# Creating an app using rails and more SIMPLE commands

Table of Contents

[Creating an app using rails and more SIMPLE commands 1](#_Toc452415473)

[1 Create the Rails app 2](#_Toc452415474)

[2 Start the server 3](#_Toc452415475)

[3 the Database Stuff 3](#_Toc452415476)

[Creating the database 3](#_Toc452415477)

[CRUD – create, Read, Update, destroy 5](#_Toc452415478)

[read 5](#_Toc452415479)

[4 CREATE the CRUD 5](#_Toc452415480)

[Views 5](#_Toc452415481)

[Seeing and managing the Content 5](#_Toc452415482)

[Associations 6](#_Toc452415483)

[Start showing associations into the views 6](#_Toc452415484)

[Common commands 6](#_Toc452415485)

[IN the console 6](#_Toc452415486)

[Rails Helpers 7](#_Toc452415487)

[NUMBER HELPERS 7](#_Toc452415488)

[TEXT Helpers 7](#_Toc452415489)

[Assets Helpers 8](#_Toc452415490)

[URL Helpers 8](#_Toc452415491)

[The Power of Rails in Terminal 8](#_Toc452415492)

[Customization and Automation! 9](#_Toc452415493)

[RAILS NEW 9](#_Toc452415494)

[RAILS SERVER 9](#_Toc452415495)

[RAKE 10](#_Toc452415496)

[Form Helpers 11](#_Toc452415497)

[TAG HELPERS! 11](#_Toc452415498)

[A Basic Rails Guide 12](#_Toc452415499)

[Creating complex database 12](#_Toc452415500)

[Creating associations 13](#_Toc452415501)

[Week 05, Day 03 13](#_Toc452415502)

[Rails folder structure 18](#_Toc452415503)

1. rails new app-name
2. cd app-name
3. Add gem 'pry-rails' into your Gemfile (in the development group)
4. run bundle
5. run rake db:create
6. 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
7. 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
8. Work out your routes file
9. Repeat this stuff as necessary

# 1 Create the Rails app

In the console, great a new application

Rails new mona\_app – if not using postgres

If using postgres

Rails new YOURAPPNAMEHERE –d postgresql

Cd into the app

 rake db:create

1. CD into app
2. Add gems into the gemfile. Typically add gem
   1. Gem ‘pry-rails’
   2. Gem ‘annotate
3. From the console, run bundle or bundle install
4. Cd into the app

# 2 Start the server

1. Start the server in a new window pan (CMD +D) From there run

Go to the browser and see if the server is working

rails s or rails server

To stop the server

hit Ctrl+C in the terminal window where it's running.

## 3 the Database Stuff

### 

1. run rake db:create
2. Generate first migrations and models

Generate your first migrations and models - either   
rails g model Model name:string type:string etc

rails g migration create\_tables

* + Add the fields you need to that file - make sure you include t.timestamps
  + rails g model ModelName

### 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.

Don’t forget to include the timestamps. [Check how to include 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 ????

r

# CRUD – create, Read, Update, destroy

## read

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

## 4 CREATE the CRUD

### 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 |  |  |  |

## 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

Code block style 2 in a table

### 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.

# Creating complex database

Create DBs etc before views.

Test in the rails console

Use the following types of puts line to show what’s in the console

"User count: #{User.all.count}" to see what’s going on

# Creating associations

To put the data in, can do it through seed.db using seed.db file.

Need also to create the relationships in the model of the data in the .rb file.

Put it into the definition of the class within the relevant .rb fle

Make sure the right table has the foreign key into it.

Run rake db:seed to incorporate the data

Belongs to and has many

|  |  |
| --- | --- |
| Relationship – belongs to and has many | Commands to go into the .rb file |
| An album has many songs | has\_many |
| A song belongs to one album. | belongs\_to |

## Week 05, Day 03

What we covered today:

* [Warmup!](https://gist.github.com/cjbarnaby/4eb73d6a8b3b5fff1106)
  + [Solution](https://github.com/ga-wolf/WDi14-Homework/tree/master/WarmupExercises/week%205)
* Helpers
* TUNR
  + Has and Belongs to Many
  + Has Many Through

#### 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)

#### Associations!

We need associations to make common tasks easier and more readable. In Rails, an association is a connection between two models that inherit from ActiveRecord::Base ( called Active Record models for understandable reasons ). There are six types of associations and we will briefly look at them now.

* belongs\_to
* has\_one
* has\_many
* has\_many :through
* has\_one :through
* has\_and\_belongs\_to\_many

For a far deeper dive though, [see here.](http://guides.rubyonrails.org/association_basics.html)

Belongs To

A belongs\_to association sets up a one-to-one connection with another model, such that each instance of the declaring model "belongs to" one instance of the other model. This is often used in conjunction with has\_one or has\_many. Remember that belongs\_to associations must be in the singular term!

class Order < ActiveRecord::Base

belongs\_to :customer

end

Has One

A has\_one association also sets up a one-to-one connection with another model, but with somewhat different semantics (and consequences). This association indicates that each instance of a model contains or possesses one instance of another model.

class User < ActiveRecord::Base

has\_one :account

end

Has Many

A has\_many association indicates a one-to-many connection with another model. You'll often find this association on the "other side" of a belongs\_to association. This association indicates that each instance of the model has zero or more instances of another model. Note that the name of the other model is pluralized when declaring a has\_many association.

class User < ActiveRecord::Base

has\_many :orders

end

Has Many Through

A has\_many :through association is often used to set up a many-to-many connection with another model. This association indicates that the declaring model can be matched with zero or more instances of another model by proceeding through a third model.

class Doctor < ActiveRecord::Base

has\_many :appointments

has\_many :patients, through: :appointments

end

class Appointment < ActiveRecord::Base

belongs\_to :doctor

belongs\_to :patient

end

class Patient < ActiveRecord::Base

has\_many :appointments

has\_many :doctors, through: :appointments

end

Has One Through

A has\_one :through association sets up a one-to-one connection with another model. This association indicates that the declaring model can be matched with one instance of another model by proceeding through a third model.

class Supplier < ActiveRecord::Base

has\_one :account

has\_one :account\_history, through: :account

end

class Account < ActiveRecord::Base

belongs\_to :supplier

has\_one :account\_history

end

class AccountHistory < ActiveRecord::Base

belongs\_to :account

end

Has And Belongs To Many

A has\_and\_belongs\_to\_many association creates a direct many-to-many connection with another model, with no intervening model.

class Assembly < ActiveRecord::Base

has\_and\_belongs\_to\_many :parts

end

class Part < ActiveRecord::Base

has\_and\_belongs\_to\_many :assemblies

end

#### Homework!

[This](https://gist.github.com/ga-wolf/f41e394126a7980cb1bac9f202ce7ac2)

Join tabes

# Rails folder structure

| **File/Folder** | **Purpose** |
| --- | --- |
| app/ | Contains the controllers, models, views, helpers, mailers, channels, jobs and assets for your application. You'll focus on this folder for the remainder of this guide. |
| bin/ | Contains the rails script that starts your app and can contain other scripts you use to setup, update, deploy or run your application. |
| config/ | Configure your application's routes, database, and more. This is covered in more detail in [Configuring Rails Applications](http://edgeguides.rubyonrails.org/configuring.html). |
| config.ru | Rack configuration for Rack based servers used to start the application. |
| db/ | Contains your current database schema, as well as the database migrations. |
| Gemfile Gemfile.lock | These files allow you to specify what gem dependencies are needed for your Rails application. These files are used by the *Bundler gem*. For more information about Bundler, see the [Bundler website](http://bundler.io/). |
| lib/ | Extended modules for your application. |
| log/ | Application log files. |
| public/ | The only folder seen by the world as-is. Contains static files and compiled assets. |
| Rakefile | *This file locates and loads tasks that can be run from the command line.* The task definitions are defined throughout the components of Rails. Rather than changing Rakefile, you should add your own tasks by adding files to the lib/tasks directory of your application. |
| README.md | This is a brief instruction manual for your application. You should edit this file to tell others what your application does, how to set it up, and so on. |
| test/ | Unit tests, fixtures, and other test apparatus. These are covered in [Testing Rails Applications](http://edgeguides.rubyonrails.org/testing.html). |
| tmp/ | Temporary files (like cache and pid files). |
| vendor/ | A place for all third-party code. In a typical Rails application this includes vendored gems. |