#Ruby on Rails

Rails is used for database backed web apps

4 parts to a rails application

- Models

-- CRUD #Create Update Destroy

-- Validation #

-- Associations

- Routes

--"For this URL, do this thing"

- Controllers – receives a web request

-- 1) Tell the models to CRUD some data

-- 2) Tell Rails what to show the user

- Views

-- Displayed to the user

**Beginning FIRST STEP**

Rails new appname --skip-test-unit

Open Gemfile #Delete those lines, maybe

**Git**

git init

git add .

git commit –m ‘bullshit message’

git push origin master

where origin is the repository (http://stackoverflow.com/questions/922210/unable-to-git-push-master-to-github)

**Database**

1) Bundle exec rails g model Class #class should be singular

2) Edit the generated file in the /DBcd ch

3) Bundle exec rake db:migrate

Find the model in models, validate terms in model

**RAILS CONSOLE**

Bundle exec rails c #exit with exit

paramspar

Class.create!(params) #params means

Class.save!

Reload!

Rm db/development/sqlite #removes DB

**Server business**

Bundle exec rails s **#Boots the rails server**

IN YOUR WEB BROWSER: Localhost:3000

Command + T for a new tab

In the event it’s still running ps ax | grep ruby

Kill <number of process>

**Controllers**

Rails g controller filename

*Bundle exec rails g scaffold\_controller <name> name:string #name is singular*

**Routes (routes refer to controller actions)**

Config/Routes

* OLD: match 'show\_first\_candy' => 'candies#show' *This is the candies controller show action*
* New Restful routes: resources :controllers #where controllers is controller name plural
* rake routes explains

Bundle exec rake routes #just lists the routes

Routes: localhost:3000/routename

Destroy command

**Index**

Apps/views/controllername/index.html.erb

**Helpers #Relate to views**

Apps/view/helpers

Alter the file: these are defined methods for formatting information for views

**Partials #sub view, signified with an underscore before**

These are in app/views?

The partial is an erb with an underscore in front of it.

Then in the view index you call the partial without the underscore

Then in original <%= render :partial => ‘name\_with\_link’, :locals => { :state => state }

Bundle exec rails runner script/filename.rb

**Callbacks**

before\_validation :favorite

**Devise: Authenticates Users (https://github.com/plataformatec/devise)**

Add: gem ‘devise’

Bundle install

Bundle exec rails generate devise:install

-----

Some setup you must do manually if you haven't yet:

1. Ensure you have defined default url options in your environments files. Here is an example of default\_url\_options appropriate for a development environment in config/environments/development.rb:

config.action\_mailer.default\_url\_options = { :host => 'localhost:3000' }

In production, :host should be set to the actual host of your application.

2. Ensure you have defined root\_url to \*something\* in your config/routes.rb.

For example:

root :to => "home#index #Where home is application(controller?) foldername

3. Ensure you have flash messages in app/views/layouts/application.html.erb.

For example:

<p class="notice"><%= notice %></p>

<p class="alert"><%= alert %></p>

4. If you are deploying Rails 3.1 on Heroku, you may want to set:

config.assets.initialize\_on\_precompile = false

On config/application.rb forcing your application to not access the DB or load models when precompiling your assets

THEN DO:

rails generate devise User

rake routes

Now you have a user model

bundle exec rake db:migrate

IN APPLICATION CONTROLLER: before\_filter :authenticate\_user!

Delete public/index.html?

1. rails generate migration add\_username\_to\_users username:string
2. Run the migration:

rake db:migrate

1. Modify the User model and add username, email, password, password confirmation and remember me to attr\_accessible

attr\_accessible :username, :email, :password, :password\_confirmation, :remember\_me

**Create a login virtual attribute in Users**

1. Add login as an attr\_accessor

*# Virtual attribute for authenticating by either username or email* *# This is in addition to a real persisted field like 'username'* **attr\_accessor** :login

1. Add login to attr\_accessible

attr\_accessible :login

**Tell Devise to use :login in the authentication\_keys**

1. Modify config/initializers/devise.rb to have:

config**.**authentication\_keys **=** **[** :login **]**

* **If you are using multiple models with Devise, it is best to set the authentication\_keys on the model itself if the keys may differ:**

devise :database\_authenticatable, :registerable, :recoverable, :rememberable, :trackable, :validatable, :authentication\_keys **=>** **[**:login**]**

1. Overwrite Devise’s find\_for\_database\_authentication method in Users model

* **For ActiveRecord:**

**def** **self.find\_first\_by\_auth\_conditions**(warden\_conditions) conditions **=** warden\_conditions**.**dup **if** login **=** conditions**.**delete(:login) where(conditions)**.**where(**[**"lower(username) = :value OR lower(email) = :value", { :value **=>** login**.**downcase }**]**)**.**first **else** where(conditions)**.**first **end** **end** *### This is the correct method you override with the code above* *### def self.find\_for\_database\_authentication(warden\_conditions)* *### end*

* **For Mongoid:**  
  Note: This code for Mongoid does some small things differently than the ActiveRecord code above. Would be great if someone could port the complete functionality of the ActiveRecord code over to Mongoid [basically you need to port the ‘where(conditions)’]. It is not required but will allow greater flexibility.

field :email **def** **self.find\_first\_by\_auth\_conditions**(warden\_conditions) conditions **=** warden\_conditions**.**dup **if** login **=** conditions**.**delete(:login) self**.**any\_of({ :username **=>** /^#{Regexp**.**escape(login)}$/i }, { :email **=>** /^#{Regexp**.**escape(login)}$/i })**.**first **else** **super** **end** **end**

**Update your views**

1. Make sure you have the Devise views in your project so that you can customize them  
   **Rails 3:**

rails g devise:views

**Rails 2:**

script**/**generate devise\_views

1. Modify the views
   * sessions/new.html.erb:

- <p><%= f.label :email %><br /> - <%= f.email\_field :email %></p> + <p><%= f.label :login %><br /> + <%= f.text\_field :login %></p>

* + registrations/new.html.erb

+ <p><%= f.label :username %><br /> + <%= f.text\_field :username %></p> <p><%= f.label :email %><br /> <%= f.email\_field :email %></p>

* + registrations/edit.html.erb

+ <p><%= f.label :username %><br /> + <%= f.text\_field :username %></p> <p><%= f.label :email %><br /> <%= f.email\_field :email %></p>

**Manipulate the :login label that Rails will display**

**Rails 3:**

en: activerecord: attributes: user: login: "Username or email"

**Gem Paperclip**

**Heroku**

New Procfile (no suffix) in the root directory of the application we put on the internet. It says:

web: bundle exec thin start –p $PORT

Then in the gemfile

Group :development, :test do

Gem ‘sqlite3’

end

Group :production do

Gem ‘pg’

end

then: bundle install --without production

gem ‘thin’

heroku login

heroku keys:add

heroku create --stack cedar

git push heroku master

**mass assign in console**

300.times { Object.create! :name =>%w( possibility other blah ).sample, :attribute => [52, 38, 87].sample }

300.times { Candy.create! :name=> %w( starburst gum ).sample, :color => [ 'red' 'blue' 'nil'].sample, :rating => [ 1 2 3 ].sample }

**Pagination**

Gem ‘kaminari’

Bundle install

Rails g kainari:config

App/views/index

<%= paginate @model %>

* 1. Git commit
  2. Delete Database
  3. Delete the migration files for Devise
  4. Copy old Voter model (not the devise parts)
  5. Rails generate destroy devise Voter
  6. Copy Voter Controller (old) \
  7. Make a devise voter model
  8. Copy in old voter model