rails

Jane’s code cheat sheet



Table of Contents

[Bash 3](#_Toc451883175)

[What is it? 4](#_Toc451883176)

[Error Driven Development 4](#_Toc451883177)

[Model, view, controller model 4](#_Toc451883178)

[Steps to create a rails application 5](#_Toc451883179)

[Creating a new Rails application 6](#_Toc451883180)

[Loading up the server – before can see anything in browser 6](#_Toc451883181)

[Creating Something to see in the browser 6](#_Toc451883182)

[Create a route for the browser in the Routes folder – in config 7](#_Toc451883183)

[Create the controller – File for pages controller – inheriting from applicationController provided by railes 7](#_Toc451883184)

[Create pages views 8](#_Toc451883185)

[Viewing things in the browser 8](#_Toc451883186)

[Creating an app using rails and more SIMPLE commands 8](#_Toc451883187)

[Create the Rails app 8](#_Toc451883188)

[the Database Stuff 9](#_Toc451883189)

[Creating the database 9](#_Toc451883190)

[Views 10](#_Toc451883191)

[Seeing and managing the Content 10](#_Toc451883192)

[Create 11](#_Toc451883193)

[Delete 11](#_Toc451883194)

[Associations 11](#_Toc451883195)

[Start showing associations into the views 12](#_Toc451883196)

[Common commands 12](#_Toc451883197)

[IN the console 12](#_Toc451883198)

# Bash

|  |  |
| --- | --- |
|  | *Command* |
| *Getting into bash* | *subl ~/.bash\_profile* |
|  |  |
|  |  |

RAILS

#### Rails

## **What is it?**

* web application framework, built with Ruby.
* Rails is very opinionated (the same as its creator), it makes an assumption there is a 'best' way to do it, and it encourages that way (but still allows for flexibility). If you follow the 'Rails Way', it will be much easier.

The Rails philosophy includes three major guiding principles, which really just stem from Ruby itself (except the MVC pattern):

* Don't Repeat Yourself (D.R.Y) - DRY is a principle of software development, not exclusive to Ruby (though Ruby is very good at it, as is its community). It's main thing is that *"Every piece of knowledge must have a single, unambiguous, authoritative representation within a system"*. That makes it sound a lot more complex than it really is. All it means is that by not writing the same code over and over again, our code is more maintable, extensible and less buggy.
* Convention Over Configuration (C.O.C) - Rails, as previously mentioned, is very opinionated. If you don't follow its guidelines (naming conventions etc.), it is difficult. But if you do, the set up is very efficient. It has the capacity to do most things by default as long as you follow the right approach.
* Model, View and Controller (M.V.C) – This is probably the most important. Rails structures everything like this, so it is important to understand this. MVC is a software architecture pattern that breaks the code into small manageable chunks and stems from the problem when trying to modularize a user interface functionality so that it is maintanable and extensible.

## Error Driven Development

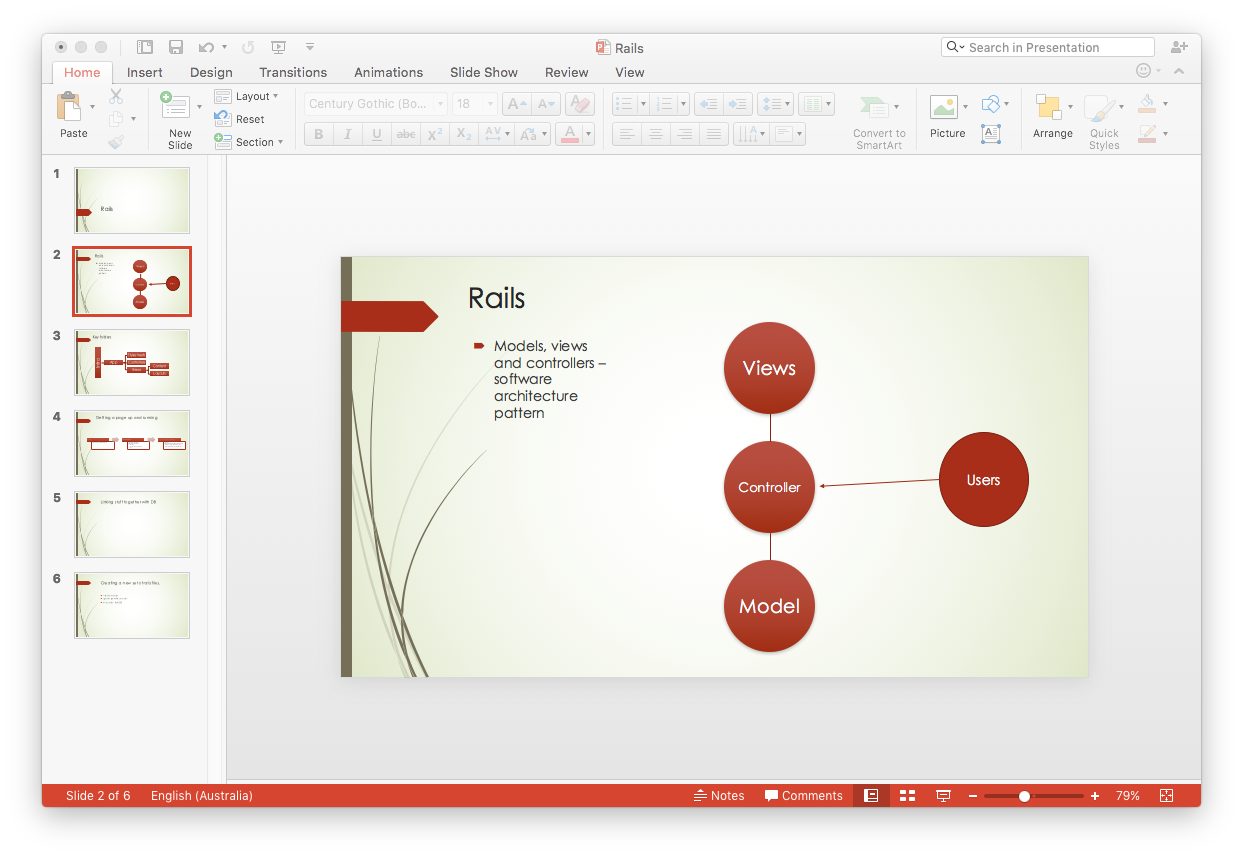
## Model, view, controller model

* Those chunks are Models, Views and Controllers as you have probably guessed. But what are they?

1. Models - The model manages the behavior and data of the application domain, also where the database classes are created.
2. Views - The view manages the display of information.
3. Controllers - The controller interprets the user behaviour ( it is like the switchboard )

An easy way to understand MVC: the model is the data, the view is the window on the screen, and the controller is the glue between the two. -- [ConnellyBarnes](http://www.connellybarnes.com/work/)

But is all about the interactions. A controller can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to the associated view to change the presentation (and content). A model stores data that is retrieved by the controller and displayed in the view. Whenever there is a change to the data it is updated by the controller. A view requests information from the model that it uses to generate an output representation to the user.



## Steps to create a rails application

1. Planning
2. rails new project\_name
3. Edit Gemfile for debugging
4. Work out your routes - config/routes.rb
5. Create controllers
6. Create methods in controllers
7. Create appropriate views
8. Repeat as necessary

## Creating a new Rails application

1. First step is always planning, before you get into a Rails project –
2. In the console: To create the Rails project

 rails new project\_name

1. Go to project folder

( cd project\_name ).

1. In Rails folders – go to Gemfile to be more suited for debugging. It’s in

Vendor/gemfile

The code we tend to use is this...

group :development do

rail

end

1. In console run

bundle

That will sort out all our gems and get all the dependencies loaded which is quite important.

## Loading up the server – before can see anything in browser

In the console - lets load up the server - rails server or rails s.

Getting out of the server???

## Creating Something to see in the browser

### Create a route for the browser in the Routes folder – in config

Now we get into the Routes (found in config/routes.rb).

This is like our phone directory or the 'gets' in Sinatra.

Approach like this

Rails.application.routes.draw do

# controller#method

root :to => 'pages#home'

# route controller#home

get '/home' => 'pages#home'

# DYNAMIC ROUTES WITH VARIABLE BITS IN PARAMS (JUST LIKE SINATRA)

get '/auto/:color' => 'auto#color'

end

### Create the controller – File for pages controller – inheriting from applicationController provided by railes

Error = uninitialized Constant.

This means that we haven't created the associated controller. In our routes, for our root page we have said that it wants "pages#home". This says we need a Pages Controller and a home method within it. So within the app/controller folder, we need to create a file called pages\_controller.rb. All our controllers inherit from ApplicationController - this is what gives it all of its functionality. Within the PagesController, at its most basic, it is going to look something like this...

class PagesController < ApplicationController

def home

end

end

### Create pages views

Once we have this setup, we can need to create our views (the error is missing template). We need to create a pages folder in our app/views folder, and within that we need a home.html.erb. Once we have that, it will be required by default for us through the method (that's why naming conventions are so important with Rails).

## Viewing things in the browser

If we go to localhost:3000, you can see an error. We are up to an Error Driven Development point now, so if we load the root page (which is pages#home), we let the errors guide where we go next.

# Creating an app using rails and more SIMPLE commands

* rails new app-name
* cd app-name
* Add gem 'pry-rails' into your Gemfile (in the development group)
* run bundle
* run rake db:create
* Generate your first migrations and models - either rails g model Model name:string type:string etc. or:
  + rails g migration create\_tables
  + Add the fields you need to that file - make sure you include t.timestamps
  + rails g model ModelName
* Generate your controllers and your views - either rails g controller Users index new create delete show etc. or:
  + rails g controller Users
  + Add your methods into the controller
  + Create views that correspond with the method names in the view folder for that particular controller
* Work out your routes file
* Repeat this stuff as necessary

## Create the Rails app

1. In the console, great a new application

Rails new mona\_app

1. CD into mona app
2. Add gems into the gemfile. Typically add gem ‘pry-rails’
3. From the console, run bundle or bundle install
4. Start the server in a new window pan (CMD +D) From there run

rails s or rails server

## the Database Stuff

### Creating the database

1. Map out on a piece of paper using the crows feet. the database and the relationships between the tables ie the one to many things and add the relevant foreign key to the relevant page. Put the foreign key in the one where it’s only one, but don’t worry about the association to start.

One person has many awesomenesses

PERSON

Person\_id Number

Name INTEGER

Location INTEGER

Nationality

Gender

AWE

Awesome\_activity TEXT

Awesome\_date DATE

Awesome\_score Number

Awesome\_image TEXT

create\_table "artists", force: :cascade do |t|

t.text "name"

t.text "nationality"

t.date "dob"

t.text "period"

t.text "image"

t.datetime "created\_at"

t.datetime "updated\_at"

end

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| PERSON |  |  | Awesome\_id |
| Person\_id Number |  |  | Awesome\_activity TEXT |
| Name INTEGER | t.text "name" |  | Awesome\_date DATE |
| Location INTEGER | t.text “location” |  | Awesome\_score Number |
| Nationality |  |  | Awesome\_image TEXT |
| Gender |  |  | Person\_id |

Don’t forget to include the timestamps.

Resources:artists

1. In the config/routes.rb file add

Then visit localhost:300/rails/info/routes

1. If you want the DB to start with things, make the models

Rails generate model artist

Add code into seed file to make sure that add “Artist.destroy\_all”

Run ????

1. Start with the READ steps to create the pages, looping through the following
   1. Controllers

def index; @artists = Artist.all; end

from the console run

rails generate model Artist –skip

Fill in the migration file with everything that is necessary

Rake db:migrate

Lood at db/schema.rb to make sure it worked

### Views

1. Mkdir app/views/artists
2. Touch app/views/index.html.erb
3. Within the new test, write whatever is necessary – typically an each loop

#### Start with read steps See all

Some pages will include links, using the ruby helpers.

<% = link\_to (“Edit Artist”, “/artist/:id/edit”) % >

for delete, make sure you add the “are you sure thing” before they delete it in the link.

## Seeing and managing the Content

|  |  |  |  |
| --- | --- | --- | --- |
| User activity | What’s in the controller | What file | What file content |
| READ ie see the data |  |  |  |
| EDIT |  | Use a partial file |  |
| CREATE |  |  |  |
| DELETE |  |  |  |

#### EDIT

Creating the form

Create the partial – using \_file

Call the partial using render

## Create

### Delete

## Associations

Two different forms

has\_many

belongs\_to

Add associations to the models using the classed.

The one that is the one ie belongs to the other, is the one that has the forieng key.

To add an extra column

Rails generate migration add\_artist\_id\_to\_works

Fill in migration

Rake db:migrate

Test associations in the console

Rails c or rails consoler

W = Work.first

w.artist\_id = Artist.first.id

Artist.first.works

Works.first.artist

### Start showing associations into the views

## Common commands

### IN the console

rails new some\_app\_name

rake db:create

rake db:migrate

rake db:seed

rails generate migration create\_artists

rails generate model Artist

rake db:rollback – reverts to the last one

rails server or rails s

rails server –p3001 (if want multiple servers running)

rails console or rails c

rails db

* + [BONUS NOTES!](https://gist.github.com/ga-wolf/5cb46d31e8dfa8c7114b7b9a28bbc6ea)

# Rails Helpers

These are things that you should use over and over, it is a collection of commonly used patterns. They can only be used in our views!!! Remember that these obviously need to be wrapped in ERB tags.

## NUMBER HELPERS

number\_to\_currency( value )

number\_to\_human( value )

number\_to\_phone( value, options )

number\_to\_phone( value, :area\_code => true )

[Here are all of the number helpers.](http://api.rubyonrails.org/classes/ActionView/Helpers/NumberHelper.html)

## TEXT Helpers

pluralize( value, 'singular\_case' )

pluralize( @person\_count, 'person' )

truncate( value, options )

truncate( @story, :length => 50 )

cycle( list\_of\_values )

cycle( 'red', 'green', 'orange', 'purple' )

[Here are all of the text helpers.](http://api.rubyonrails.org/classes/ActionView/Helpers/TextHelper.html)

## Assets Helpers

image\_tag( 'path', options )

image\_tag( 'funny.jpg' )

image\_tag( 'http://fillmurray.com/500/500', :class => "oh-bill" )

[Here are all of the asset helpers.](http://api.rubyonrails.org/classes/ActionView/Helpers/AssetUrlHelper.html)

## URL Helpers

link\_to( 'Home', root\_path )

link\_to( 'Work Path Show', work\_path( work.id ) )

link\_to( 'Work Path Show', work\_path( work ) )

link\_to( 'Work Path Show', work )

button\_to( 'Test Path', root\_path, :method => 'GET' )

[Here are all of the url helpers.](http://api.rubyonrails.org/classes/ActionView/Helpers/UrlHelper.html)

# The Power of Rails in Terminal

There are a few commands that are absolutely essential for Rails development. The more you know them, the better it is!

At its most basic:

* rails new - Generate a new application
* rails server - Runs the server
* rails generate - Generate a whole heap of things within an application
* rails console - Opens up a console
* rails dbconsole - Opens up a direct connection to SQL
* rake - Does thousands of things
* bundle - Install gems and their dependencies

Running any command with -h or --help at the end will show you the documentation for that particular command. But the real power comes from...

## Customization and Automation!

### RAILS NEW

rails new app\_name

rails new app\_name -T # Skips the Test Suite

rails new app\_name --database=postgresql # Specifies the Database (changes it from sqlite3)

rails new app\_name -d postgresql

### RAILS SERVER

rails server

rails s # Shorthand for rails server

rails server -p 3001 # Specifies another port (have multiple servers at once!)

rails server -e production # Changes the state of the application (different gem sets etc. - don't worry about this one)

*RAILS GENERATE*

rails generate controller ControllerName list of actions

rails generate controller Greetings index create

rails g controller Greetings index create

# THIS WILL CREATE VIEWS, JS, CSS AND ACTIONS IN CONTROLLERS (PLUS TESTS)

rails g model ModelName field:type

rails g model Painting name:string year:date

# THIS WILL CREATE MODELS, MIGRATIONS, AND TESTS

rails g scaffold ModelName field:type field:type

rails generate scaffold Painting name:string year:date

# THIS WILL CREATE EVERYTHING

*RAILS CONSOLE*

rails console # OPENS UP YOUR RAILS APP IN PRY OR IRB

rails c # SHORTHAND

rails console staging # OPENS UP A SPECIFIC ENVIRONMENT

rails console --sandbox # CAN'T MAKE ANY ACTUAL CHANGES

*RAILS DBCONSOLE*

rails dbconsole # OPENS UP A DIRECT CONNECTION TO YOUR DATABAS

rails db # SHORTHAND

*BUNDLE*

bundle install # INSTALLS GEM AND DEPENDENCIES

bundle # SHORTHAND

## RAKE

This is crazy powerful and does a million things, but here are some of the more important ones that you might need to know. We will go into this in a lot more detail though!

rake --tasks # Lists everything it can do

rake about # Lists everything about your Rails app

rake db:drop # DROPS THE DATABASE

rake db:create # CREATES THE DATABASE

rake db:migrate # MIGRATES TABLES INTO THE DATABASE (FROM db/migrations)

rake db:rollback # GOES BACK ONE STEP IN THE DATABASE (BACK ONE MIGRATION)

rake routes # LIST ALL OF YOUR ROUTES

rake stats # LINES OF CODE ETC.

rake notes # SEE HERE - http://guides.rubyonrails.org/command\_line.html#notes

# Form Helpers

At its most basic...

<%= form\_tag("/search", method: "get") do %>

<%= label\_tag(:q, "Search for:") %>

<%= text\_field\_tag(:q) %>

<%= submit\_tag("Search") %>

<% end %>

Binding a form to an object...

# OUR CONTROLLER

def new

@article = Article.new

end

<!-- OUR ASSOCIATED VIEW -->

<%= form\_for @article, url: {action: "create"}, html: {class: "nifty\_form"} do |f| %>

<%= f.text\_field :title %>

<%= f.text\_area :body, size: "60x12" %>

<%= f.submit "Create" %>

<% end %>

## TAG HELPERS!

<%= radio\_button\_tag(:age, "adult") %>

<%= label\_tag(:age\_adult, "I'm over 21") %>

<%= text\_area\_tag(:message, "Hi, nice site", size: "24x6") %>

<%= password\_field\_tag(:password) %>

<%= hidden\_field\_tag(:parent\_id, "5") %>

<%= search\_field(:user, :name) %>

<%= telephone\_field(:user, :phone) %>

<%= date\_field(:user, :born\_on) %>

<%= datetime\_field(:user, :meeting\_time) %>

<%= datetime\_local\_field(:user, :graduation\_day) %>

<%= month\_field(:user, :birthday\_month) %>

<%= week\_field(:user, :birthday\_week) %>

<%= url\_field(:user, :homepage) %>

<%= email\_field(:user, :address) %>

<%= color\_field(:user, :favorite\_color) %>

<%= time\_field(:task, :started\_at) %>

<%= number\_field(:product, :price, in: 1.0..20.0, step: 0.5) %>

<%= range\_field(:product, :discount, in: 1..100) %>

<%= select\_tag(:city\_id, '<option value="1">Lisbon</option>...') %>

<% cities\_array = City.all.map { |city| [city.name, city.id] } %>

<%= options\_for\_select(cities\_array) %>

<%= time\_zone\_select(:person, :time\_zone) %>

<%= select\_date Date.today, prefix: :start\_date %>

Thousands of things you can do, go through [here!](http://guides.rubyonrails.org/form_helpers.html)

# A Basic Rails Guide

Treat this as a really rough guide, definitely don't always follow it. You'll figure out your approach soon.