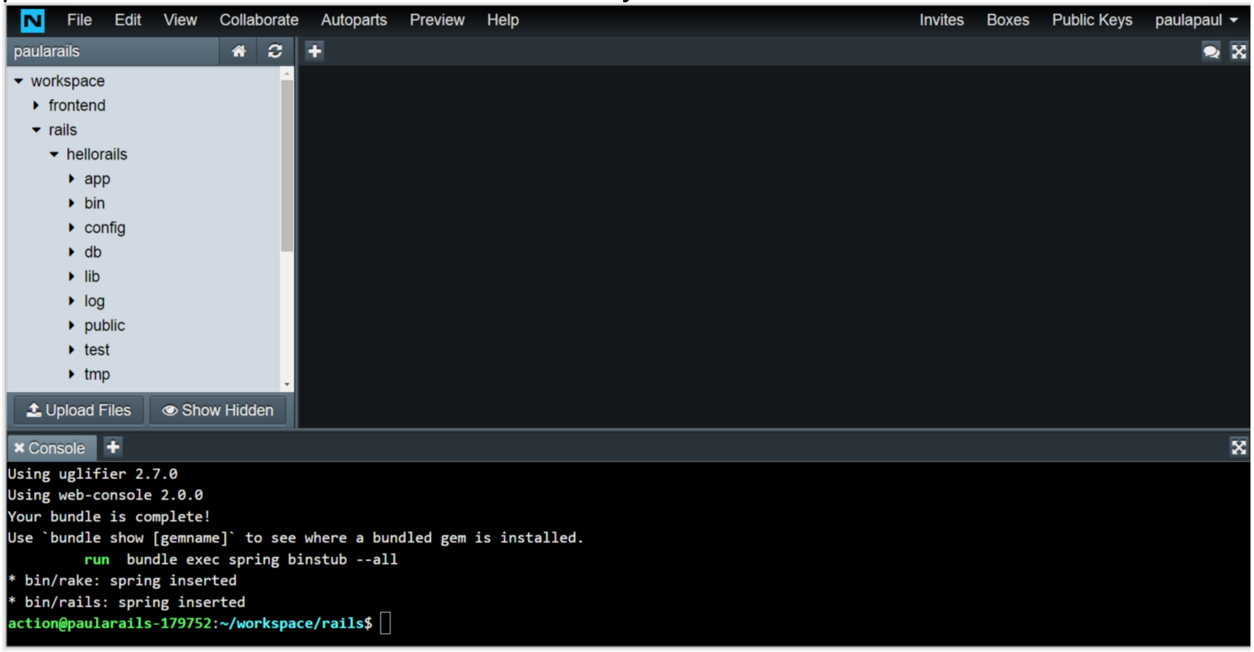
rails new hello_world

Rails has created a 'hello_world' directory under your rails working directory, and has placed all sorts of files and folders there for you:



```
Using uglifier 2.7.0
Using web-console 2.0.0
Your bundle is complete!
Use `bundle show [gemname]` to see where a bundled gem is installed.
         run  bundle exec spring binstub --all
* bin/rake: spring inserted
* bin/rails: spring inserted
action@paularails-179752:~/workspace/rails$ 
```

Change to the hello_world directory and enter a command to 'scaffold' a web page:

cd hello_world

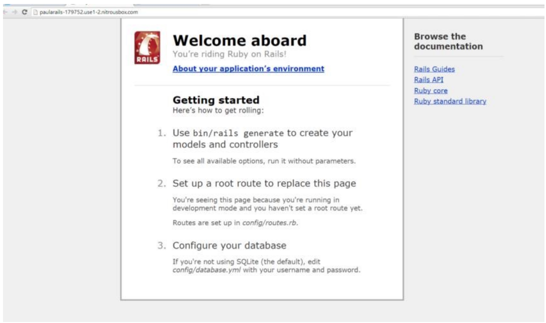
rails generate controller Pages home

We're done! You've just created a Rails application with a Home page that can be styled with HTML/CSS.

To see what you have so far, start your rails server by entering:

rails s **…**or… **rails server**

At http://localhost:3000, within your browser, you should now see a Rails Application!



Welcome aboard

You're riding Ruby on Rails!

About your application's environment

Getting started

Here's how to get rolling:

1. Use `bin/rails generate` to create your models and controllers

   To see all available options, run it without parameters.

2. Set up a root route to replace this page

   You're seeing this page because you're running in development mode and you haven't set a root route yet.

   Routes are set up in *config/routes.rb*.

3. Configure your database

   If you're not using SQLite (the default), edit *config/database.yml* with your username and password.

Browse the documentation

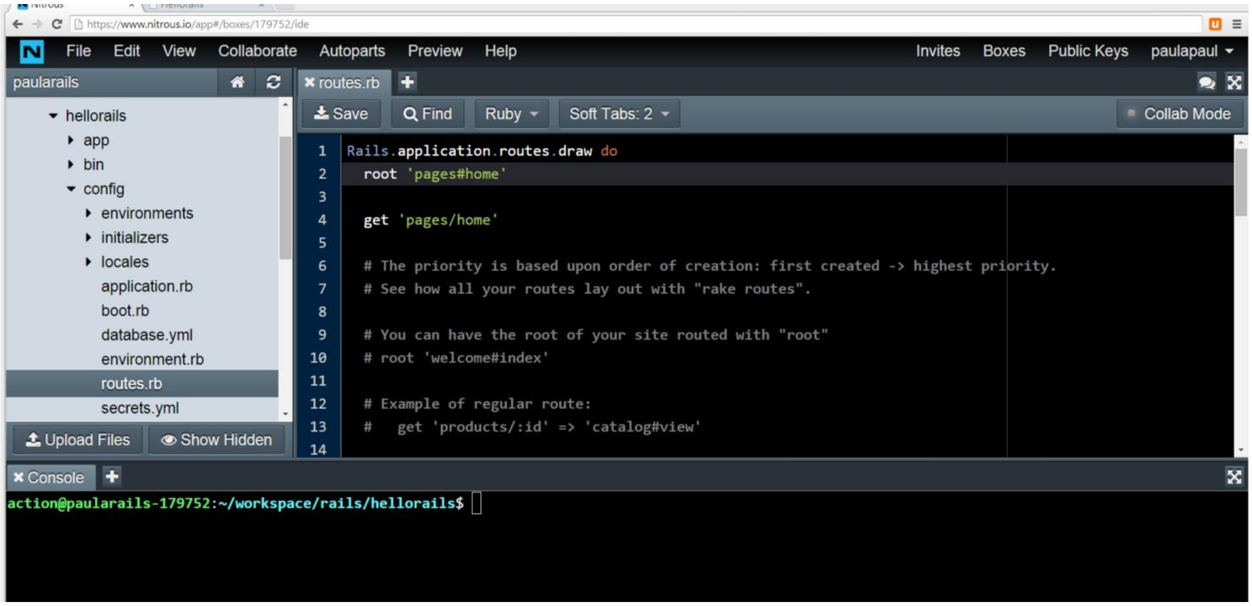
Rails Guides
Rails API
Ruby core
Ruby standard library

Click on the application file in …/config/routes.rb.

At the top of the file, enter the line:

root ‘pages#home’

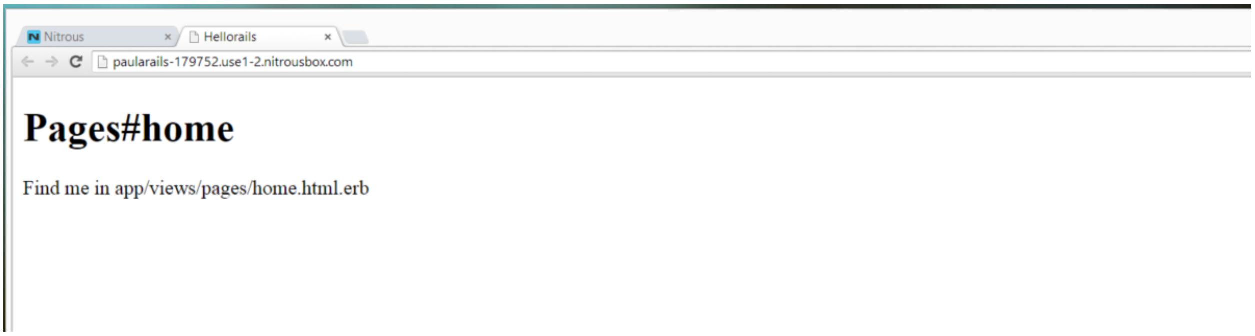
Once your file looks like this, be sure to save it:



```
Rails.application.routes.draw do
  root 'pages#home'

  get 'pages/home'

  # The priority is based upon order of creation: first created -> highest priority.
  # See how all your routes lay out with "rake routes".

  # You can have the root of your site routed with "root"
  # root 'welcome#index'

  # Example of regular route:
  #   get 'products/:id' => 'catalog#view'
```

```
action@paularails-179752:~/workspace/rails/hellorails$
```

Restart your server, and check out your application now!



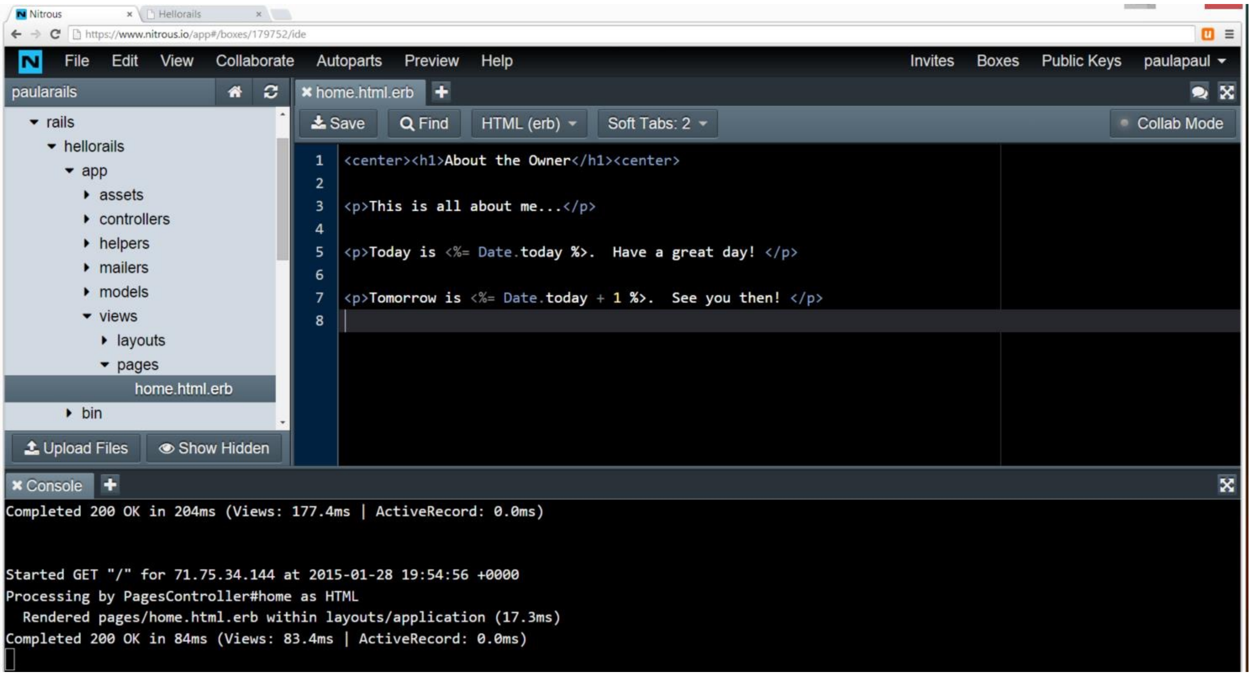
Pages#home

Find me in app/views/pages/home.html.erb

There it is! Your home page.

Now, let’s use some HTML and styling to make it look a little better, and include a little Ruby code as well.

The default page created by Rails tells us where to find the page, in **app/views/pages/home.html.erb**, so open that file and let's write a little more code...



```
<center><h1>About the Owner</h1><center>

<p>This is all about me...</p>

<p>Today is <%= Date.today %>.  Have a great day! </p>

<p>Tomorrow is <%= Date.today + 1 %>.  See you then! </p>

```

```
Completed 200 OK in 204ms (Views: 177.4ms | ActiveRecord: 0.0ms)


Started GET "/" for 71.75.34.144 at 2015-01-28 19:54:56 +0000
Processing by PagesController#home as HTML
  Rendered pages/home.html.erb within layouts/application (17.3ms)
Completed 200 OK in 84ms (Views: 83.4ms | ActiveRecord: 0.0ms)
```

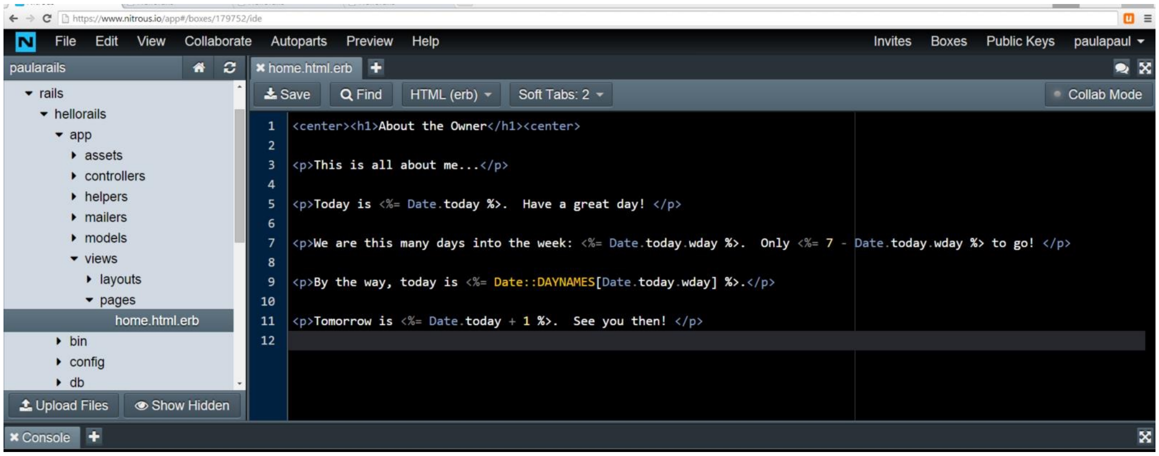
NOTE: The <center> tag shown above is not supported by HTML5, which is why we did not use it in class!

Notice we've added some HTML tags to create the heading, and have included some **E**mbedded **R**u**B**y Code (erb). Files named <filename>.html.erb are usually found in the views directory of your app folder, and can contain a mixture of HTML tags with embedded dynamic functions provided by the Ruby language.

This is the power of Ruby on Rails...

Rails is a framework that ties together Ruby and front end HTML/CSS/Javascript to create dynamic web applications.

What do you think happens if we change a few more lines, and do the following:

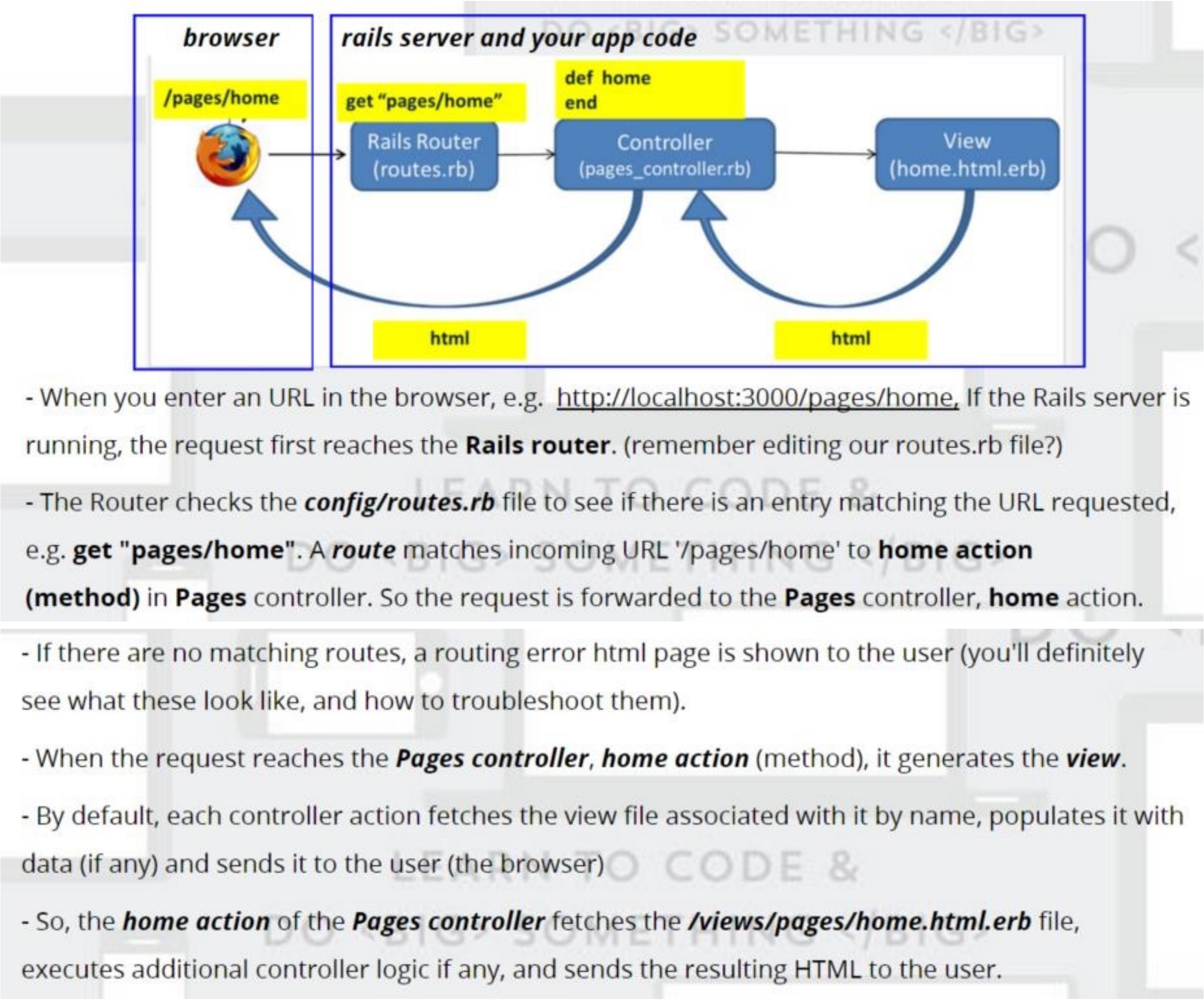


```
<center><h1>About the Owner</h1><center>

<p>This is all about me...</p>

<p>Today is <%= Date.today %>.  Have a great day! </p>

<p>We are this many days into the week: <%= Date.today.wday %>.  Only <%= 7 - Date.today.wday %> to go! </p>

<p>By the way, today is <%= Date::DAYNAMES[Date.today.wday] %>.</p>

<p>Tomorrow is <%= Date.today + 1 %>.  See you then! </p>

```

Can you update your home page to tell you when it is a class day?

We'll see the concept of blocks of reusable code now as we look more closely at the anatomy of a rails application. First, we'll go back to our 'Hello World' Rails app and add a Model to our existing Hello View and Controller.

We reviewed the MVC flow in class:



- When you enter an URL in the browser, e.g. http://localhost:3000/pages/home, If the Rails server is running, the request first reaches the **Rails router**. (remember editing our routes.rb file?)

- The Router checks the ***config/routes.rb*** file to see if there is an entry matching the URL requested, e.g. **get "pages/home"**. A ***route*** matches incoming URL '/pages/home' to **home action (method)** in **Pages** controller. So the request is forwarded to the **Pages** controller, **home** action.

- If there are no matching routes, a routing error html page is shown to the user (you'll definitely see what these look like, and how to troubleshoot them).

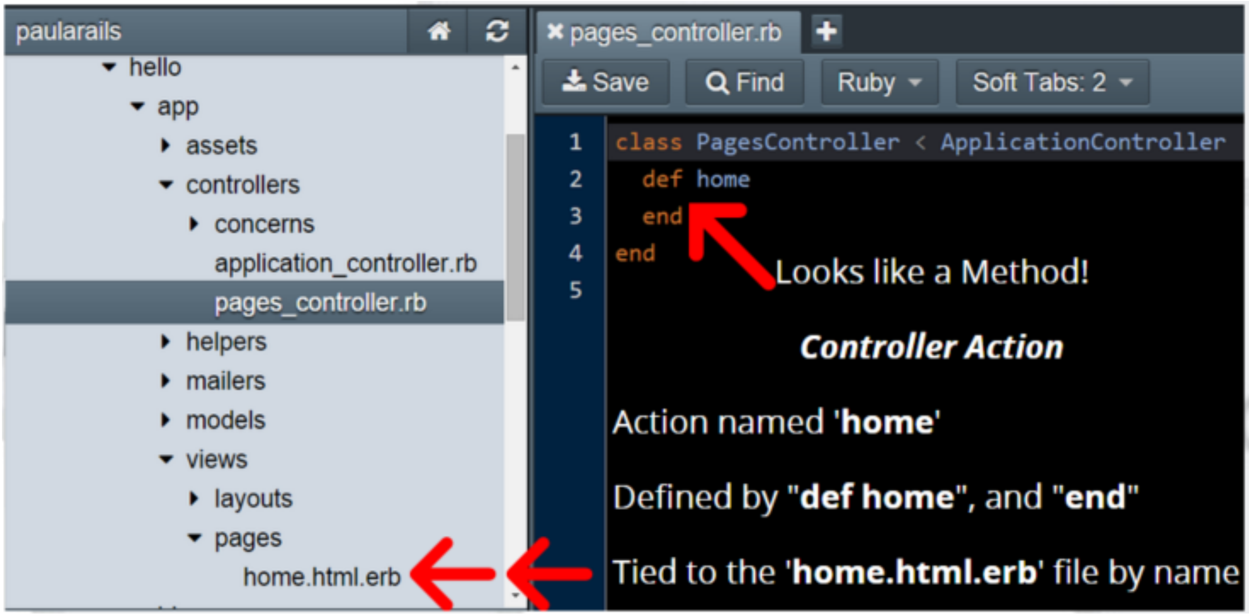
- When the request reaches the ***Pages controller***, ***home action*** (method), it generates the ***view***.

- By default, each controller action fetches the view file associated with it by name, populates it with data (if any) and sends it to the user (the browser)

- So, the ***home action*** of the ***Pages controller*** fetches the ***/views/pages/home.html.erb*** file, executes additional controller logic if any, and sends the resulting HTML to the user.

Our Hello rails app only has a View (**app/views/pages/home.html.erb**) and Controller (**app/controllers/pages_controller.rb**). The Pages controller has an empty method called 'home', to match the view (home.html.erb).

The methods in a controller are called '**Actions**'.



```ruby
class PagesController < ApplicationController
  def home
  end
end
```

Looks like a Method!

Controller Action

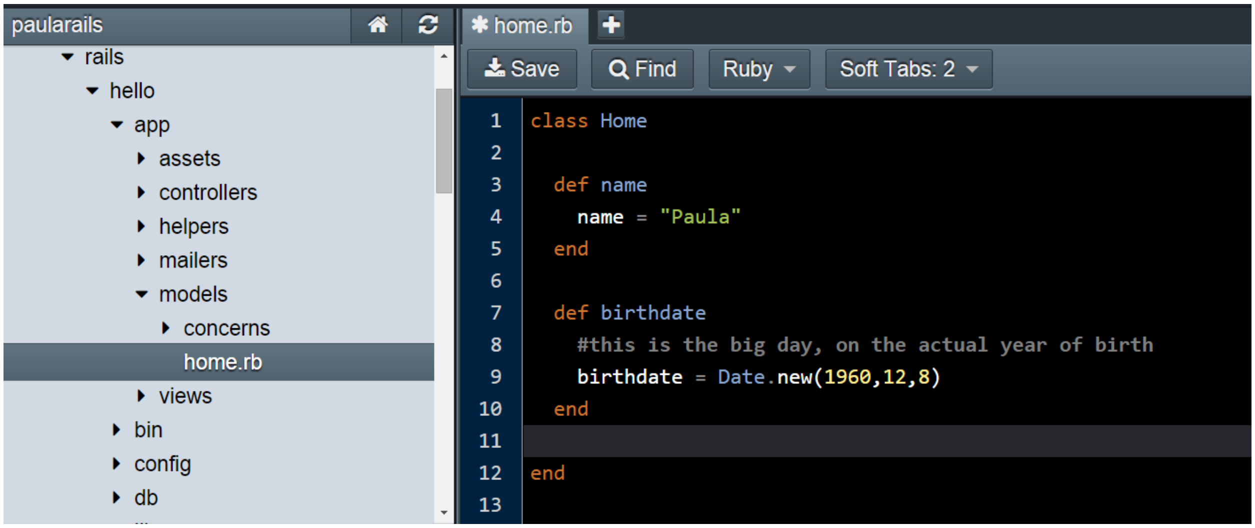
Action named '**home**'

Defined by "**def home**", and "**end**"

Tied to the '**home.html.erb**' file by name

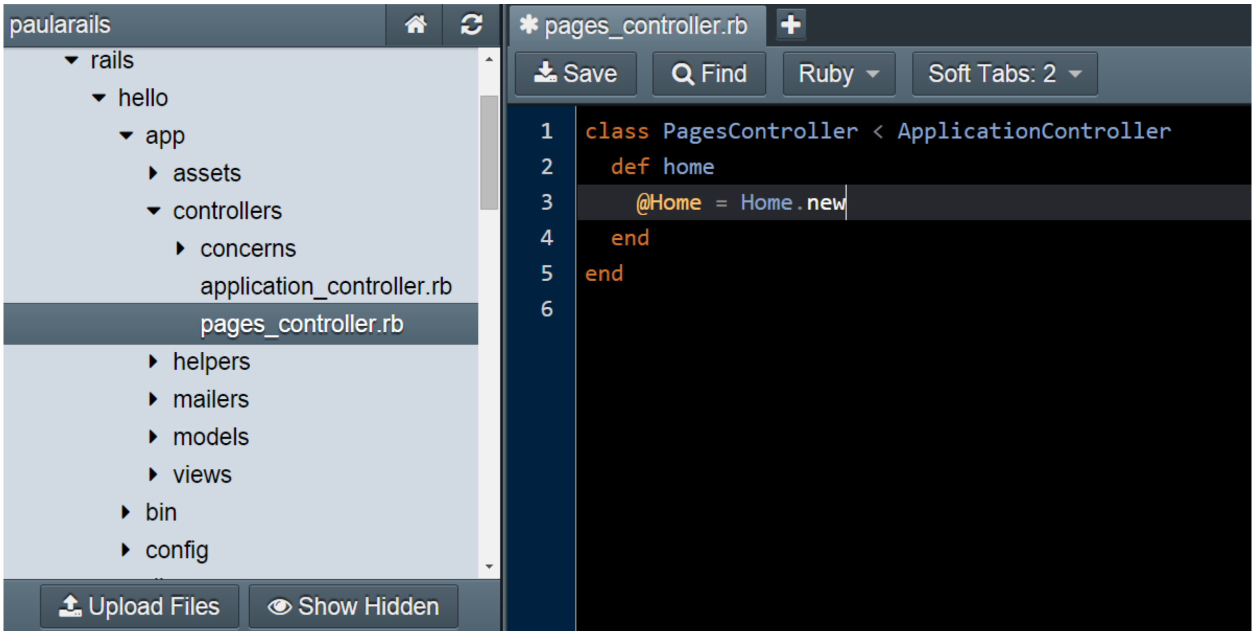
The Model files in your app allow you to store and manipulate data and persistent information associated with your app. Let's add a model to your Hello app.

To do this, right/control click on the 'models' folder in your Rails app and add a new file, named 'home.rb', then add some information about yourself:



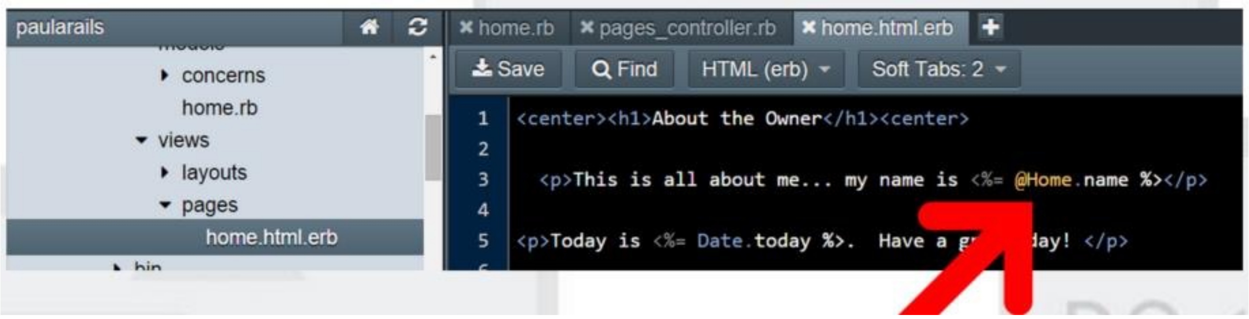
```ruby
class Home

  def name
    name = "Paula"
  end

  def birthdate
    #this is the big day, on the actual year of birth
    birthdate = Date.new(1960,12,8)
  end

end
```

Save your model file, then open your Pages controller, and connect the information in your model to the controller, in the 'home' action:



```ruby
class PagesController < ApplicationController
  def home
    @Home = Home.new
  end
end
```

Finally, flow that information in your Model into the View (via the Controller) by including some embedded Ruby in your app/views/pages/home.html.erb file:



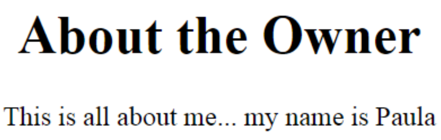
```erb
<center><h1>About the Owner</h1><center>

  <p>This is all about me... my name is <%= @Home.name %></p>

<p>Today is <%= Date.today %>.  Have a gr   day! </p>
```

Save all your files, then start your Rails server in terminal:

cntrl-c, then **rails s (**if your server was still running)

(Always be sure that you are in the workspace/rails/hello working directory!)

Preview your app at localhost:3000 in the browser to see what happens! Information from your Model is presented in the View, courtesy of your Controller...



About the Owner

This is all about me... my name is Paula

Model files are where we can integrate databases; front-end developers can create great looking styles in the View files, and we can tie it all together with additional logic in Controllers. MVC!

Now would be a great time to add and commit your files to git!