Introduction to Rails

The History

The History of Rails

- David Heinemeier Hansson 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

Links

Some 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?

What is it?

- It is a Web Application Framework
- It provides structure
- It saves repetition
- It gives us a strong starting point

What is it?

- A very opinionated piece of software
    - It makes "assumptions" about the best way to do things
    - We call this "Convention Over Configuration"
    - Follow the Rails Way!
- Remember, the Ruby on Rails Doctrine

What is it?

- A command line tool
    - Used to make creating and working with apps easier

What is it?

- Something that takes a **long** time to get used to
    - But it will be worth it

The Guiding Principles

The Guiding Principles

Don't Repeat Yourself

Convention Over Configuration

The Guiding Principles

Models, Views and Controllers (M.V.C.)

MVC is a software architecture pattern. It attempts to:

- Break code down into manageable chunks
- Make it easy for lots of developers to work together

The Guiding Principles

Models

- Models manage behaviour and data
- It is where the database classes are
    - And the associations
- It is often a direct reflection of the table
- This is where the "business logic" should be
- The model manages the data, logic and rules of the application

The Guiding Principles

Views

- Manage the presentation of the data
    - Often an HTML page
- Is given information by the **controller**

The Guiding Principles

Controllers

- The controller interprets user behaviour
- Acts as the glue between views and models
- Makes changes to the models
- Gives information to the views

The Guiding Principles

M.V.C.

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

Installation

Installation: Linux Subsystem Users

```
wget -qO- https://deb.nodesource.com/setup_11.x | sudo -E bash -

sudo apt-get install -y nodejs

sudo apt-get install -y build-essential

gem install rails
```

Installation: Mac Users

```
brew install node

gem install rails
```

Creating a Rails Project

Creating a Rails Project

```
rails new my_new_app --skip-git

cd my_new_app
```

Warning: Linux Subsystem Users!

Add this at the end: `--skip-spring --skip-listen`

Rails Overview

The files we care about today...

```
├── Gemfile
├── Gemfile.lock
├── app
│   ├── assets
│   ├── controllers
│   ├── models
│   └── views
├── config
│   └── routes.rb
```

Routing

Let's talk about URLs

```
https://www.work.google.com:80/calendar/view?type=personal#home
```

Let's break down that URL

```
https://www.work.google.com:80/calendar/view?type=personal#home
```

Part	Description
`https`	Protocol
`www`	Information Space
`work`	Subdomain
`google.com`	Domain
`:80`	Port
`/calendar/view`	Path
`?type=personal`	Query String / Parameters
`#home`	Fragment / Anchor

We only need to worry about the path today!

We can make our own servers!

We call this the **localhost**

Localhost is a **local** server, with *many ports*

A Rails Overview (pt. 2)

- User requests a page
- The request is sent to our backend
    - The *path* is interpreted by `config/routes.rb`
- `config/routes.rb` tells Rails what:
    - **Controller** to run
    - **Action** to run (in that controller)
- The **controller** does a lot of work...
    - It also has access to **params**
- Then sends a **view** back to the browser
- The view is picked by:
    - The controller name
    - The action name

Everything lines up!

- `get "/intro" => "welcome#index"`
- Goes to the WelcomeController
    - `app/controllers/welcome_controller.rb`
- Runs the index method
    - `def index ... end`
- Loads the view
    - `app/views/welcome/index.html.erb`
- Sends that view to the browser

Routes

What is `config/routes.rb`?

Your Rails application receives a request

It asks the routes file to match it to:

- A controller
- A method

Working with Routes

```
# config/routes.rb

get "/homepage", to: "welcome#home"
```

Goes to the WelcomeController, runs the `home` method

Working with Routes

```
# config/routes.rb

get "/users/:username", to: "users#show"
```

Goes to the UsersController, runs the `show` method

But, it has access to `params[:username]`

See here for more information

Controllers

What does a controller do?

It is the C in **MVC**

- It connects the data in our database (models)
- To the views

Naming Conventions

It is very picky with the names!

For example, if in the `config/routes.rb` you had:

```
get "/users", to: "users#index"
```

Naming Conventions

- The file must be named:
    - `app/controllers/users_controller.rb`
- In that file, there must be the following code:

```ruby
class UsersController < ApplicationController
  def index
  end
  # ...
end
```

What can controllers do?

- They can work with the database
    - Using **models**
- They can interpret the request
    - Using **params**
- They can pass information to the views
    - Using **instance variables**

What can controllers do?

- You can specify what to render in the browser
- Most of the time it finds the file automatically
    - Uses the controller name to select which folder
    - Uses the action name to select which file

See here for more information

Views

What are views?

- They are the presentation of data
- Normally, they are HTML files
    - But with a little extra, **ERB**
- They get given information by the controller
    - Using **instance variables**

What is ERB?

- It stands for Embedded Ruby
- A way to make HTML files dynamic
    - By allowing you to write Ruby in them
- You can write whatever Ruby code you want
    - Loops, conditionals etc.
- The syntax takes a while to get used to!

ERB Syntax

```
Execute and print the result: <%= %>

Execute: <% %>

Commented out: <%# %>
```

How does a View get sent to the browser?

- The controller loads a specified view
- `app/layouts/application.html.erb` is loaded
    - The `<%= yield %>` statement runs
- Yield means:
    - Take the code from the specified view
    - Put it here

How does a View get sent to the browser?

- UsersController's index method runs
- `app/layouts/application.html.erb` is run
    - The ERB is executed
    - The `<%= yield %>` statement is seen
- The `app/views/users/index.html.erb` file is opened
    - The ERB is executed
- The resulting HTML is returned by the yield statement

`app/layouts/application.html.erb`?

- This is the layout for your whole site
- Very useful for things like:
  - Navigation bars
  - Footers
  - etc.

See here for more information

Homework

- Go through the Getting Started with Rails guide
- Experiment with Rails as much as possible
  - Games on Rails
  - Number Guessing Game
  - Rock, Paper, Scissors
- Learn the Rails Directory Structure