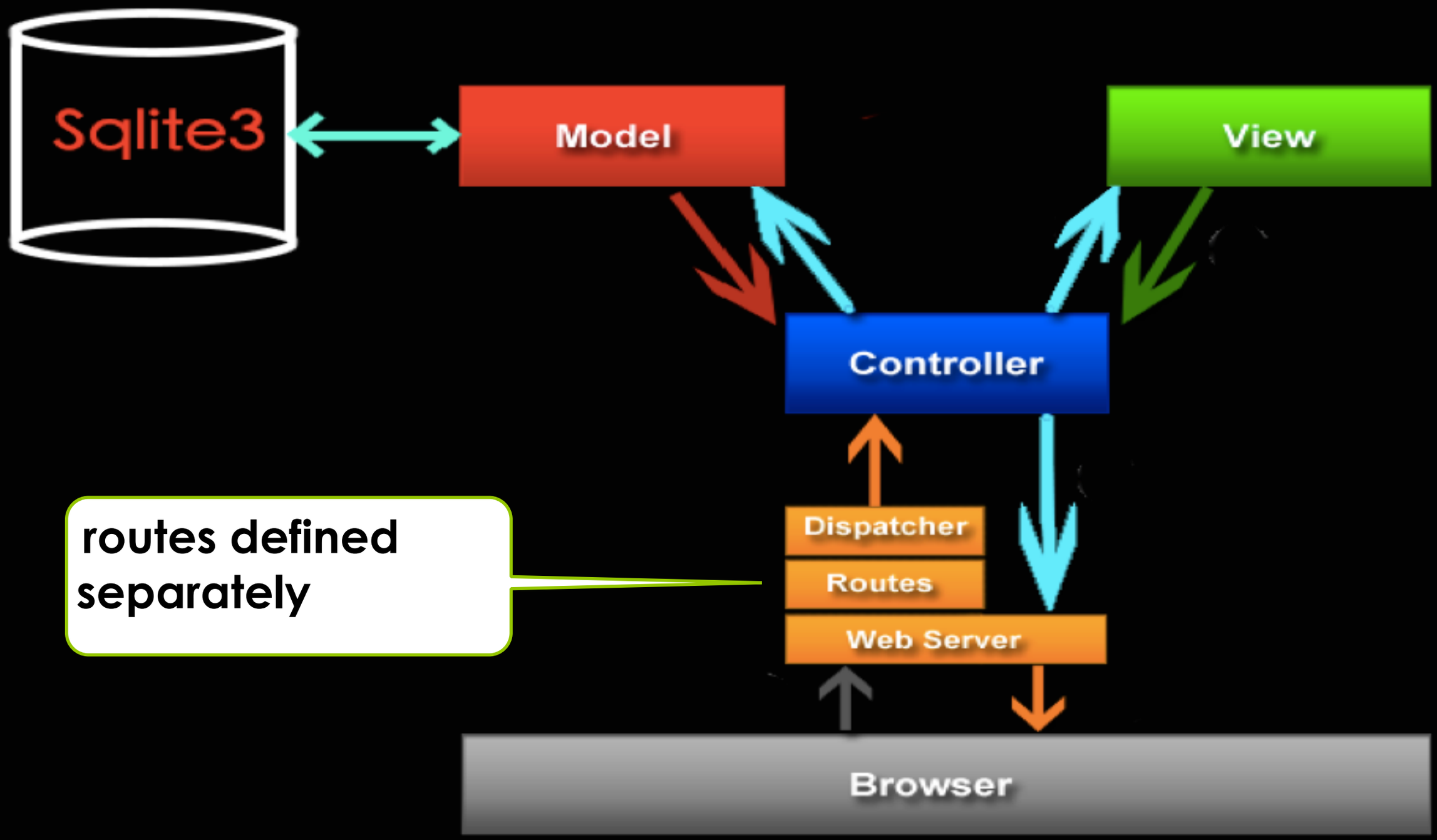
identify the problem that is frustrating you the most, and then realize that there are thousands or even millions of people out there frustrated by the same problem, and who would be willing to pay money to anyone who can solve it.

Ruby - Re-examine Sinatra



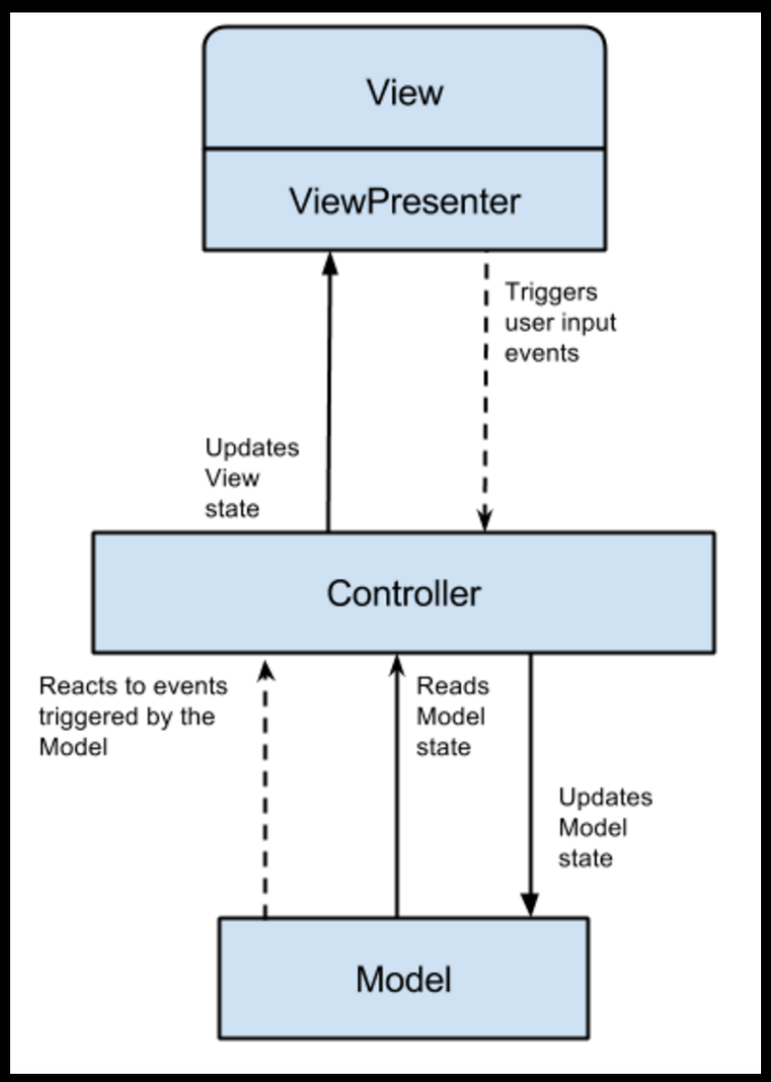
Ruby - Ruby on Rails

Ruby on Rails actually bears some similarities with Sinatra - similar MVC model



Ruby - Ruby on Rails

MVC was originally intended for desktop GUI application



facts about MVC

- A design pattern for the architecture of GUI applications.
- It works to separate data, UI and its control for a more cohesive and modularized system.
- Trygve Reenskaug formulated the MVC pattern for GUI design in 1979 while visiting the Xerox Palo Alto Research Center (PARC).
- Used in making Apple interfaces (Lisa and Macintosh).

Ruby - Ruby on Rails

MVC - Model-View-Controller

Model: database, data model,
central component
manage data, business logic

View: for data presentation or visualization
(user interface)

Controller: accepts input (request) and convert it
to the commands for the Model
or the View

Ruby - Ruby on Rails

Using Rails:

xcode command line tool
need to be installed first:

$xcode-select -install

Two options:

1. Install on local machine:

    > gem install rails

    to check version:

    > rails -v

2. Use cloud environment

    https://c9.io

    https://pro.nitrous.io

For Windows, checkout http://railsinstaller.org

Ruby - Ruby on Rails

The difference between Sinatra and Rails:
(for developing web applications)

Sinatra: basically you need to set up the whole application structure and create files manually.

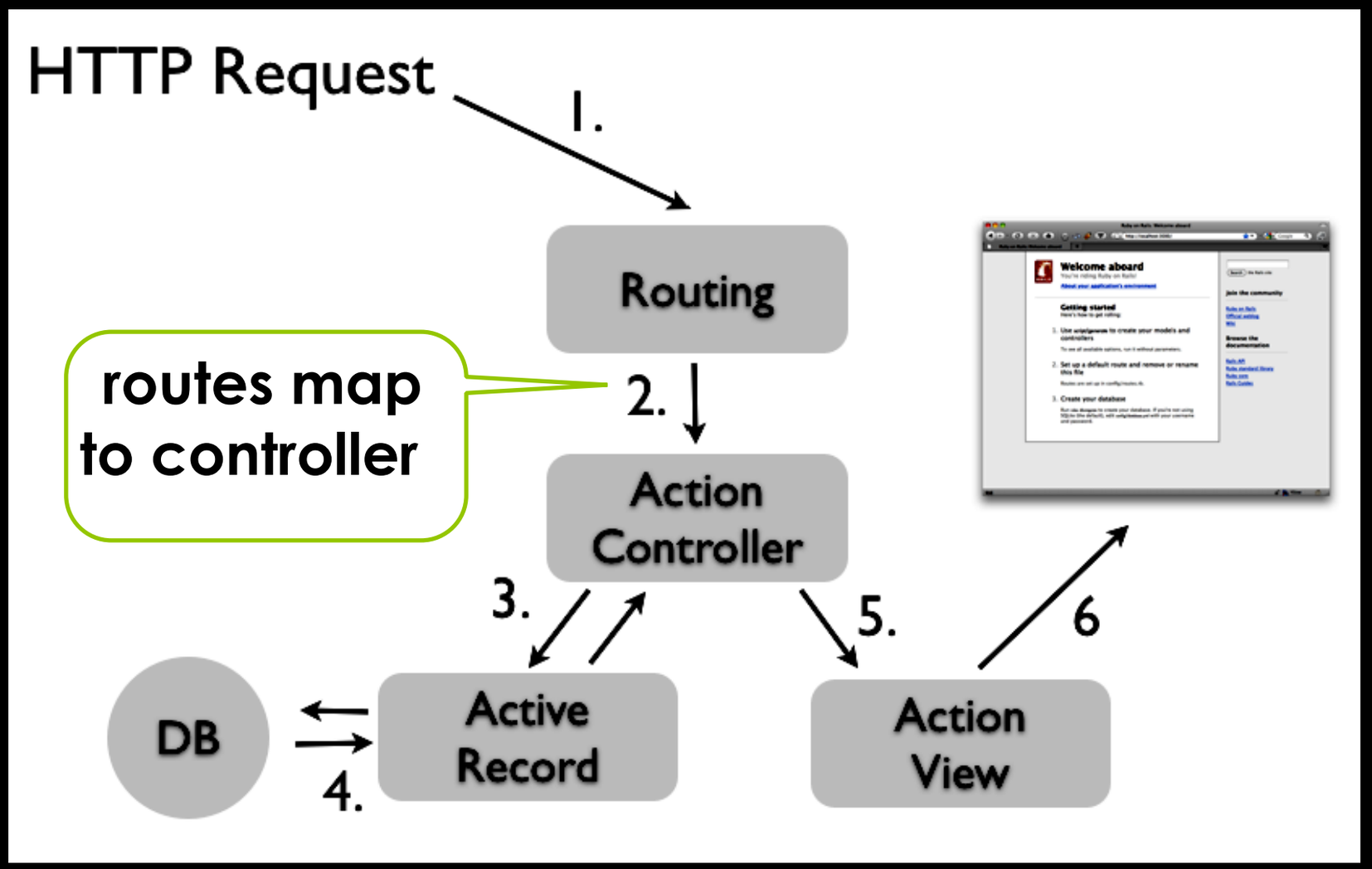
Rails: you can use the rails program to generate a default starter app for you with all the initial routes, controllers and views folders.

Ruby - Ruby on Rails

One technical difference:

Sinatra: routes are not defined separately (separate file)
routes/controllers are together

Rails: routes are defined separately (config/routes.rb)
routes are pointing to controllers, controllers are pointing to views



Ruby - Ruby on Rails

Programs you will have:

- **Puma server**
  (come with Ruby, default server for Rails 5 application
  Rails 4 was using Webrick)

- **rake: to manage task (as 'make' in UNIX)**

- **rails: to create new application.**
  to automatically generate different components of application.

  to start the rails server

Ruby - Ruby on Rails

Database:

Rails uses SQLite3 by default

Rails supported the following databases:

MySQL,
Oracle,
PostgreSQL,
SQLite,
Frontbase,
DB2
SQLServer…

ORM: ActiveRecord

Ruby - Ruby on Rails

Creating new application

```
>  rails new app1      # create an initial application
                       # with all default folders and
                       # initial code
```

To create new application using different database:

```
>  rails new app1 --database=mysql
```

With “rails new”, there are basically just folders with some initial files inside.

Ruby - Ruby on Rails

An application folder app1/ will be created with these content:

Gemfile
Gemfile.lock
README.rdoc
Rakefile
app
bin
config
config.ru
db
lib
log
public
test
tmp
vendor

routes.rb
for all route

database files

public/
404.html
422.html
500.html

app/
assets
controllers
helpers
mailers
models
views

similar to 'public' in 'Sinatra'

MVC in separate folders

table class definitions

views/
layouts/
application.html.erb

application.html.erb

```
<!DOCTYPE html>
<html>
<head>
  <title>App1</title>
  <%= stylesheet_link_tag    'application', media: 'all', 'data-turbolinks-track' => true %>
  <%= javascript_include_tag 'application', 'data-turbolinks-track' => true %>
  <%= csrf_meta_tags %>
</head>
<body>
<%= yield %>
</body>
</html>
```

Ruby - Ruby on Rails

We haven't done any coding yet, but the application is runnable, to start this new web application:

```
> rails server
=> Booting Puma
=> Rails 5.0.3 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.8.2 (ruby 2.2.3-p173), codename: Sassy Salamander
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://0.0.0.0:3000
Use Ctrl-C to stop
```

default port

Ruby - Ruby on Rails

From browser: localhost:3000

RAILS

Yay! You're on Rails!

Rails version: 5.0.3

Ruby version: 2.2.3 (x86_64-darwin14)

Ruby - Ruby on Rails

Right now, there is nothing to do except showing a default page, which is not even in our application directory (it is not “public/index.html” or something).

Lets start by adding a static page to the current application:

```
<html>
<head></head>
<body>
<h1>Something more</h1>
</body>
</html>
```

Save this file in public/ as page1.html
open browser and check it out: localhost:3000/page1

Ruby - Ruby on Rails

Convention for "routes - controller - view"

Student table

ID	Firstname	Lastname	Age

Song table (a resource)

ID	Title	Length	Release

Course table (a resource)

ID	Name	Unit	Credit

This is an entity representing certain information, we call it a "resource"

for example, Courses is a resource, Songs is a resource.

Each resource has a set of standard actions and corresponding views:

by convention, all actions is in a file called "resource"_controller.rb

under a class:
"ResourceController"

routes	student_controller.rb class StudentController	/student/
student/index route	index - list all rows	index view
student/new route	new - create new row	new view
student/show route	show - display a row	show view
student/edit route	edit - edit a row	edit view
student/delete route	delete - delete a row	delete view

all views are organized under a folder name:

"resource"

routes is in the format of "resource"/actions

Ruby - Ruby on Rails

How can we create something similar to Sinatra, with route handlers and ERB views?

For example, o create 2 pages (2 views) in the application

— see demo

Ruby - Ruby on Rails

Rails can create the structure automatically:

For example, if you want to create 2 pages (2 views) in the application, it can be easily done like this:

```
> rails generate controller Home page1 page2
```

detail running output next slide

Ruby - Ruby on Rails

```
> rails generate controller Home page1 page2
      create  app/controllers/home_controller.rb
       route  get 'home/page2'
       route  get 'home/page1'
      invoke  erb
      create    app/views/home
      create    app/views/home/page1.html.erb
      create    app/views/home/page2.html.erb
      invoke  test_unit
      create    test/controllers/home_controller_test.rb
      invoke  helper
      create    app/helpers/home_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/home.coffee
      invoke    scss
      create      app/assets/stylesheets/home.scss
```

Several new folders are added

Compared with Sinatra, the application structure is not difficult to make sense

Ruby - Ruby on Rails

First check out the controller created, the app/controllers directory has one more file created:

app/controllers/home_controller.rb

```
class HomeController < ApplicationController
  def page1
  end

  def page2
  end
end
```

HomeController inherits from ApplicationController, which inherits from ActionController
The two methods (actions) are similar to route handler in Sinatra, they will call the corresponding views

notice the name convention here, HomeController and home_controller.

we will see lots of “name convention” later. they are very important this is one of the Rails philosophy: Convention over Configuration

Ruby - Ruby on Rails

Views (templates) will be called automatically

Rails searches the directory app/views, look for subdirectory with the same name as the controller (home), and within that directory, look for a file named after the actions (page1, page2):

page1.html.erb

page2.hmtl.erb

```
Gemfile
Gemfile.lock
README.rdoc
Rakefile
app/
  controllers/
    home_controller.rb
  views/
    home/
      page1.html.erb
      page2.html.erb
bin
config/
  routes.rb
config.ru
db
lib
```

Ruby - Ruby on Rails

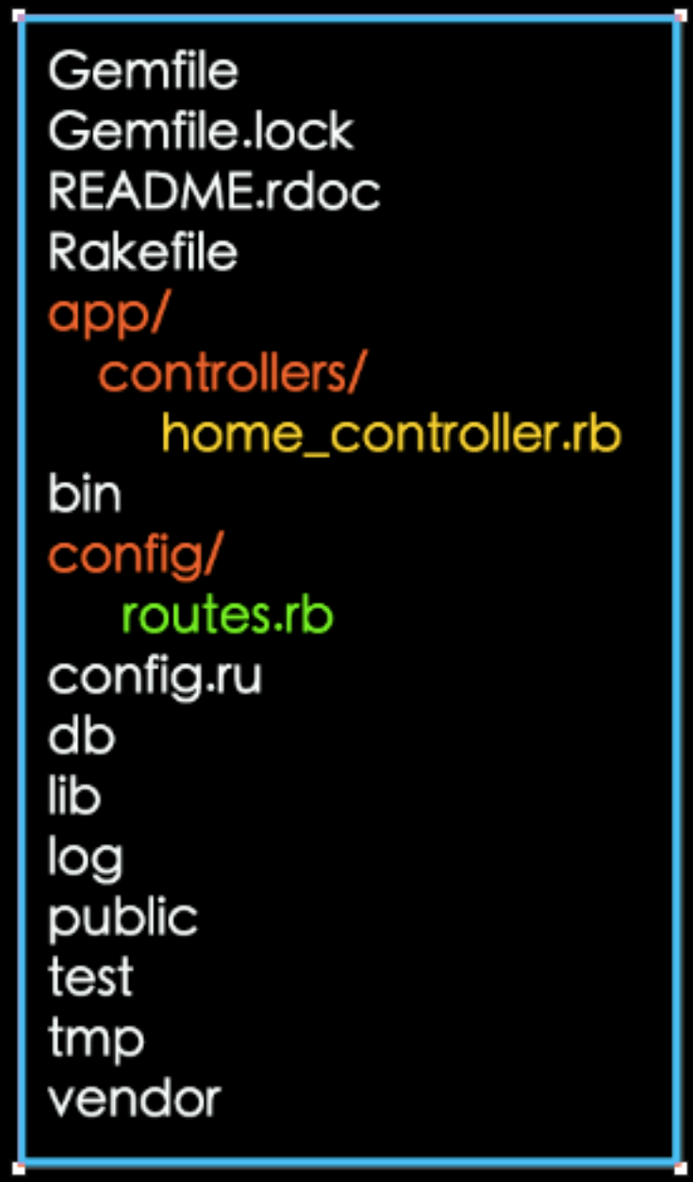
Unlike Sinatra, where routes and handler are together, in Rails, routes and handlers are defined separately

routes are defined in:

config/routes.rb

handlers are defined in controllers:

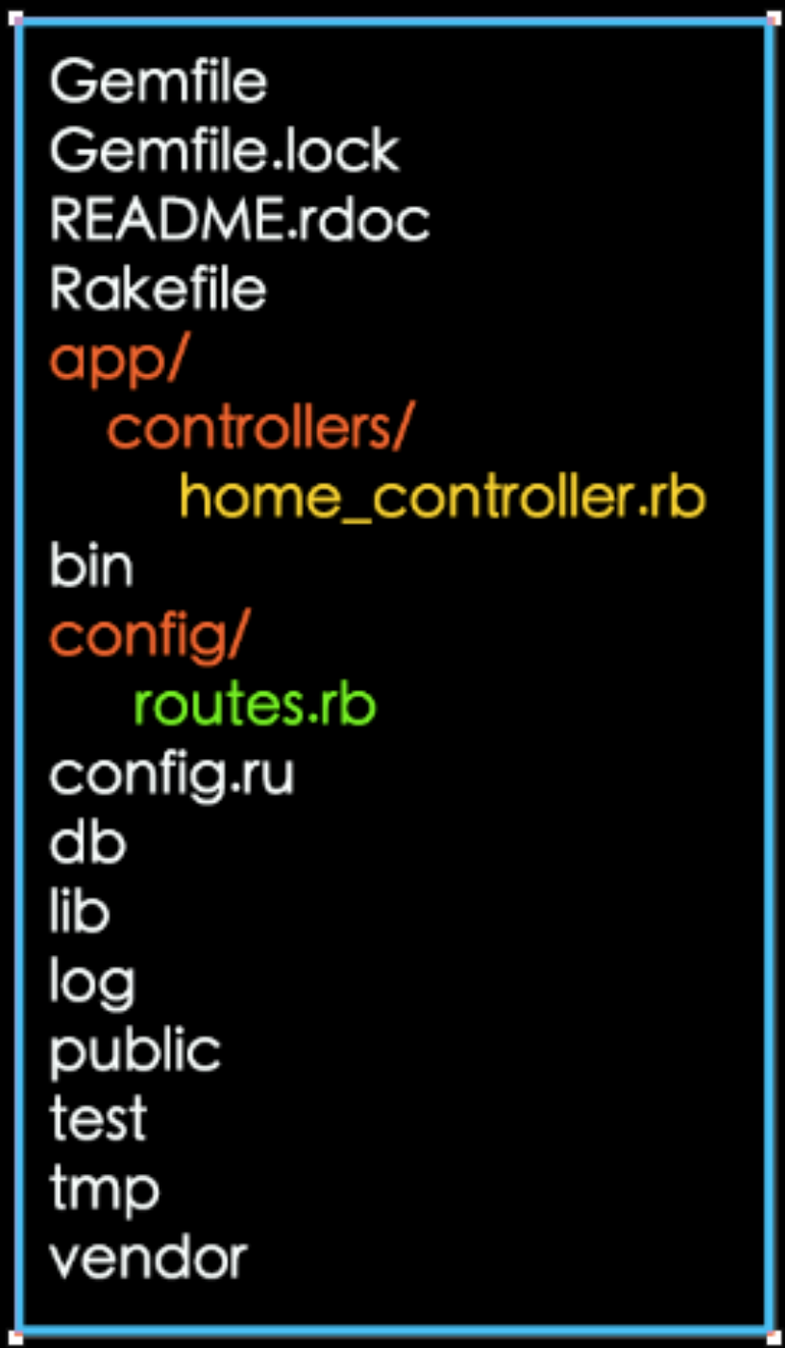
app/controllers/home_controller.rb



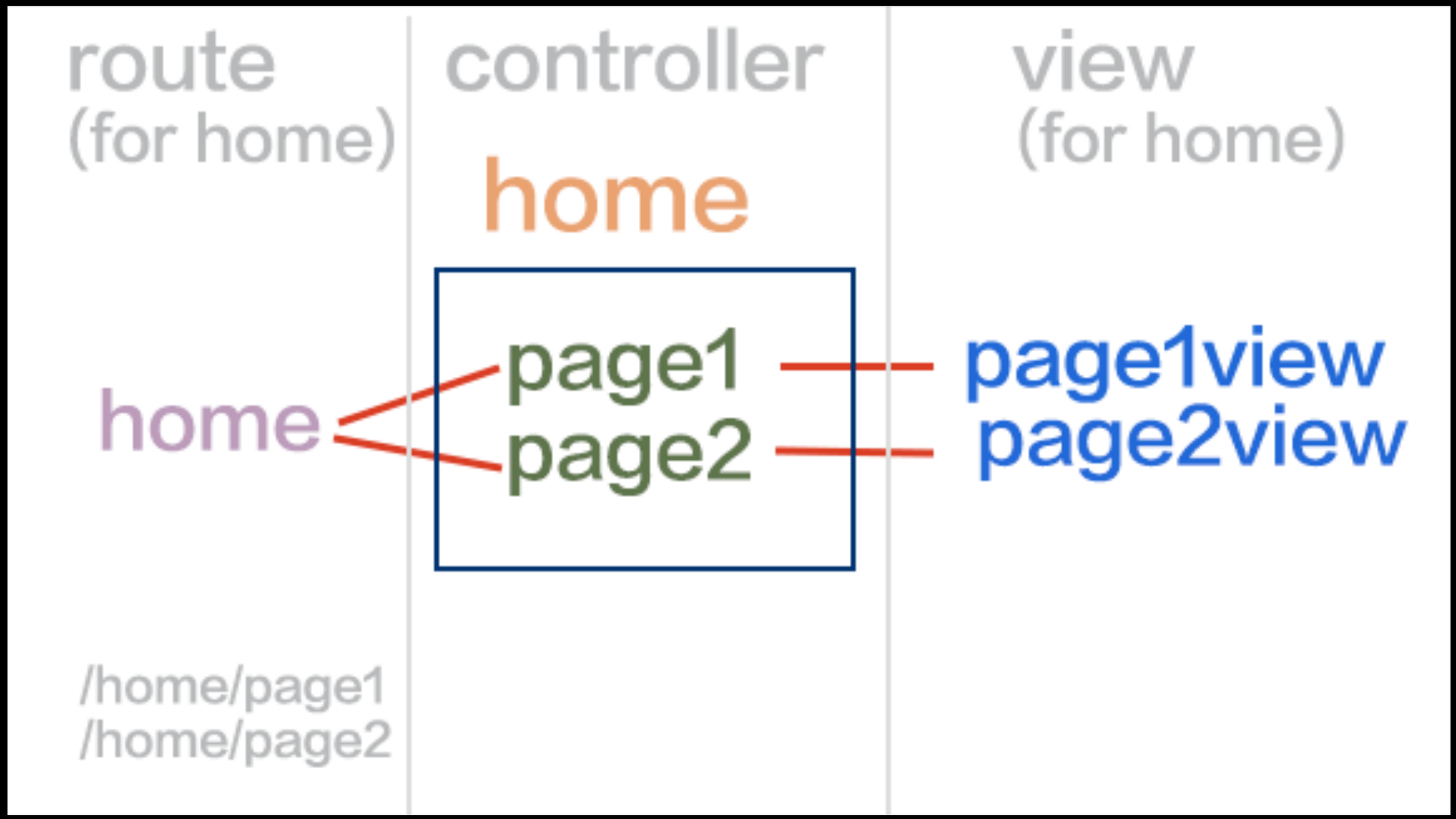
Ruby - Ruby on Rails

Lets check out routes.rb

```
Rails.application.routes.draw do

  get 'home/page1'

  get ‘home/page2'

end
```



Ruby - Ruby on Rails



Ruby - Ruby on Rails

How do they all related?

#to check the routes:

> rails routes

```
    Prefix        Verb   URI   Pattern                 Controller#Action
home_page1  GET   /home/page1(.:format) home#page1
home_page2  GET   /home/page2(.:format) home#page2
```

Rails has this mapping from url to controller actions and then to views.

Ruby - Ruby on Rails

Lets look at the views created by Rails:

page1.html.erb

```
<h1>Home#page1</h1>
<p>Find me in app/views/home/page1.html.erb</p>
```

page2.html.erb

```
<h1>Home#page2</h1>
<p>Find me in app/views/home/page2.html.erb</p>
```

Ruby - Ruby on Rails

Check it out from browser

Ruby - Ruby on Rails

Make modifications to the controllers and views.

First, as Sinatra, we pass value from controller to views by instance variables:

home_controller.rb

```
class HomeController < ApplicationController
  def page1
    @time = Time.now
  end

  def page2
  end
end
```

Ruby - Ruby on Rails

In the views: - hard code the links

page1.html.erb

```
<h1>Hello from page1</h1>
<p>It is <%=@time %> now</p>
<a href="/home/page2>page2</a>
```

page2.html.erb

```
<h1>Hello from page2</h1>
<p>It is <%=@time %> now</p>
<a href="/home/page1>page1</a>
```

Ruby - Ruby on Rails

Helper methods are methods that are create for us to create HTML easier.

using helper “link_to()” method:

page1.html.erb
```
<h1>Hello from page1</h1>
<p>It is <%=@time %> now</p>
<%= link_to “page2”, home_page2_path %>
```

page2.html.erb
```
<h1>Hello from page2</h1>
<p>It is <%=@time %> now</p>
<%= link_to “page1”, home_page1_path %>
```

home_page1_path and home_page1_path return values Rails make available to the application views,

in this example, they evaluate to

/home/page1
and
/home/page2

Ruby - Ruby on Rails

partials

some common pieces of ERB code can be included into different templates. for example, both page1 and page2 have a footer like this:

```
<hr>
<p>Copyright 2010-<%= Date.today.year %> Web2</p>
```

save this file in the same directory as “_footer.html.erb”

note: must start with “_”.

Ruby - Ruby on Rails

partials (continue)

Then in page1 and page2, use 'render' to call partial:

page1.html.erb

```
<h1>Hello from page1</h1>
<p>It is <%=@time %> now</p>
<%= link_to "page2", home_page2_path %>
<%= render "footer"%>
```

page2.html.erb

```
<h1>Hello from page2</h1>
<p>It is <%=@time %> now</p>
<%= link_to "page1", home_page1_path %>
<%= render "footer"%>
```

Ruby - Ruby on Rails

passing values to partials

page1.html.erb

```
<h1>Hello from page1</h1>
<p>It is <%=@time %> now</p>
<%= link_to “page2”, home_page2_path %>
<%= render partial： “footer”, locals： {start_year : ‘2012’} %>
```

page2.html.erb

```
<h1>Hello from page2</h1>
<p>It is <%=@time %> now</p>
<%= link_to “page1”, home_page1_path %>
<%= render partial： “footer”, locals： {start_year : ‘2012’} %>
```

Ruby - Ruby on Rails

passing values to partials

_footer.html.erb

```
<hr>
<p>Copyright
<%= “#{start_year}” - if defined? start_year %>
<%= Date.today.year %> Web2
</p>
```