Jackie Askins

Ruby on Rails Crash Course

Where to find these slides

- These slides are available on GitHub:
  - https://github.com/jackieaskins/pennapps-rails-workshop-17f

Outline

- What You'll Need
- Ruby
- HTTP & MVC
- Ruby on Rails
- Rails Demo

What You'll Need

Text Editor & Command Line

- Text Editor:
  - Sublime Text
    - https://www.sublimetext.com/3
  - Atom
    - https://atom.io/
  - Vim/Emacs
- Command Line:
  - Required to run Ruby programs
  - Required for most Rails commands

Installing Ruby & Rails

- Install Ruby Version Manager (RVM)
  - Not required, but highly recommended
  - https://rvm.io/
- Install Ruby 2.4.1 through rvm
  - `rvm install 2.4.1`
- Install Ruby on Rails
  - Create a gemset: `rvm use 2.4.1@rails513 --create`
  - `gem install rails -v 5.1.3`

Ruby

Writing & Running Ruby

- Use a REPL (Read-Execute-Print-Loop):
  - In Command Line:
    - Run: `irb`
    - Execute lines of Ruby code
    - Exit with `quit`
  - This is great for testing
- Running Ruby programs:
  - Write code in *.rb files
  - In Command Line:
    - Change into directory containing file
    - Run: `ruby file_name.rb`

Ruby Resources

- Ruby Lang Documentation
  - https://www.ruby-lang.org/en/documentation/
- Ruby Docs
  - http://ruby-doc.org/core-2.4.1/

Printing in Ruby

- You can output data in Ruby in 3 different ways:
  - `print` outputs the value and returns `nil`
  - `puts` outputs the value with a new line and returns `nil`
  - `p` outputs and returns the value
- I will use #=> to denote output

```ruby
p 'Hello World' #=> "Hello World"
```

Variables

- Ruby is dynamically typed
- Variables do not need to be initialized
- They can be assigned & re-assigned to objects of different types
- Variable names should be in snake_case

```ruby
my_var = 'This is my var'
p my_var #=> "This is my var"
my_var = 15
p my_var #=> 15
```

Commonly Used Objects

- Numerics
  - Integers & Floats are the most common
- Strings
  - Represented with either single or double quotes
- Symbols
  - Often used as keys in hashes
- Booleans
  - true & false
- Nil
  - Represents ‘nothingness’
- Arrays
  - Holds data of any type
- Hashes
  - Mapping of keys to values

Iterators

- Always use iterators instead of for loops in Ruby

```ruby
[0, 1, 2].each do |num|
  print num
end
#=> 012

p [0, 1, 2].map { |num| num * 2 } #=> [0, 2, 4]

p({ one: 1, two: 2, three: 3 }.reject { |_, v| v.even? })
#=> { :one => 1, :three => 3}
```

Flow Control

```ruby
if 3.even?
  p 'Three is even'
elsif 3.odd?
  p 'Three is odd'
else
  p "I don't know what 3 is"
end
#=> 'Three is odd'

p 'Three is odd' if 3.odd? #=> "Three is odd"
p 'Three is not even' unless 3.even? #=> "Three is not even"
p 3.even? ? 'Three is even' : 'Three is odd' #=> "Three is odd"
```

Methods

```ruby
def hello_world
  'Hello World'
end
p hello_world #=> "Hello World"



def hello(name)
  "Hello #{name}"
end
p hello 'Jackie' #=> "Hello Jackie"



def goodnight(name = 'Moon')
  "Goodnight #{name}"
end
p goodnight 'Jackie' #=> "Goodnight Jackie"
p goodnight #=> "Goodnight Moon"
```

Classes

- Class names should be in PascalCase

```ruby
class MyClass
end
my_instance = MyClass.new
p my_instance.class #=> MyClass
```

Constructors & Instance Variables

```ruby
class Hacker
  attr_reader :name
  attr_accessor :hours_of_sleep


  def initialize(name, hours_of_sleep = 0)
    @name = name
    @hours_of_sleep = hours_of_sleep
  end
end
hacker = Hacker.new('Jackie')
p hacker.name #=> "Jackie"
p hacker.hours_of_sleep #=> 0
```

Class Instance Variables

```
class Hackathon
  @best_hackathon = 'PennApps'


  def self.best_hackathon
    @best_hackathon
  end
end
p Hackathon.best_hackathon #=> "PennApps"
```

Methods in Classes

```ruby
class MyClass
  def self.class_method
    'This is a class method'
  end


  def instance_method
    'This is an instance method'
  end
end
p MyClass.class_method #=> "This is a class method"
p MyClass.new.instance_method #=> "This is an instance method"
```

Inheritance

```
class Duck
  def quack; 'Quack'; end
end
class Mallard < Duck
end
p Mallard.superclass #=> Duck
p mallard.quack #=> "Quack"
```

Managing Dependencies

- Ruby libraries are called gems
- The command to install them is `gem install gem_name`
- Ruby programs use a Gemfile to manage the list of gems for a specific project
- You can install all gems in a Gemfile by running:
  - `gem install bundler` (Only the first time)
  - `bundle install`

HTTP & MVC

HTTP (HyperText Transfer Protocol)

- A **client** sends a **request** to a **server**
- The **server** receives the **request** and sends back a **response**
- The **response** is generally in the form of a webpage (i.e. HTML) or data (i.e. XML or JSON)

HTTP Verbs

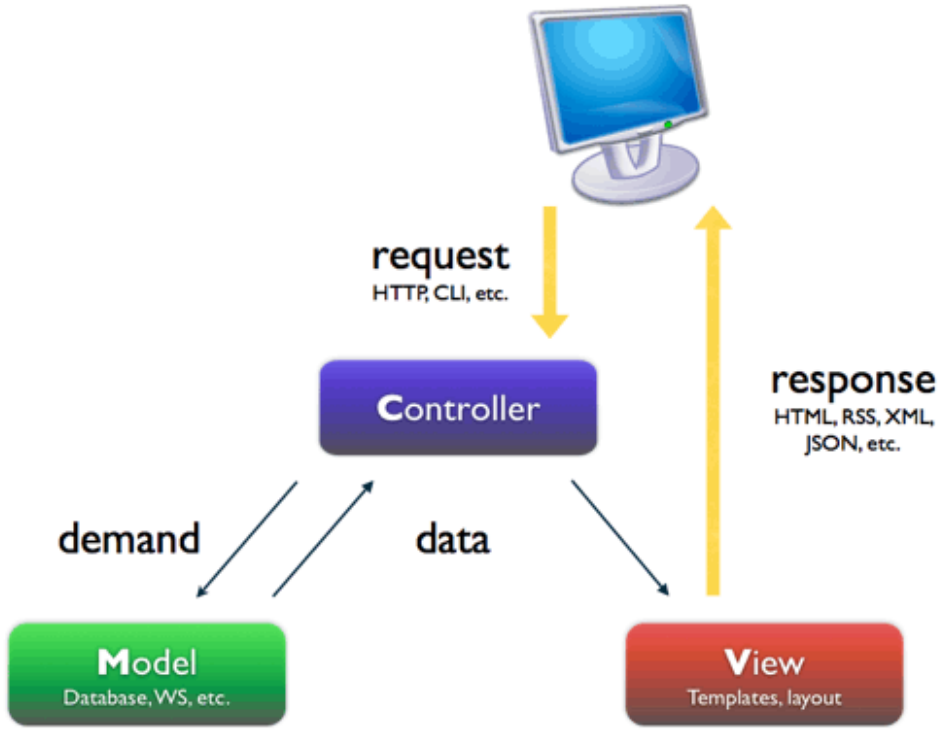
- The most common verbs are as follows:
  - GET
    - Default type of request
    - Should only be used to get data
  - POST
    - Used to send data from client to server
    - More secure than a GET request
  - PUT/PATCH
    - Used to update something on the server
  - DELETE
    - Used to delete something on the server

MVC (Model-View-Controller)

- Every community has different conventions for MVC
  - Rails is no different
- Rails is a huge proponent of Convention over Configuration

MVC Layers

- Model:
  - Contains the bulk of the application's logic
  - Main place where database is accessed
- View:
  - What the user sees
  - Should have as little logic as possible
- Controller:
  - Handles HTTP requests



REST (REpresentational State Transfer)

- A set of conventions to expose certain HTTP endpoints
- Convenient for CRUD (Create-Read-Update-Delete) apps
- The below example is for a model representing hackers:

Name	Path	Verb	Usage
Index	/hackers	GET	Show a list of hackers
New	/hackers/new	GET	Show a form to create a new hacker
Create	/hackers	POST	Create a new hacker
Show	/hackers/:id	GET	Show info on a specific hacker
Edit	/hackers/:id/edit	GET	Show a form to update an existing hacker
Update	/hackers/:id	PUT/PATCH	Update an existing hacker
Destroy	/hackers/:id	DELETE	Delete an existing hacker

Ruby on Rails

Ruby on Rails

- Ruby on Rails (RoR) is a very popular web framework built on the Ruby language
- We will be using the most recent version, 5.1.3

Creating a Rails app

- Run `rails new app_name`
- This creates a directory with the name of your app & all of the directories & files common to a Rails app
- This also installs all of the gems in the Gemfile

Rails Commands

- `rails server` (or `rails s` for short)
  - This starts the server on port 3000 by default
  - You can visit the app by going to http://localhost:3000 in a web browser
- `rails console` (`rails c`)
  - This starts an interactive console where you can access

Rails Generators

- You can use pre-defined Rails Generators that will create various files for you
  - `rails generate generator_name …`
- The ones I use the most:
  - `migration`, `model`, `controller`
- Scaffold generator:
  - Scaffolding helped to make Rails famous
  - Generates model, views, controller, tests, routes, and migrations
  - Not great for real world use

Rails App Directory Structure

- app
  - Organizes application components
  - Models, views & controllers go here
- bin
- config
  - All of the configuration for your app
  - You won't spend much time in here
  - Routes are defined here
- db
- lib
- log
- public
  - Static web files
- test
- tmp
- vendor
  - Libraries from third-party vendors

The Demo will be available on GitHub after this workshop

Rails Demo

Thanks

- Thank you everyone for coming!
- Thanks to Justin Kim who led this workshop in the past
  - Slides are based off of his workshop