Introduction to Rails

Our Goals

- Understand the history of Rails
- Get Rails installed
- Get a feel for the principles of Rails
- Be able to talk about the MVC structure
- Build our first application

Still Cool... F🤬k the Haters

Sites built with Rails:

- Twitter*
- GitHub
- AirBnB
- Twitch
- Etsy, KickStarter, GoodReads, AirTasker, etc etc...

So, take heart:

- Still lots of Rails jobs
- Ideal for learning backend concepts; excellent support, most problems solved
- *But you might have to ignore your loud Node/React–obsessed developer friends for a while...*

History of Rails

- David Heinemeier Hansson ("DHH") is the creator
  - Twitter
  - Github
  - Medium
- Started as a part of BaseCamp
- Released as Open Source in July, 2004
- Commit rights given out in February, 2005

So much for "Rails", Mr Racing-Car-Guy

😘 DHH & Matz 😍

Welcome You to RailsTown

Important Links

- The Rails Doctrine
- The Ruby on Rails website
- The Ruby on Rails guides
- A study of the Rails application structure

What is it?

- It is a "web application framework"
  - Gives structure
  - Saves repetition
  - Comes with a ton of features ready to go (i.e. ActiveRecord) so you can get on with your project
  - Makes the insignificant decisions for you, so you can move faster
- It is quite opinionated about "best practice"
- The Ruby on Rails Doctrine 👇

1. Love of Ruby

"I've described this discovery of Ruby in the past as finding a magical glove that just fit my brain perfectly. Better than I had ever imagined any glove could ever fit. But it was even more than that. It was the event that marked my own personal transition from 'doing programming because I needed programs' to 'doing programming because I fell in love with it as a mode of intellectual exercise and expression'. It was finding a fountain of flow and being able to turn it on at will."

2. Seriously, A Lot of Ruby Love

"I’m not exaggerating when I say that Ruby transformed me and set the course for my life’s work. So deep was the revelation. It imbued me with a calling to do missionary work in service of Matz’s creation. To help spread this profound creation and its benefits."

3. Convetion Over Configuration

"Who cares what format your database primary keys are described by? Does it really matter whether it's 'id', 'postId', 'posts_id', or 'pid'? Is this a decision that’s worthy of recurrent deliberation? No.

Part of the Rails mission is to swing its machete at the thick, and ever growing, jungle of recurring decisions that face developers creating information systems for the web. There are thousands of such decisions that just need to be made once, and if someone else can do it for you, all the better."

4. *a.k.a. "Omakase"*

"How do you know what to order in a restaurant when you don't know what's good? Well, if you let the chef choose, you can probably assume a good meal, even before you know what "good" is. That is omakase. A way to eat well that requires you neither be an expert in the cuisine nor blessed with blind luck at picking in the dark.

There are so many conventions in Rails that a beginner doesn't even need to know about, but can just benefit from in ignorance. It's possible to create great applications without knowing why everything is the way it is."

5. Still Offer Flexibility

"While Rails is an omakase stack, it still allows you to replace certain frameworks or libraries with alternatives. It just **doesn't require you to**. Which means *you can delay those decisions until you've developed a clear, personal palette that may prefer the occasional difference*."

The Guiding Principles

Convention over Configuration (C.o.C)

If you travel by rail instead of by car, you give up some of the freedom to decide which way to go, but you can get to your destination faster.

But beware:

- If you don't strictly follow the guidelines that it sets out (naming conventions, singular vs plural etc.), it will be a lot more difficult
- Some things may feel like total magic.. but you can trust it

The Guiding Principles

Don't Repeat Yourself (D.R.Y)

- "*Every piece of knowledge must have a single, unambiguous, authoritative representation within a system*"
- Write once, use it everywhere
- Makes code more maintainable, extensible, less buggy and less surprising

The Guiding Principles

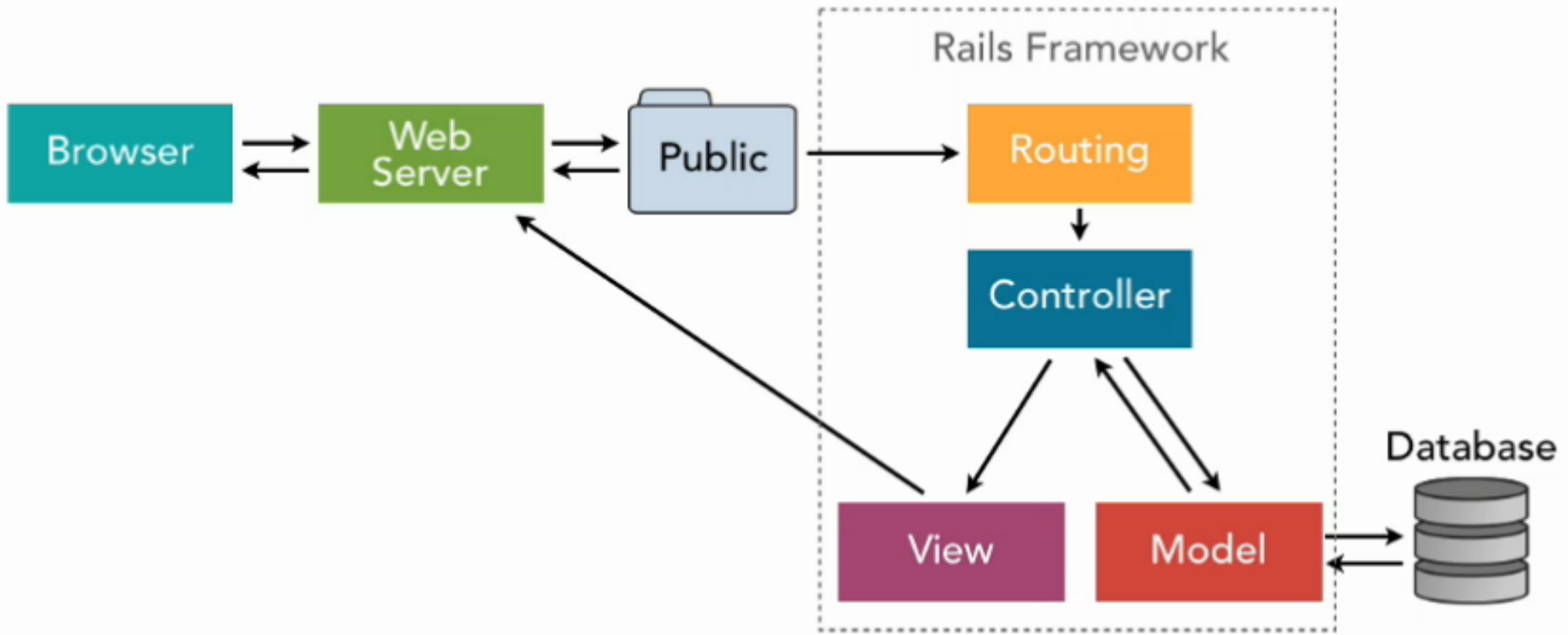
Models, View and Controllers (MVC)

(*this is probably the most important*)

- MVC is a "software architecture pattern". It is an attempt to:
  - Break code down into manageable chunks
  - Define clear boundaries between the different subsystems within an application, i.e.
  - Enforce Separation of Concerns
  - Make it easy for developers to work together

👇

Rails architecture



The Guiding Principles

Models

- The model manages the behavior and data of the application domain
- In Rails this is done with ActiveRecord classes, which are an OO representation of a table and its rows
- It is meant to be where all of the "business logic" is kept - you can add custom methods to model classes
- *i.e.*, the model directly manages the data, logic and rules of the application

The Guiding Principles

Views

- Manages the display of information
- Is given information by the controllers
- A *view* can be any output representation of information - web page, mobile app, printed page?

The Guiding Principles

Controllers

- The controller interprets the user behaviour
- Acts as the glue between the views and the models
- Makes changes to the models, palms details off to the views

The Guiding Principles

MVC in a nutshell

- The model is the **data**
- The view is the **window** on the screen
- The controller is the **glue** between the two

Let's install it

```
gem install rails
```

Let's create a Rails app

```
rails new some_app_name

cd some_app_name
```

The Rails Directory Structure

These are the ones we will be caring about today

- app
  - assets
  - controllers
  - models
  - views
- config
  - routes
- Gemfile

The approach

1. rails new app_name
2. Edit Gemfile for debugging and run bundle
3. Work out config/routes.rb
4. Create controllers
5. Add methods to controllers
6. Create appropriate views
7. Repeat as necessary

Your homework