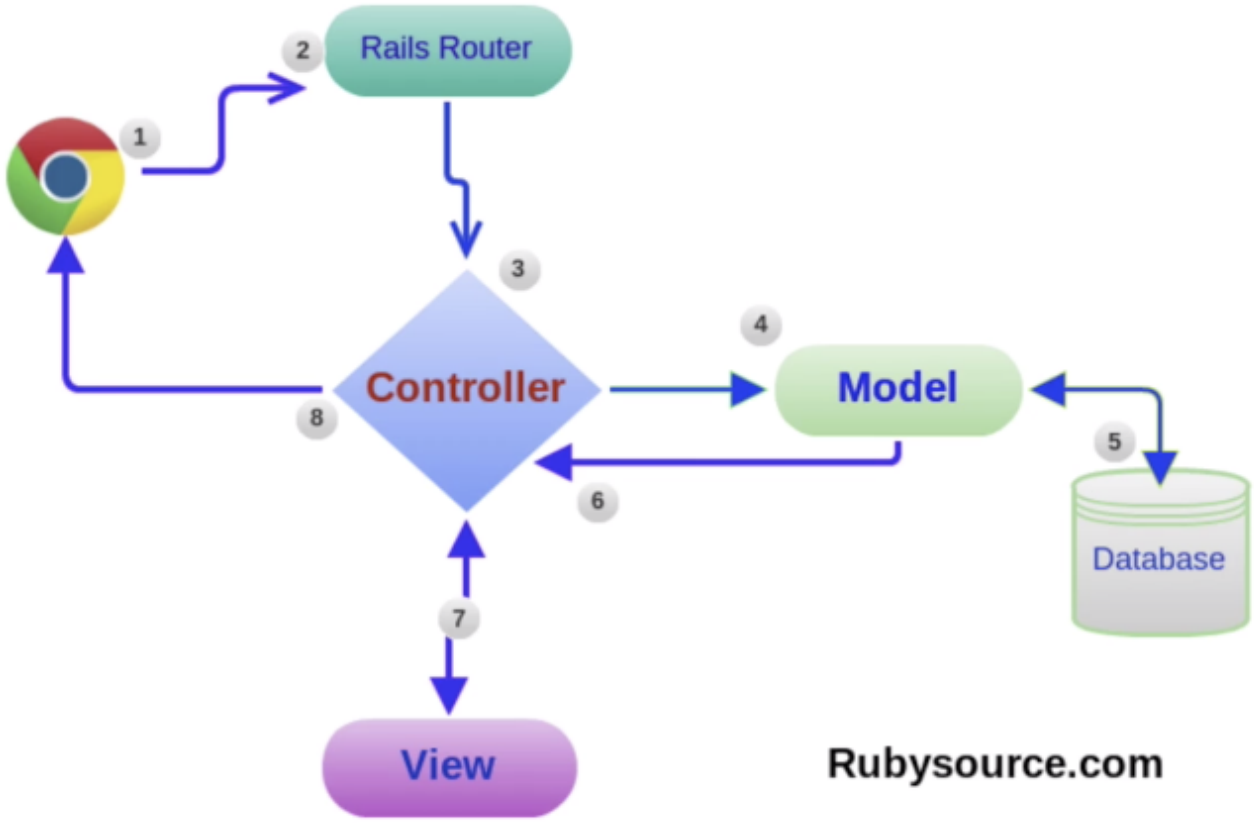
Rails Controllers - Index:

Link to lesson

- A link to the lesson can be found here.

What is a controller?



- The **controller** is essentially the **middle man** between our *model* or our data layer and the client side layout which is the *view*.
- In this lesson we are going to focus on the controller.
- Another important thing that a **controller manages** is the **requests** that are sent to our **application**.
- It also manages the **responses** that we send back.
- In the image above you can see the google chrome logo which represents a client computer (our computer).
- This is interacting with an application. To do this it will send:
    - HTTP request.

- First of all it will run through the **routes file** and they will be **directed** into the **controller.**
- Once all of the activity happens in the controller, then **controller** is then **responsible** for sending a **http response** back to the client.
- This is the premise of this lesson.

Generate a new rails app (project mgmt app)

```
rails new projects_managment_app
```

- Hit enter and run the generator (will take roughly 30 secs)
- Next change in to your rails app

  ```
  cd projects_management_app
  ```

- Just to verify that everything is working:

  ```
  rails s
  ```

- This will start up our rails server
- Then we go to our browser and enter:

  ```
  localhost: 3000
  ```

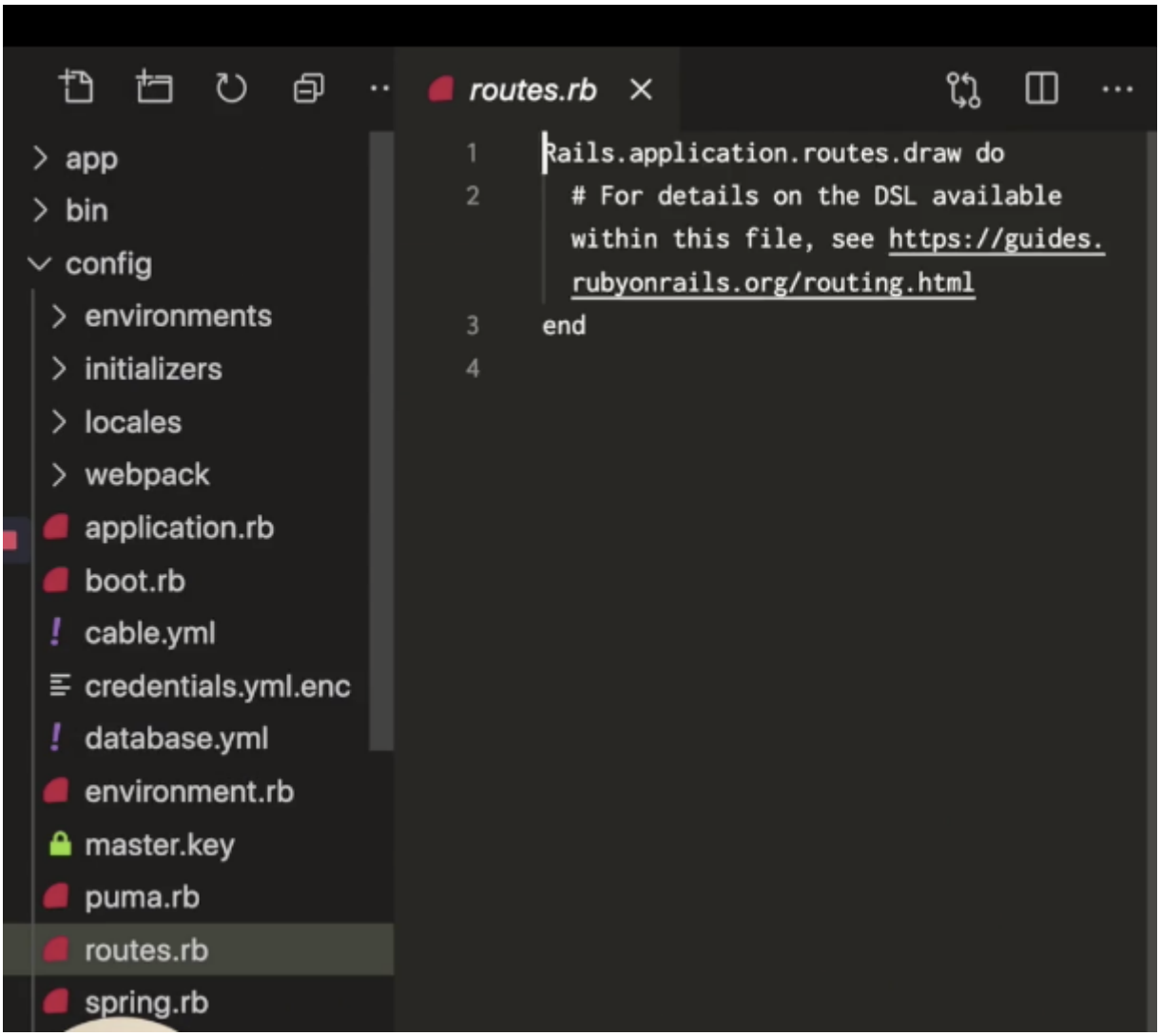
- You should get a page saying **"Yay you are on Rails"**

The First Thing You Need To Do!

- The first thing you need to do before you even start a Rails application is:
  - Create some routes (or a single route).
- So lets **stop running our rails server** and then open up the app in vs code

  ```
  code .
  ```

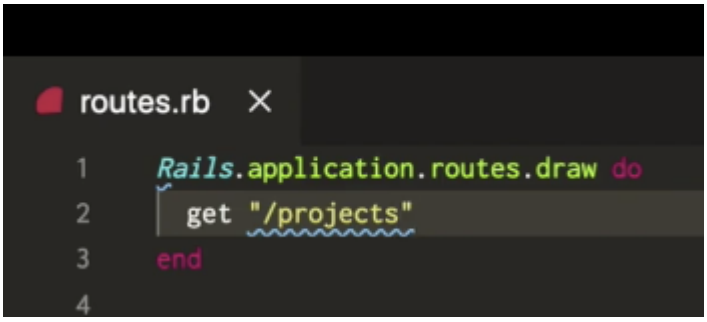
- Next lets open up a file called routes.rb



- Now we can remove the default comment.

Define first route

- The **route** we want in this first instance is a **Get Route.**
- A **GET request** is a request that is going to be **sent to the client.**
- In this example, the route will be /projects (see image below):



- This is going to go to a **controller action.**

```
routes.rb
1  Rails.application.routes.draw do
2    get "/projects", to: "controller#action"
3  end
4
```

NOTE: none of these have been created yet.

- For now these are **just place holders.**
- **Before** we create our **controller**, lets run our rails server:

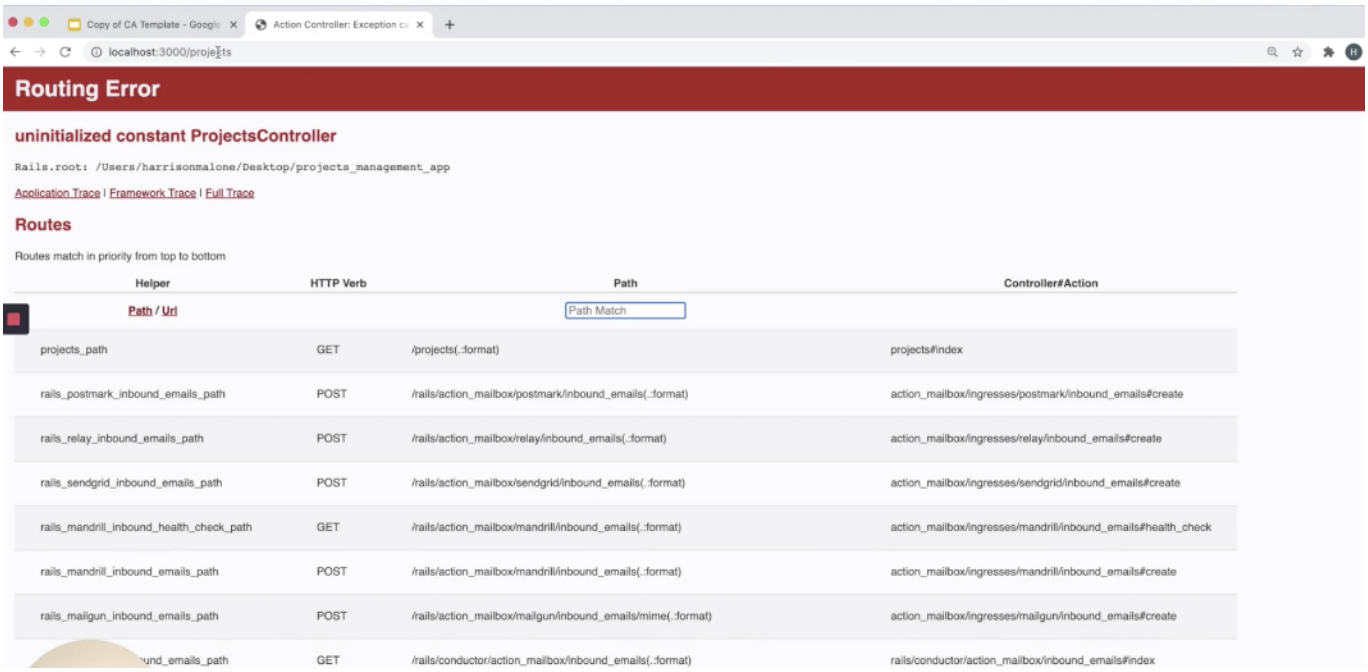
```
rails s
```

```
~/Desktop/projects_management_app rails s
=> Booting Puma
=> Rails 6.0.3.3 application starting in development
=> Run `rails server --help` for more startup options
Puma starting in single mode...
* Version 4.3.6 (ruby 2.7.1-p83), codename: Mysterious T
raveller
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://127.0.0.1:3000
* Listening on tcp://[::1]:3000
Use Ctrl-C to stop
```

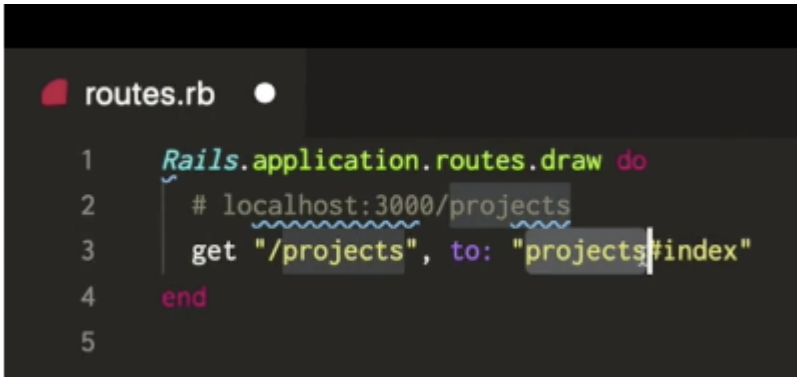
- Then lets go to localhost: 3000/projects

```
routes.rb
1  Rails.application.routes.draw do
2    # localhost:3000/projects
3    get "/projects", to: "projects#index"
4  end
```

- Because this is a **GET request**, we can make this request just from the browser.
- In the url bar



- Now we are getting an error:
- unititialized constat ProjectsController
- This is occuring because we are saying in the route:
  - projects controller
  - And this projects controller currently DOES NOT exist.



- So we need to create the projects controllers.

Creating projects controllers

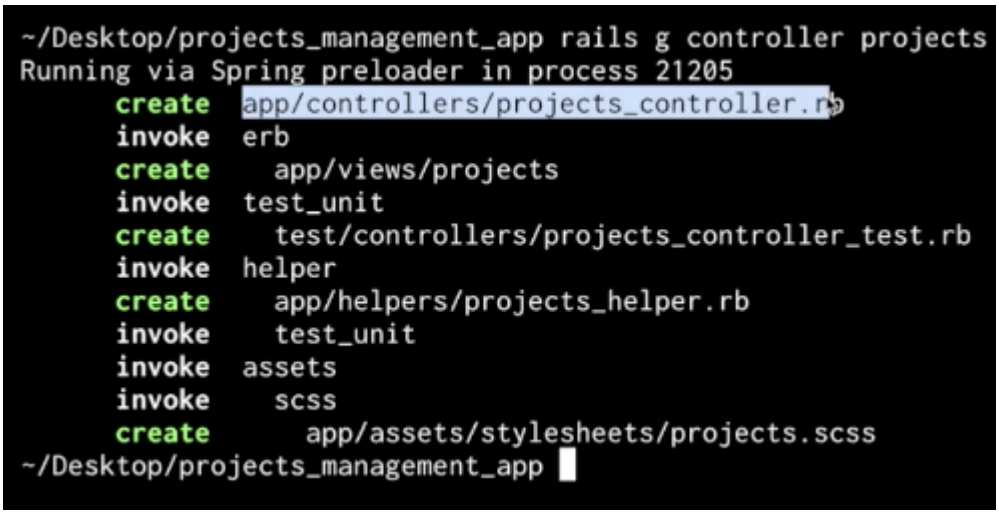
- The slow way would be to go into our app directory
- Then go into the controllers directory and then define a new file called projects_controller.rb and then write some code in this particular file.
- The preferred way to create a controller is (stop running server first)
- Go into terminal and use the command:

```
rails g controller (name of controller)
```

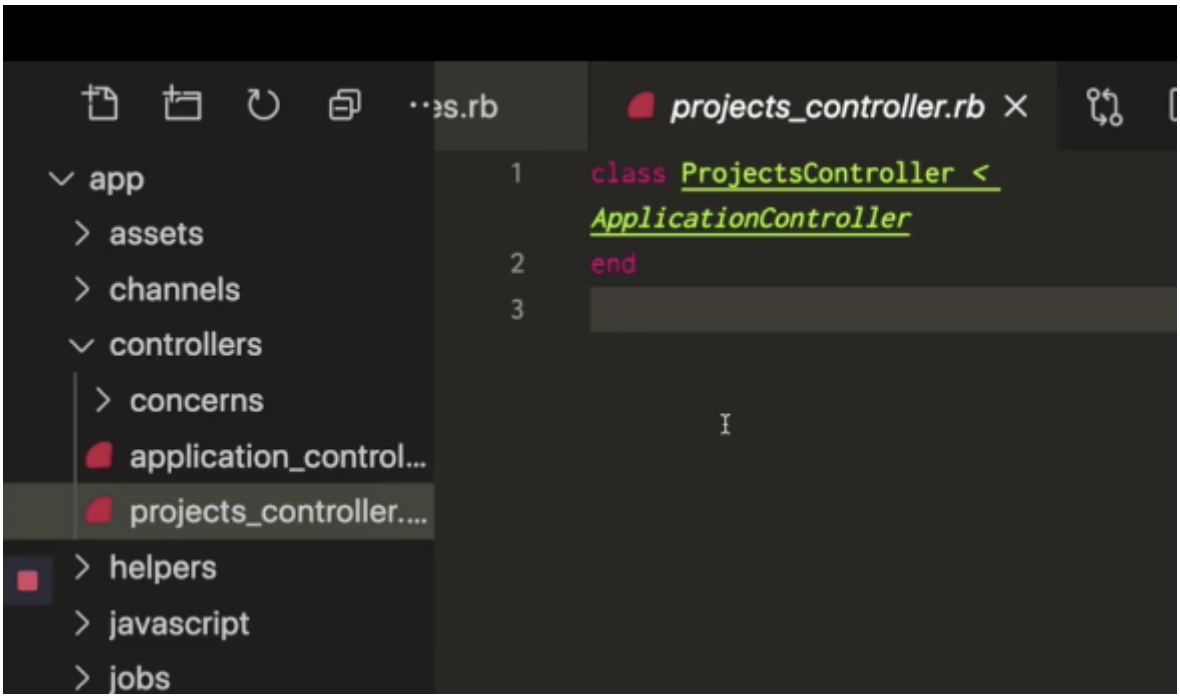
- This is the rails controller generator command.

```
rails g controller projects
```

- A bunch of files will be created after running this command.
- All that we care about is the first one:



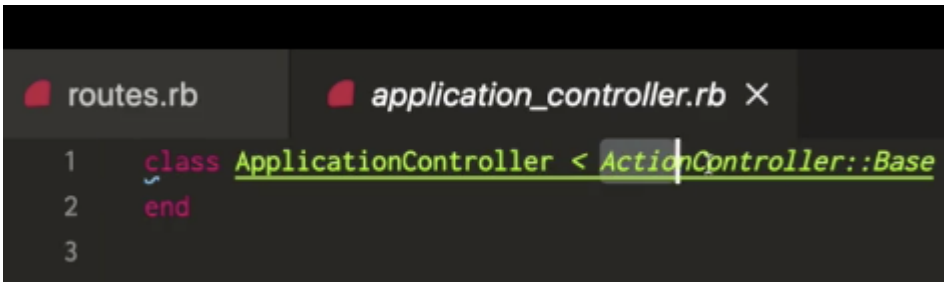
- This displays that a new file has been added to the app directory inside of the controllers directory and we now have this projects_controller.rb file (which is our controller file).
- We can see that in vs too:



All of the controllers that we create will inherit from ApplicationController

- This is a file that we have when we run the rails new command.

- If we go to the ApplicationController file:

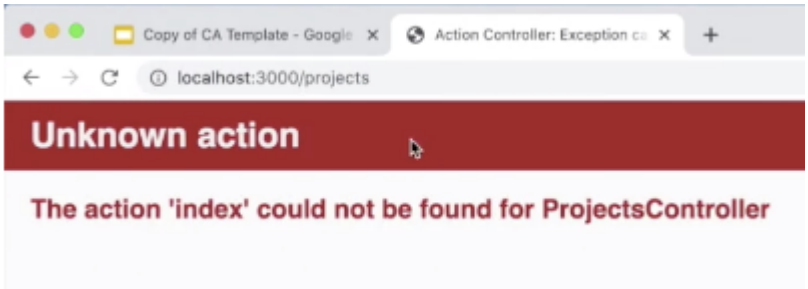


- We can see that the ApplicationController is inheriting from the ActionContoller module and the Base class in that particular module.
- We don't really need to understand much about this apart from that fact that:
  - **Through inheritance** we have a **bunch of methods** that we can use in our **controllers** that we **DEFINE ourselves**.
- Now lets run our server again:

```
rails s
```

Then refresh our browser.

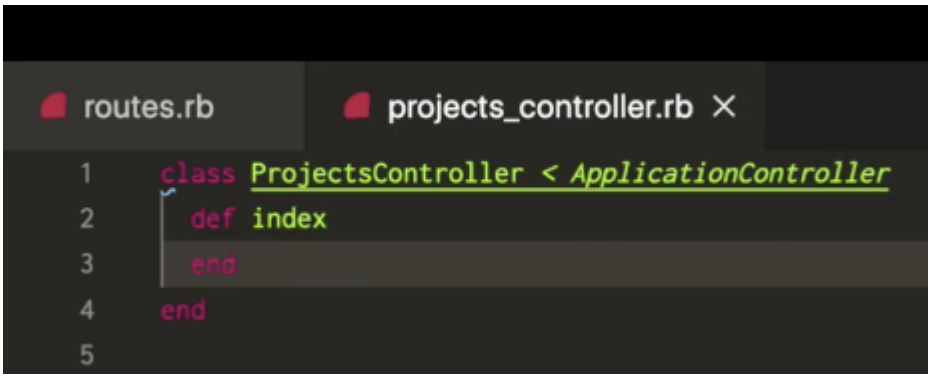
- And we have another error:



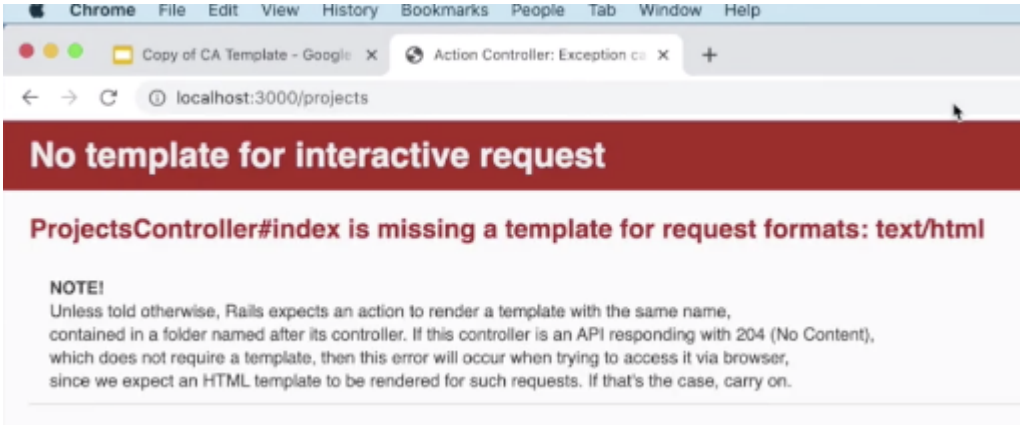
Whenever we see this word ACTION in realtion to a Controller:

- It is talking about a method:
  - A controller method.
- So what we are missing right now is a method in our projects controller named index.
- To fix this we go to our projects_controller.rb file and add a method called index to our ProjectsController.

```
def index
end
```



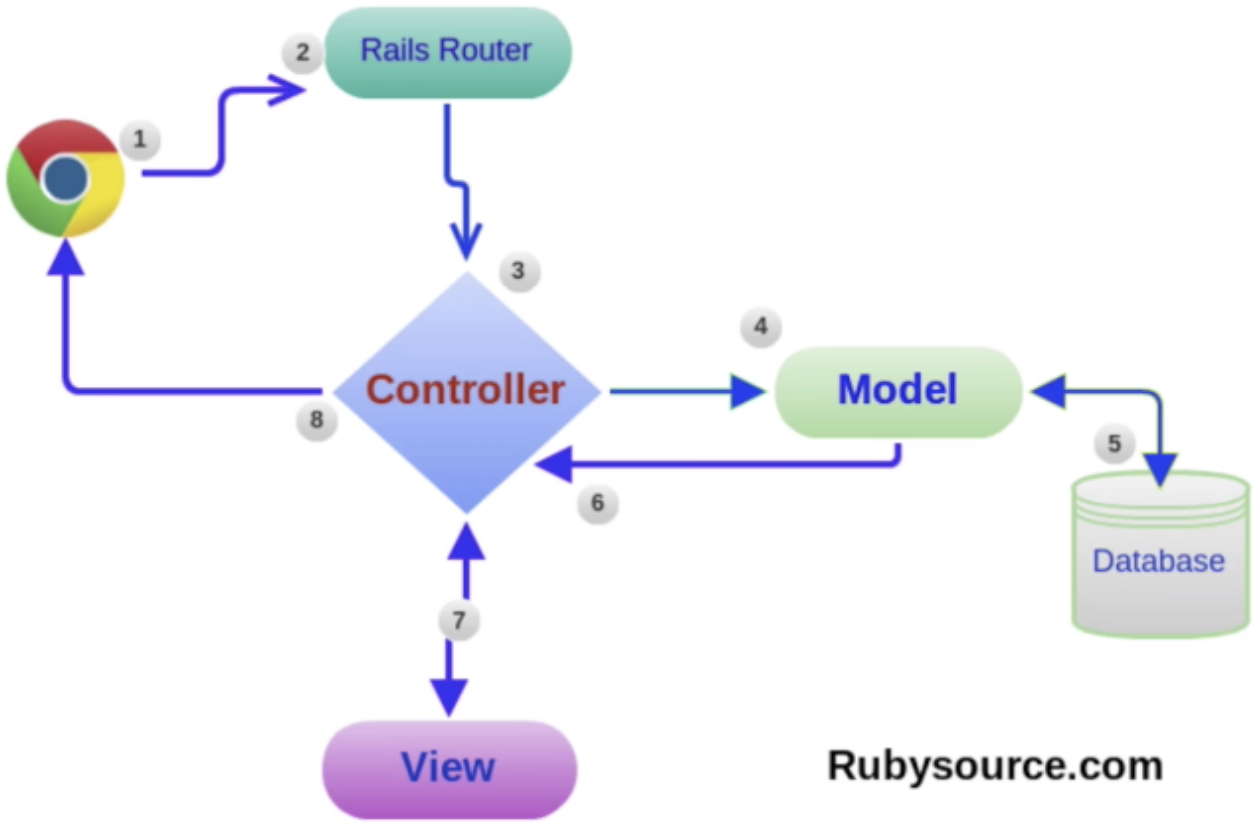
- Then we can refersh in the browser.
- And we are getting another error



- This error is a little more confusing as it can mean a number of different things.
- What is boils down to is:

WE ARE NOT SENDING BACK A RESPONSE TO THE CLIENT

- If we take a look at our image again:



- Our chrome client is sending a request.
- It is going through our rails router.
- And it is going to our controller.

The problem we currently have is:

- The controller and the action in the controller (specifically the index action)...
- IS NOT sending back a response to our client.
- There is no response being sent back so this number 8 line (in the image) is not occuring.

Send back a response (html)

- To send back a response from our controller actions we have a number of different ways to do so.
- One way in which we can send a response back to the client is:
    - To use the **render** keyword.
- Render will send back a response to our client.

```
class ProjectsController < ApplicationController
  def index
```

```
    render
  end
```

Render

- Render is a method.
- Render can take a number of different arguments.
- With a number of different notations. so

Render Notation

- keyword argument notation:
- "I want render to send back some html"
- We will need to pass a string.

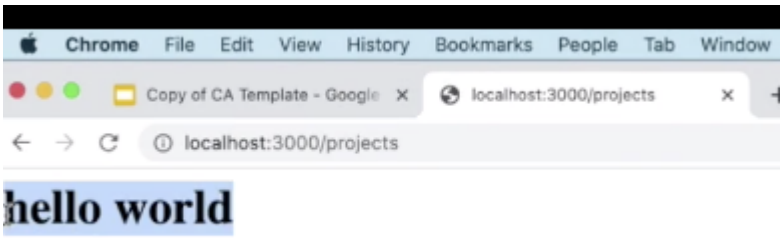
```
class ProjectsController < ApplicationController
  def index
    render html: "<h1> Hello World!</h1>"
  end
```

- We also need to chain on this html_safe method.

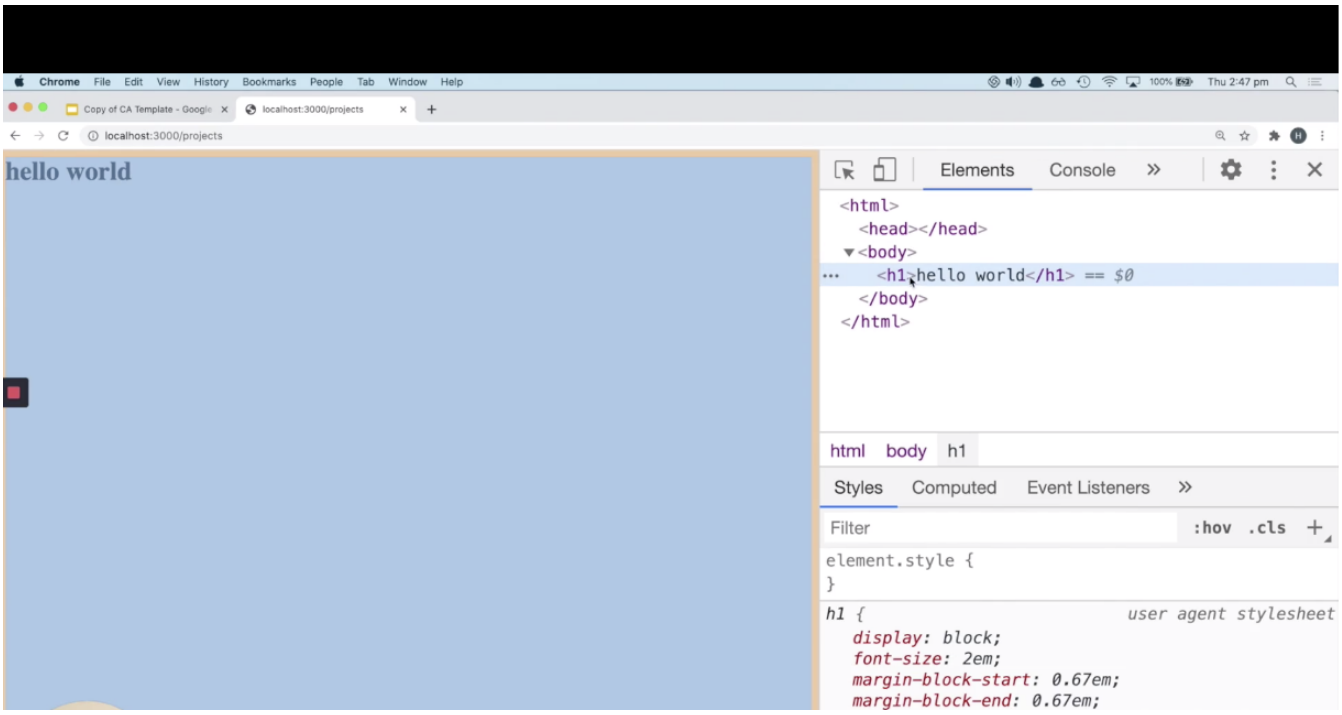
```
  class ProjectsController < ApplicationController
  def index
    render html: "<h1> hello word</h1>".html_safe
  end
```

Refresh browser

- You can see that hello world has been sent from:
    - Our controller action BACK to the client.



- Now it is displaying our html in the browser.
- If we look with ou dev tools we can see the h1 is there:

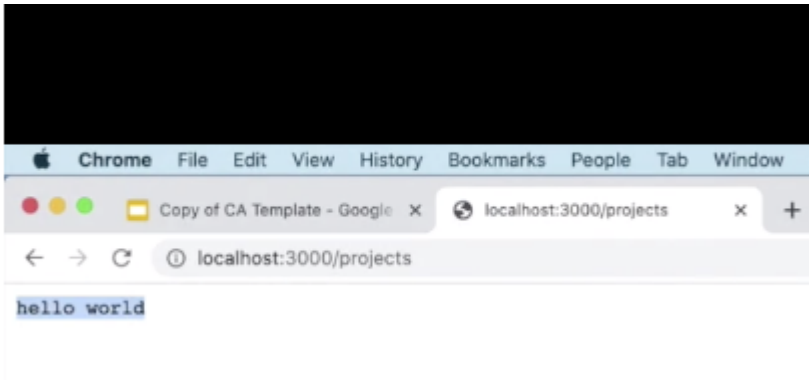


Send back a response (plain text)

```
class ProjectsController < ApplicationController
    def index
      render plain: "hello word"
    end
```

Refresh browser

- We will get some plain text in the browser.



Send back a response (json)

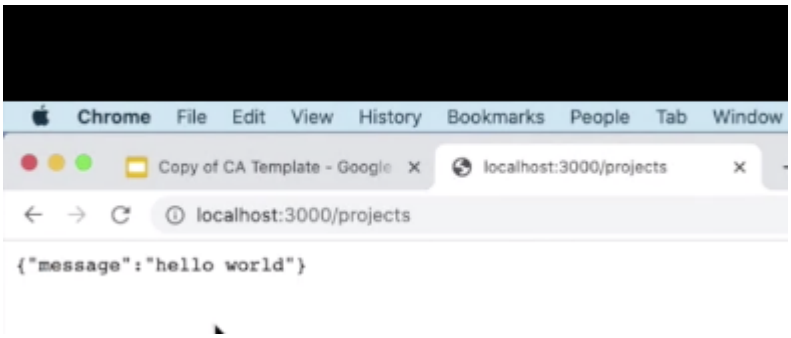
- Json will accept a string:

```
class ProjectsController < ApplicationController
  def index
    render json: "hello word"
  end
```

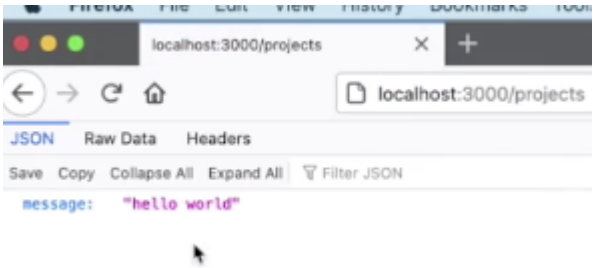
- Json will accept a hash:

```
class ProjectsController < ApplicationController
  def index
    render json: { message: "hello word" }
  end
```

- if we send that back to the client (refresh browser)
- In **chrome**, we will see:



- In **firefox**, we will see:



- That we are acutally sending JSON back to the client.

Send back another json response

- We are going to check th hash that we send backin the response.

```
class ProjectsController < ApplicationController
  def index
```

```
    render json: { message: "hello word" }
  end
```

- Instead of the placeholder above, we are going to send back some real projects back to the client. We

Define a variable

- We will name that variable projects:

```
class ProjectsController < ApplicationController
  def index
    projects =
    render json: { message: "hello word" }
  end
```

- Projects is going to store an array of hashes
- Each hash will represent one single project
- Each has will have a unique key value attribute or key value pair (id)

```
class ProjectsController < ApplicationController
  def index
    projects = [
        {
          id: 1,
          name: "rails project",
          github_status: false
        },
        {
          id: 2,
          name: "terminal app",
          github_status: true
        }
    ]
    render json: { message: "hello word" }
  end
```

- Instead of sending back our placeholder key value pair (message hello world), lets change message to projects (this will be our key that we are sending back in the response)
- The value can be the array of hashes defined in the the action. So we can just pass the variable projects - as a value.
- In this manner, the value is an array of hashes.

```
render json: { projects: projects }
```

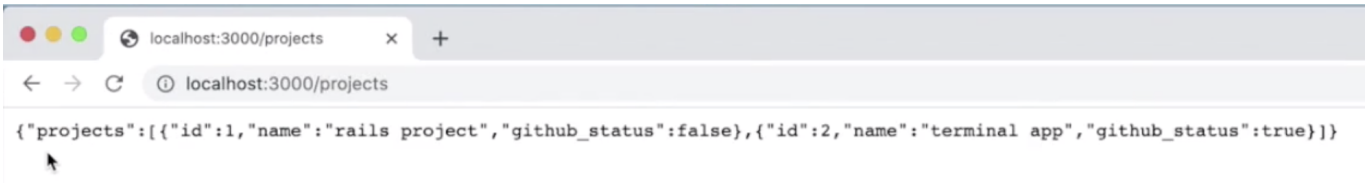
- Code below:

```
class ProjectsController < ApplicationController
  def index
    projects = [
        {
          id: 1,
          name: "rails project",
          github_status: false
        },
```

```
            {
              id: 2,
              name: "terminal app",
              github_status: true
            }
        ]
        render json: { projects: projects }
      end
```

Back to browser

- If we refresh and go back to the browser: 22
- We are no longer getting our placeholder message but rather, we are getting an array of hashes with some real data.



One step further

- We could just send back the array of hashes as a response WITHOUT the key to indentify that we are actually sending back projects.

```
    render json: projects
```

Code:

```
class ProjectsController < ApplicationController
        def index
          projects = [
              {
                id: 1,
                name: "rails project",
                github_status: false
              },
              {
                id: 2,
                name: "terminal app",
                github_status: true
              }
          ]
          render json: projects
        end
```

routes.rb | projects_controller.rb

```ruby
class ProjectsController < ApplicationController
  def index
    projects = [
      {
        id: 1,
        name: "rails project",
        github_status: false
      },
      {
        id: 2,
        name: "terminal app",
        github_status: true
      }
    ]
    render json: projects
  end
end
```

Back to browser

- If we save and refresh our browser:
- Now we are just getting back the array of hashes back to the client side.

localhost:3000/projects

localhost:3000/projects

[{"id":1,"name":"rails project","github_status":false},{"id":2,"name":"terminal app","github_status":true}]

Index action

- The convention for an index action in a Rails application is to:

Send back all of the resources to the client.

- By resources, we mean projects (in this example)
- If we were making a "to do" app:

    - in the index action you would send back all of the "to do's".
- If you were making an app that was storing all of the bookings you would:
    - Send back all of the bookings in the index action inside of the bookings controller.