Rails Controllers - Show:

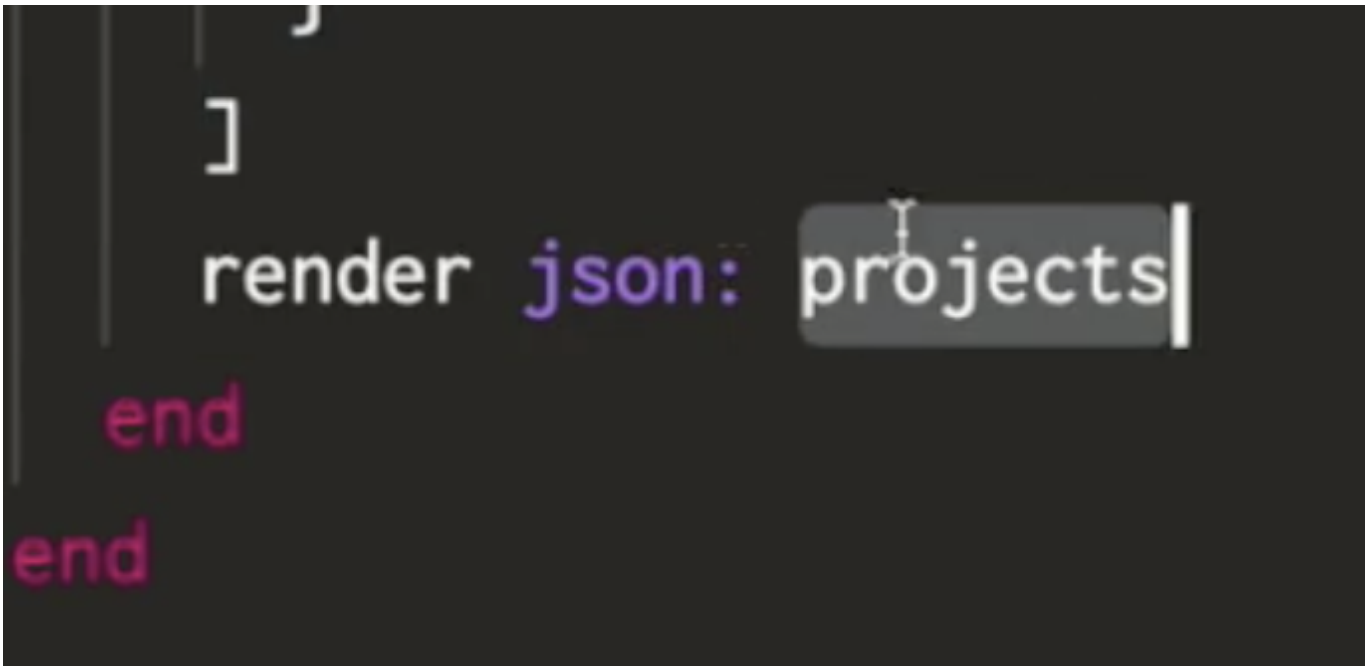
Link to lesson

- A link to the lesson can be found here.
- We saw in the previous lesson that we were sending back all of our projects.
- Basically we were storing all of our projects in an array

routes.rb projects_controller.rb

```
1   class ProjectsController < ApplicationController
2     def index
3       # send back all of the resources to the client
4       projects = [
5         {
6           id: 1,
7           name: "rails project",
8           github_status: false
9         },
10        {
11          id: 2,
12          name: "terminal app",
13          github_status: true
14        }
15      ]
16      render json: projects
17    end
18  end
```

- Then we were sending back a response to the client (see image below)



- The response was formatted as JSON; meaning we were sending back a JSON data structure or a JSON format back to the client side.
- So when we typed into our URL

```
localhost: 3000/projects
```

- We saw our array of hashes

Sending back one project at a time (see image below)

- The way in which we can achieve this is by:
    - Defining a new route and by having something particular in that route.

Route URL

- What we want the route to look like is something like this:

```
localhost: 3000/projects/(a number)
```

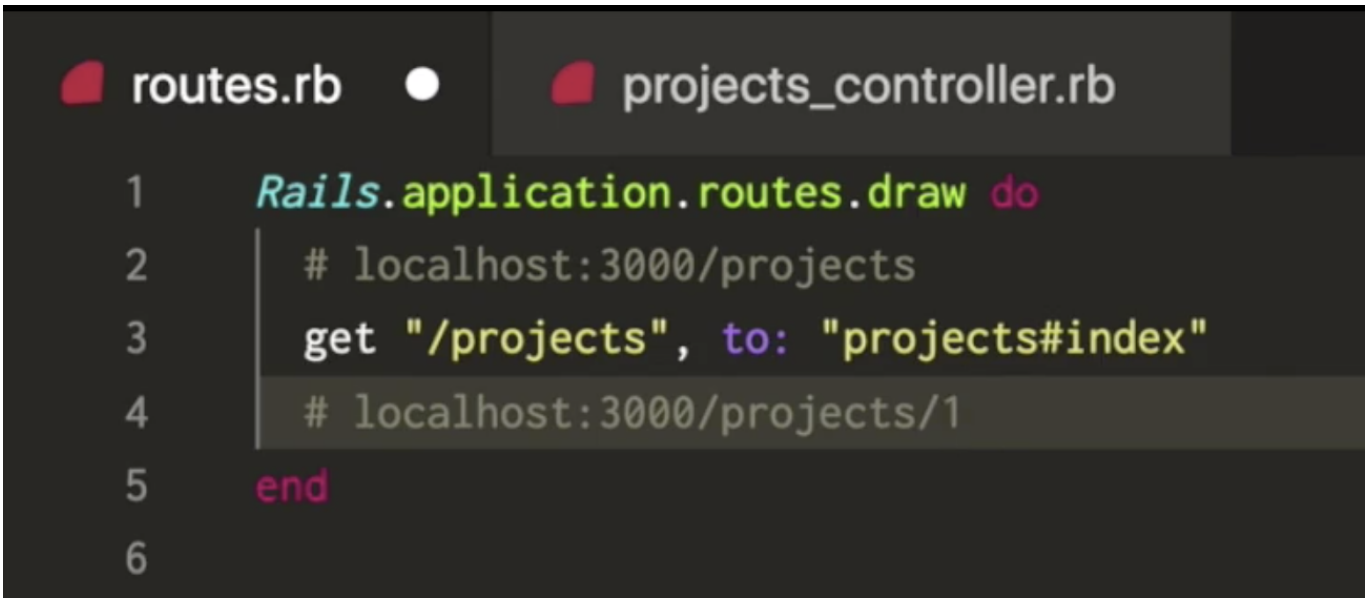
For example:

```
localhost: 3000/projects/1
```

OR

```
localhost: 3000/projects/2
```

ETC



- This number is going to correlate to an id (image below highlights id 1)

routes.rb projects_controller.rb

```
1   class ProjectsController < ApplicationController
2     def index
3       # send back all of the resources to the client
4       projects = [
5         {
6           id: 1,
7           name: "rails project",
8           github_status: false
9         },
10        {
11          id: 2,
12          name: "terminal app",
13          github_status: true
14        }
15      ]
16      render json: projects
17    end
18  end
```

- Therefore, if we type in the number 1 into our url:

```
localhost: 3000/projects/1
```

- We should be able to fetch our correcting id from our projects_controllers file (the particular hash in the image below):

routes.rb ● projects_controller.rb ×

```ruby
class ProjectsController < ApplicationController
  def index
    # send back all of the resources to the client
    projects = [
      {
        id: 1,
        name: "rails project",
        github_status: false
      },
      {
        id: 2,
        name: "terminal app",
        github_status: true
      }
    ]
    render json: projects
  end
end
```

- The same goes if we were to type in the number 2 into our url:

```
localhost: 3000/projects/2
```

- We should be able to fetch the id 2 from our projects_controller file (see image below):

routes.rb projects_controller.rb

```ruby
class ProjectsController < ApplicationController
  def index
    # send back all of the resources to the client
    projects = [
      {
        id: 1,
        name: "rails project",
        github_status: false
      },
      {
        id: 2,
        name: "terminal app",
        github_status: true
      }
    ]
    render json: projects
  end
end
```

Define the Route

- To define the route head to the routes.rb file
- This will use a get route (get request)
- Then we can add the name of the route.

```
get “/projects/“
```

- Then the number could be anything as it refers to the particular project we want to fetch.

Passing a placeholder

- This will represent the part of the route that can change.
- The notation for passing a placeholder is:
    - A colon and then the placeholder name for this particular route (id in this example)

      get “/projects/:id”

routes.rb ● projects_controller.rb

```
1  Rails.application.routes.draw do
2    # localhost:3000/projects
3    get "/projects", to: "projects#index"
4    # localhost:3000/projects/5
5    get "/projects/:id"
6  end
```

- id will represent some kind of number (could be 1 could be 2 could be 10 etc)
- Let’s now define our key value pair
- This will go to the projects folder
- remembering that the first part of the to: is the name of the controller (projects in this example)
- The second part is the name of the controller action - separated with a #
- In this example we are using show as the controller action (a convention in Rails)

```
get “/projects/:id”, to: “projects#show”
```

routes.rb × projects_controller.rb

```
1  Rails.application.routes.draw do
2    # localhost:3000/projects
3    get "/projects", to: "projects#index"
4    # localhost:3000/projects/5
5    get "/projects/:id", to: "projects#show"
6  end
```

Show action

- The show action is for sending back only one particular resource (one project in this example)

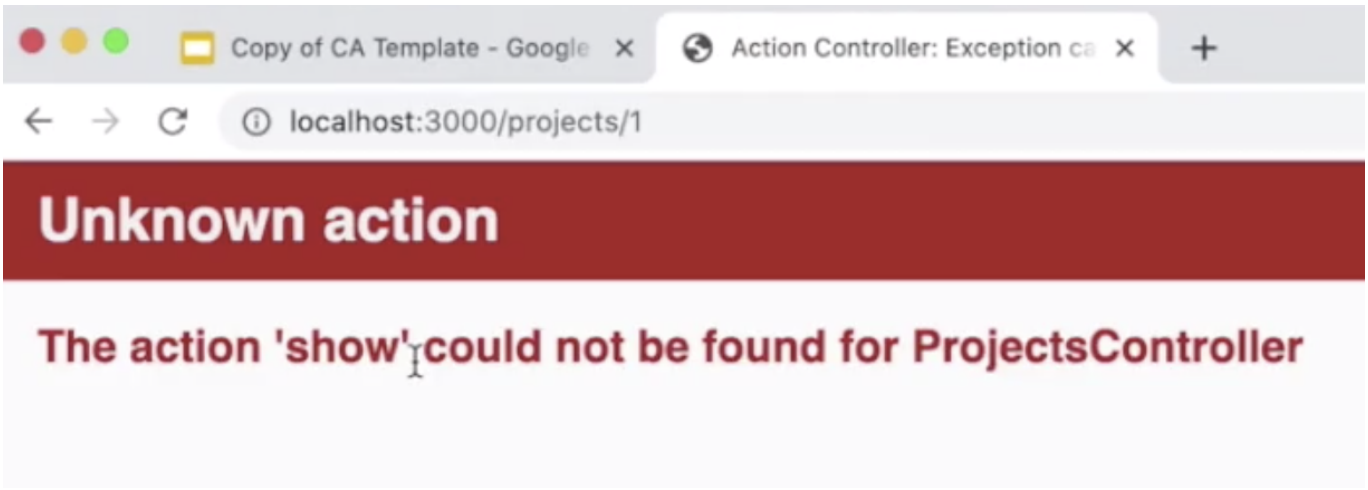
Checking show action

- Let's type out our url (see images below) into our browser and see what we get.

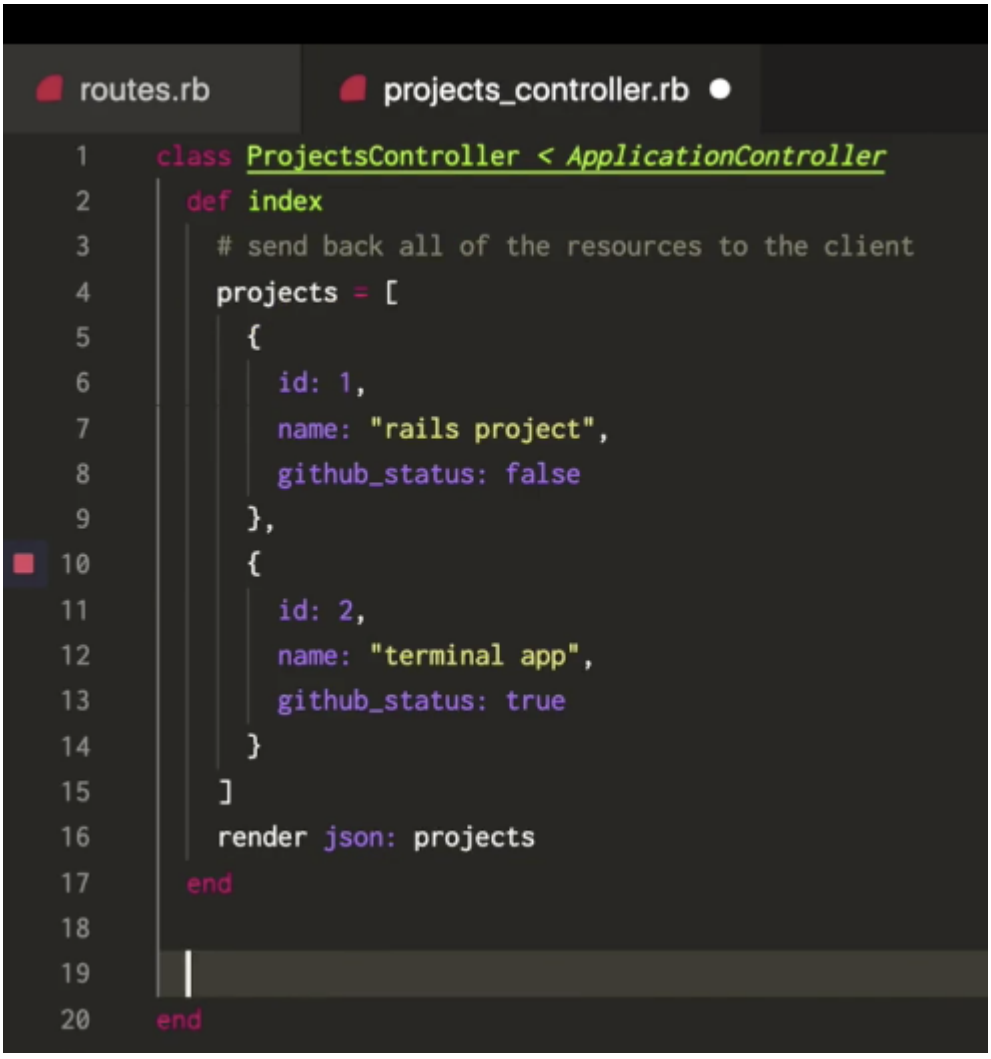
routes.rb ● projects_controller.rb

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

- We are getting an error:



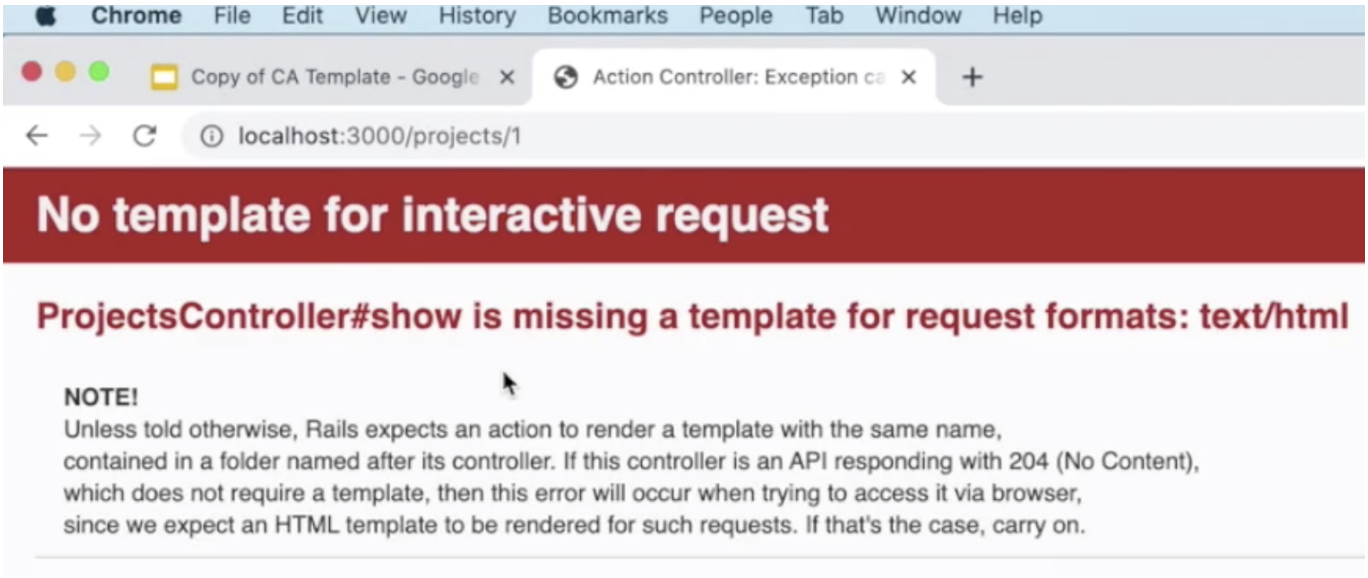
- The error is pretty clear and is an easy fix.
- First lets go to our projects_controller.rb file



- All we have to do is define the show action

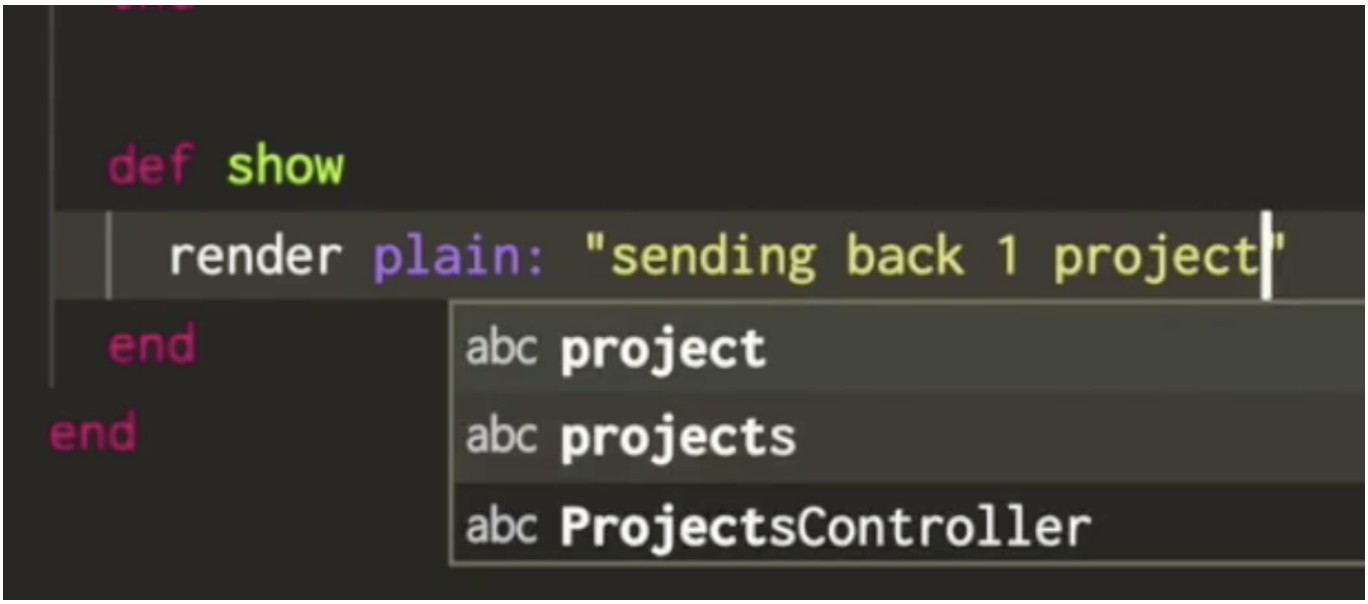
```
def show
end
```

- Now let's refresh our browser.
- Add we see a different error (see image below):

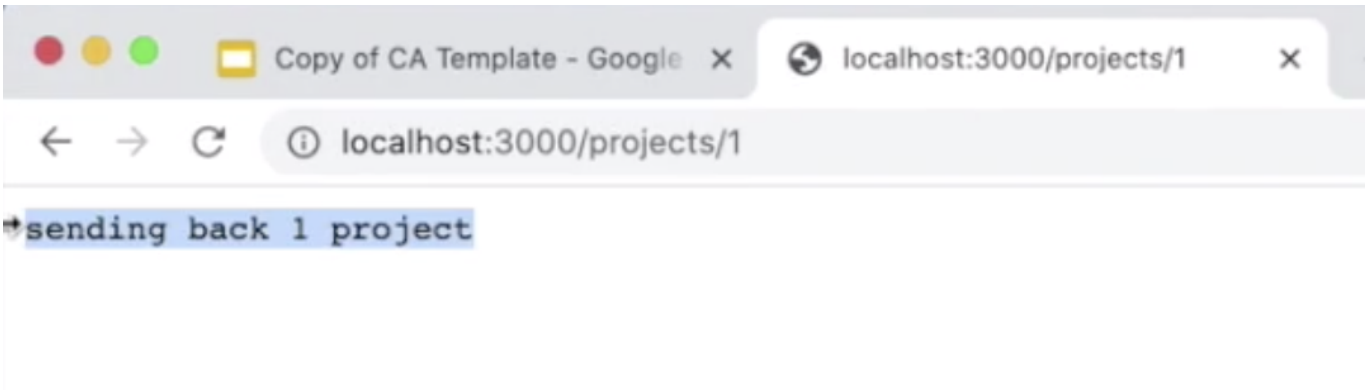


- Now it's saying that the show action is not sending back a response. (that's what missing a template refers to).
- We can fix this by using the render key word and add some plain text to see it's working

```
def show
  render plain: “sending back 1 project”
end
```



- Now let's refresh the browser to see if this renders.



- Although it is working, it is not very useful given we are just sending back plain text and not being back a project from our array of hashes.

routes.rb projects_controller.rb

```
3     # send back all of the resources to the client
4     projects = [
5       {
6         id: 1,
7         name: "rails project",
8         github_status: false
9       },
10      {
11        id: 2,
12        name: "terminal app",
13        github_status: true
14      }
15    ]
16    render json: projects
17  end
18
19  def show
20    render plain: "sending back 1 project"
21  end
22 end
```

Finding item

- First step is to grab our array of hashes.
- This won't be very DRY but we can refactor later.

NOTE: show is for sending back only one resource.

- A method we can use to find one item in an array is:

```
.find
```

- We can call the dot find method on an array (in this case it's our projects array)

```
projects.find
```

- Find then takes a block:

```
projects.find do |project|

end
```

- Within the do block we are going to find a project based on it's id
- Because each project is a hash, we will use the hash notation:

```
projects.find do |project|
  project[:id]
end
```

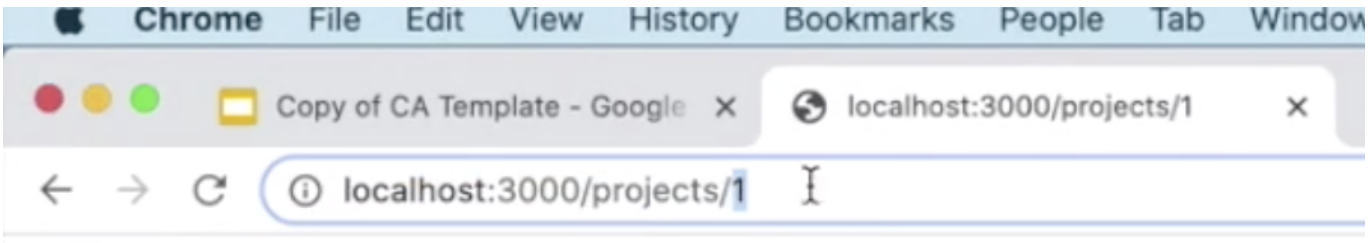
- We want to find the one that is equal to something

```
projects.find do |project|
  project[:id] ==something
end
```

routes.rb projects_controller.rb

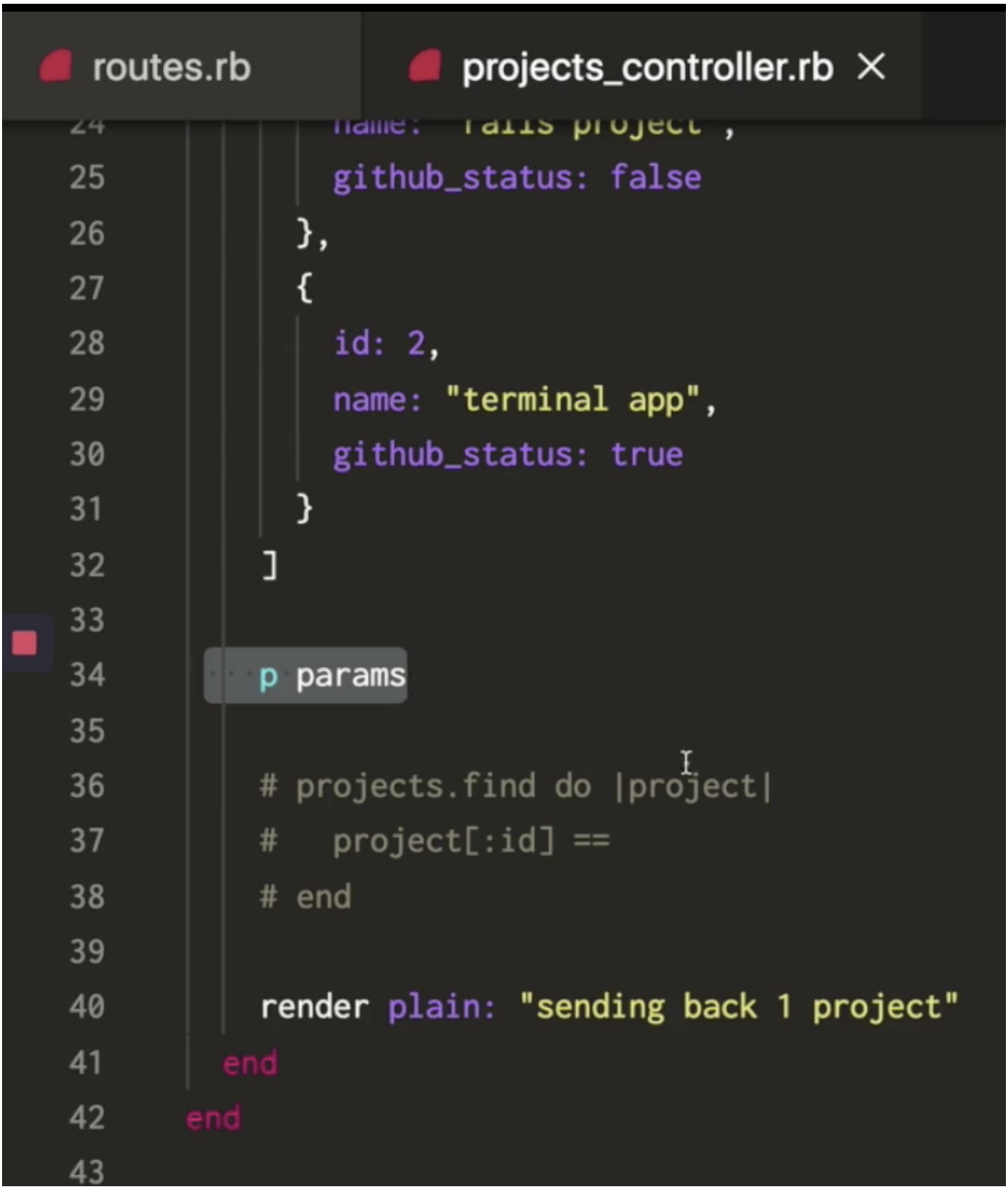
```
24            name: "rails project",
25            github_status: false
26          },
27          {
28            id: 2,
29            name: "terminal app",
30            github_status: true
31          }
32        ]
33
34        projects.find do |project|
35          project[:id] == something
36        end
37
38        render plain: "sending back 1 project"
39      end
40    end
41
```

- The “something” is going to represent the number in the url (see image below):



Accessing URL path in controller action

- We can use a special method params
- If we comment out the block of code we just did and print params



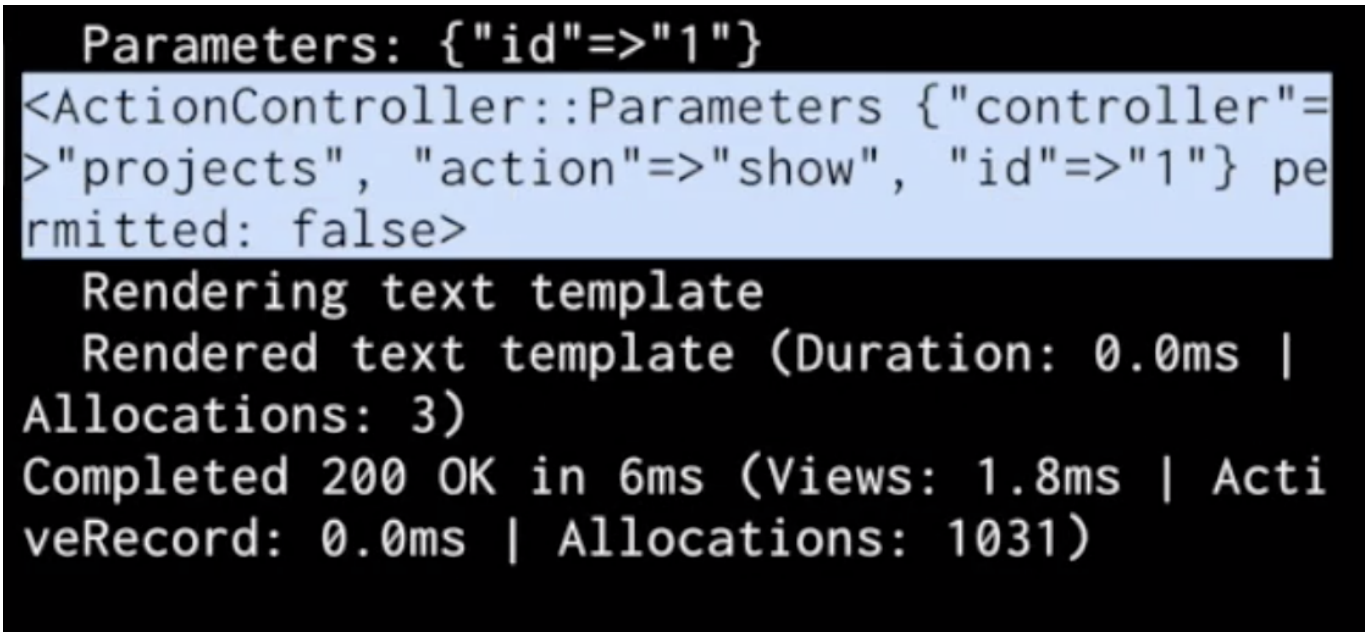
- We will be able to see that p statement in our terminal.
- Lets restart out rails server

```
rails s
```

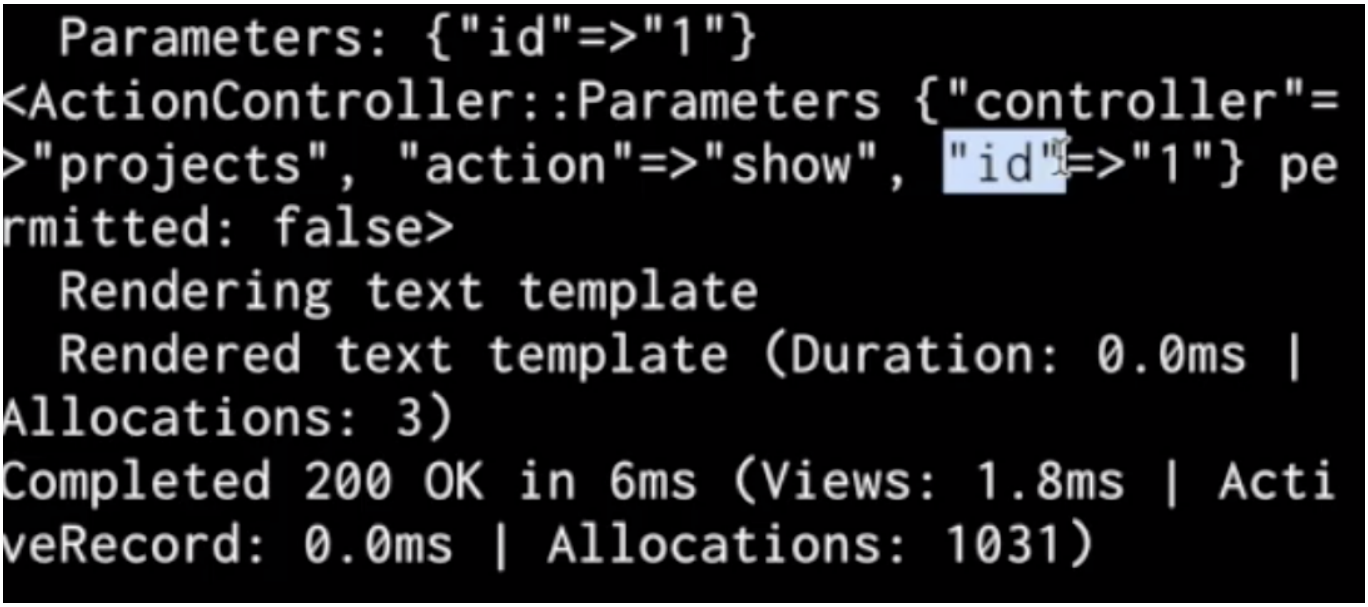
- Then we will go to the route:

```
localhost: 3000/projects/1
```

- This will still work
- Then what we can see is our print statement (see image below):



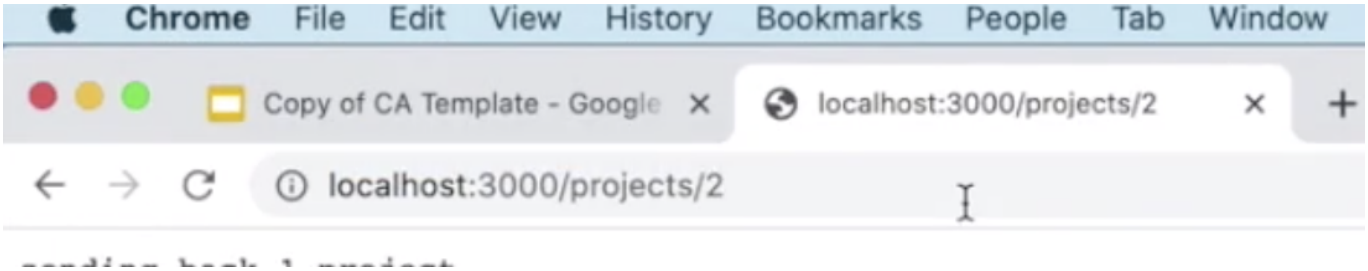
- We printed out params
    - params is a special object in which you can access everything in it using the hash notation.
- In this special params object, you can see that there is an id key:



- And it's value

```
  Parameters: {"id"=>"1"}
<ActionController::Parameters {"controller"=
>"projects", "action"=>"show", "id"=>"1"} pe
rmitted: false>
  Rendering text template
  Rendered text template (Duration: 0.0ms |
Allocations: 3)
Completed 200 OK in 6ms (Views: 1.8ms | Acti
veRecord: 0.0ms | Allocations: 1031)
```

- Through this params method we can access the placeholder or the number that we pass into the URL.
- Therefore is this case we pass in 2



- You will see that the id is the key and 2 is now the value.

```
Started GET "/projects/2" for ::1 at 2020-09
-11 10:02:29 +1000
Processing by ProjectsController#show as HTM
L
  Parameters: {"id"=>"2"}
<ActionController::Parameters {"controller"=
>"projects", "action"=>"show", "id"=>"2"} pe
rmitted: false>
  Rendering text template
  Rendered text template (Duration: 0.0ms |
Allocations: 2)
Completed 200 OK in 1ms (Views: 0.3ms | Acti
```

Accessing id value

- All we would need to do is use the hash notation (square brackets and name of the key):

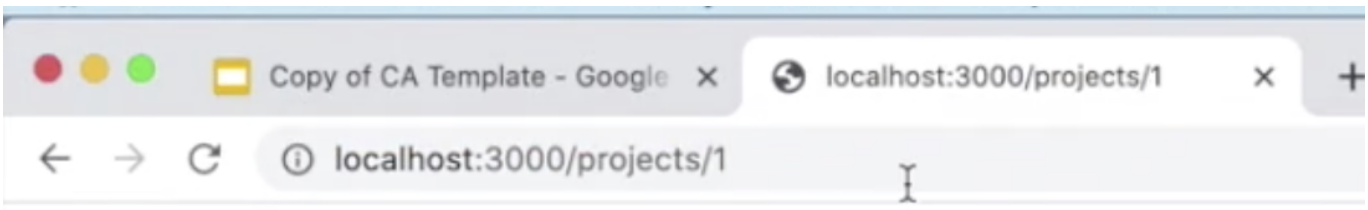
```
p params[:id]
```

- This will access the value
- Show if we save this file

routes.rb projects_controller.rb

```
24          name: "rails project",
25          github_status: false
26        },
27        {
28          id: 2,
29          name: "terminal app",
30          github_status: true
31        }
32      ]
33
34      p params[:id]
35
36      # projects.find do |project|
37      #   project[:id] ==
38      # end
39
40      render plain: "sending back 1 project"
41    end
42  end
```

- Then change our route back to 1



- now we are just accessing 1 in our terminal

```
Started GET "/projects/1" for ::1 at 2020-09
-11 10:03:23 +1000
   (0.1ms)  SELECT sqlite_version(*)
Processing by ProjectsController#show as HTM
L
  Parameters: {"id"=>"1"}
"1"
  Rendering text template
  Rendered text template (Duration: 0.0ms |
Allocations: 2)
Completed 200 OK in 2ms (Views: 1.5ms | Acti
```

Coming back to our find block

```
    p params[:id]

    # projects.find do |project|
    #   project[:id] ==
    # end

    render plain: "sending back 1 project"
  end
end
```

- We are finding a project based on its id in the hash.

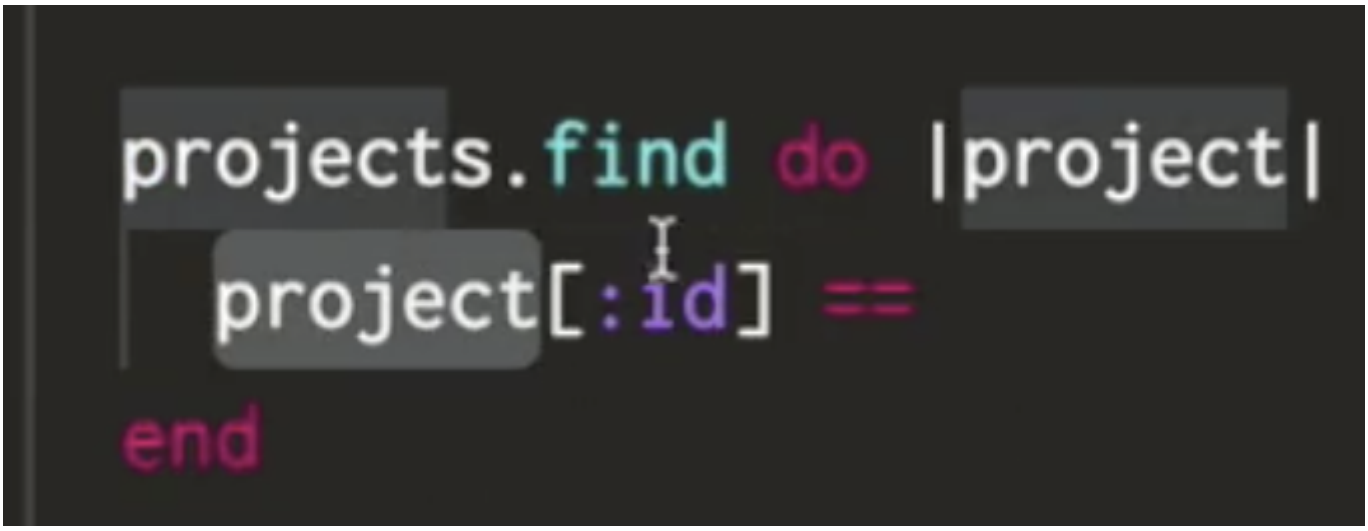
routes.rb projects_controller.rb

```
19  def show
20    # show is for sending back only 1 resource
21    projects = [
22      {
23        id: 1,
24        name: "rails project",
25        github_status: false
26      },
```

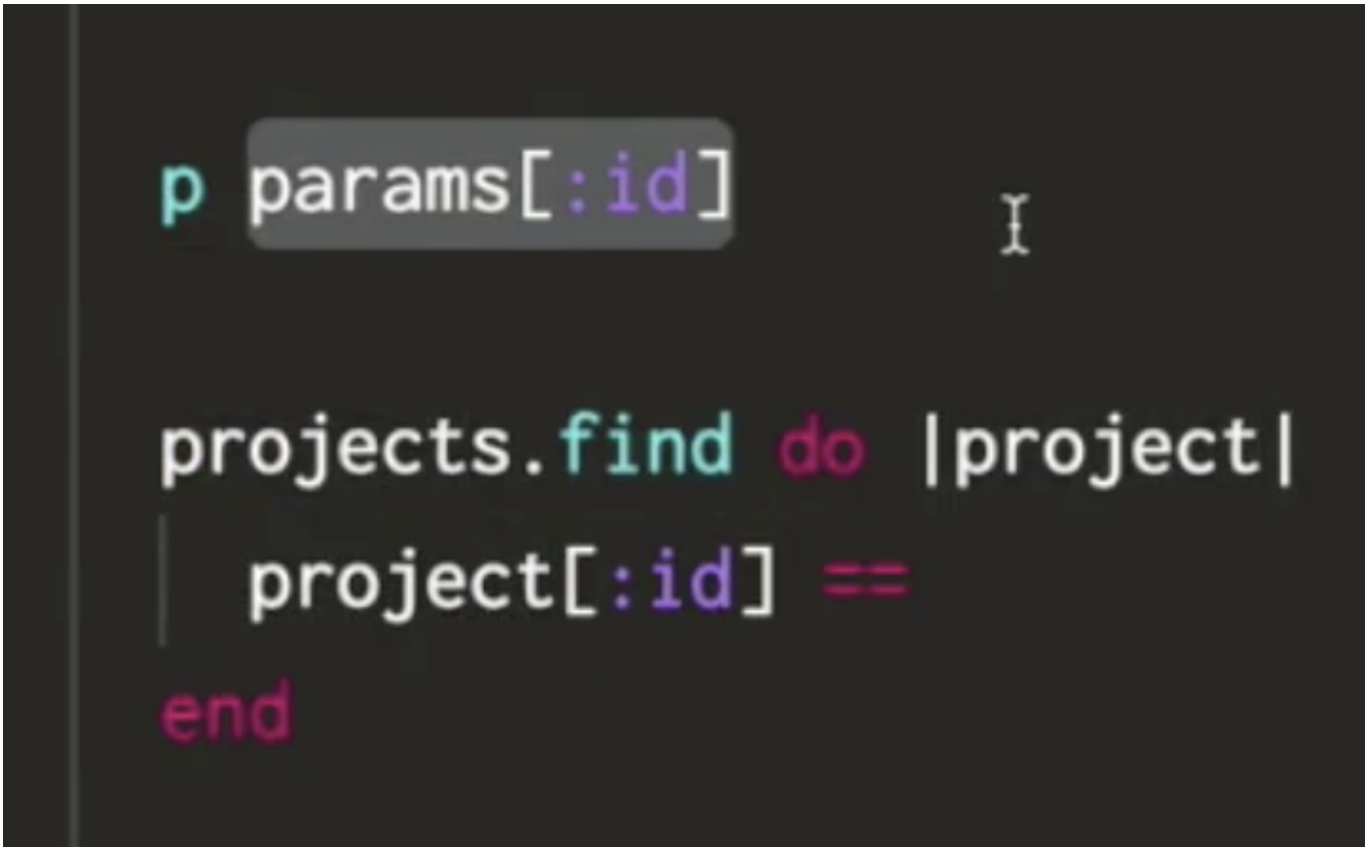
routes.rb projects_controller.rb

```
19  def show
20    # show is for sending back only 1 resource
21    projects = [
22      {
23        id: 1,
24        name: "rails project",
25        github_status: false
26      },
27      {
28        id: 2,
29        name: "terminal app",
30        github_status: true
31      }
32    ]
```

- Then we are going to find a particular project:



- Based on the param that we are working we:



- So we can copy this value and paste into our find (see image below):

routes.rb projects_controller.rb

```
25          github_status: false
26        },
27        {
28          id: 2,
29          name: "terminal app",
30          github_status: true
31        }
32      ]
33
34      projects.find do |project|
35        project[:id] == params[:id]
36      end
37
38      render plain: "sending back 1 project"
39    end
40  end
```

- We can then store the result of the find in a variable (found_project in this example):

```
found_project = projects.find do |project|
  project[:id] == params[:id]
end
```

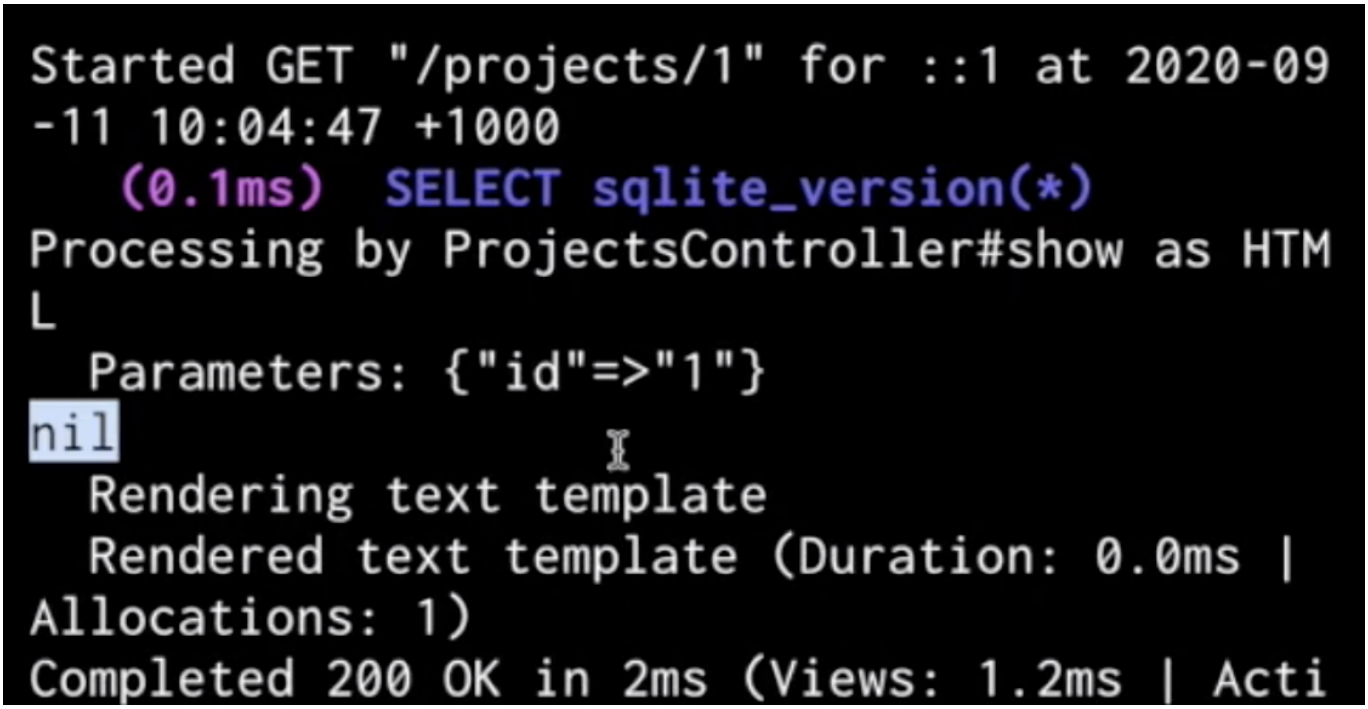
- Then we can print found_project

```
p found_project
```

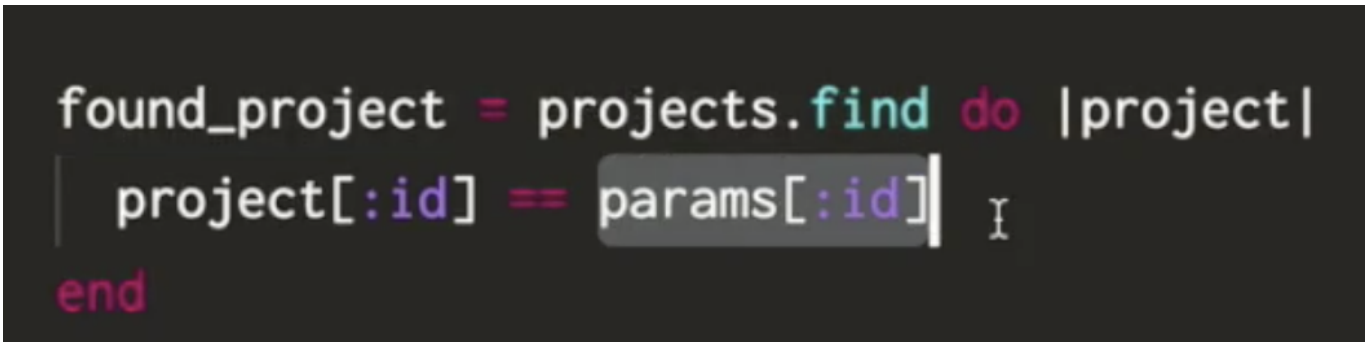
- Send the request again:

```
localhost:3000/projects/1
```

- And at the moment the return of our found_project is nil

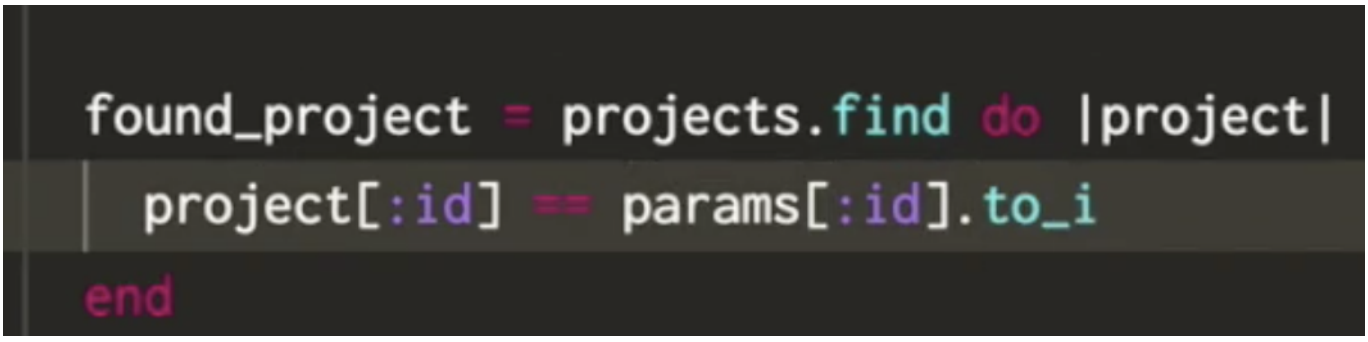


- The reason that it is nil is because:
- The param we are working with:

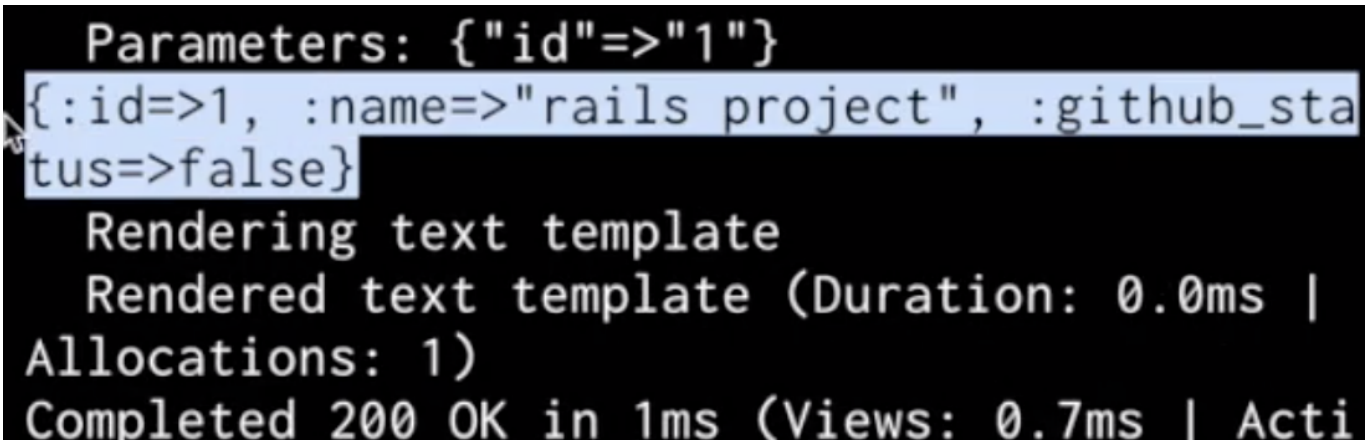


- This id that we are pulling out of params is actually a string.
- Where as the project[:id]

- that we are getting from the hash:
- is an integer.
- So we are comparing an integer to a string.
- In this case we have to do some explicit type casting and converting the string to an integer.



- Then we can save our file and refresh our browser
- Now we are pulling out one single hash.



- We can confirm this by sending back that hash to the client.

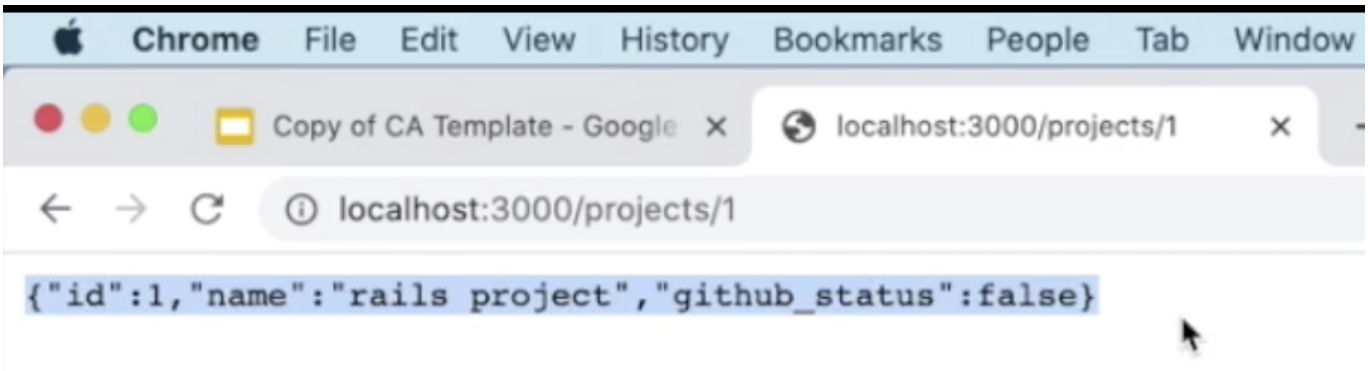
Sending back hash to client:

- Instead of plain text we can render JSON and send back the found_project to the client side.

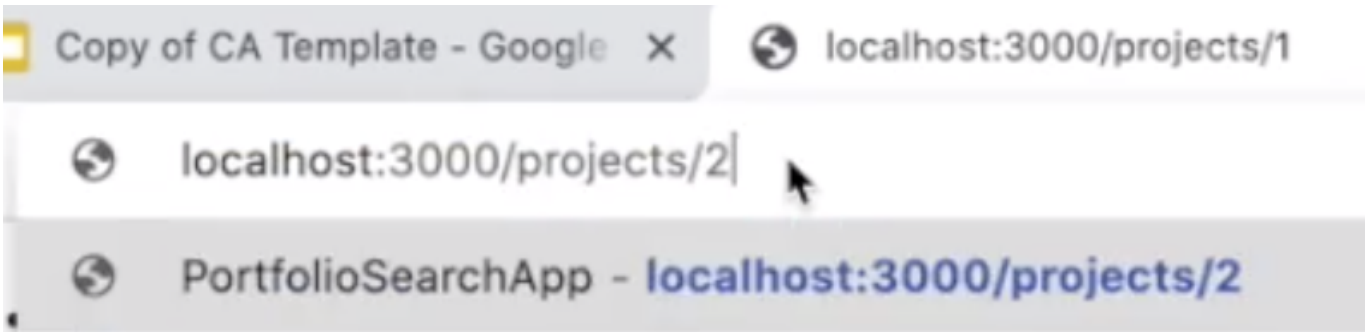
- Lets restart our server:

```
rails s
```

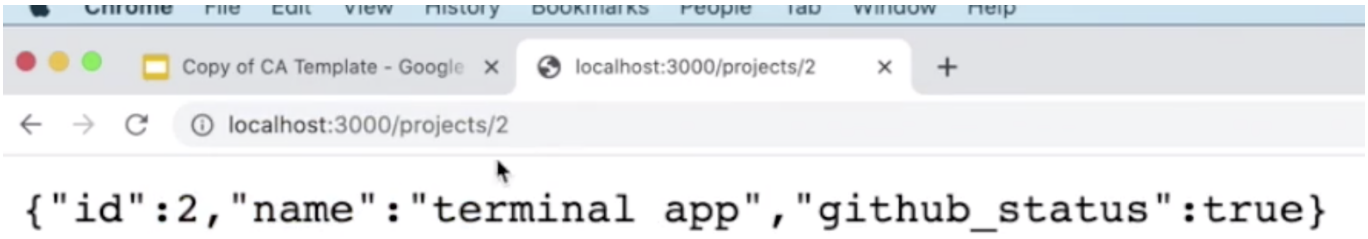
- refresh our browser



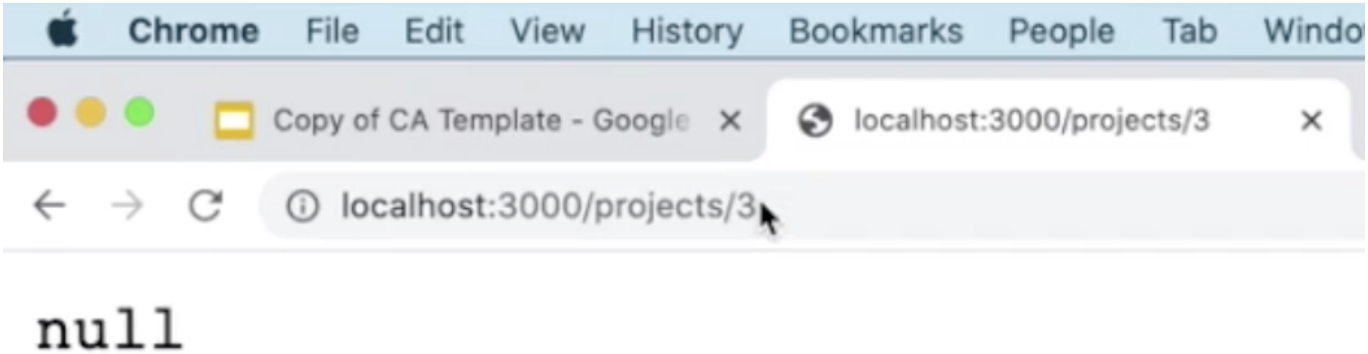
- And now we are getting just the first project of id 1.
- If we pass in 2 the browser:



- We will get the second project of id 2:



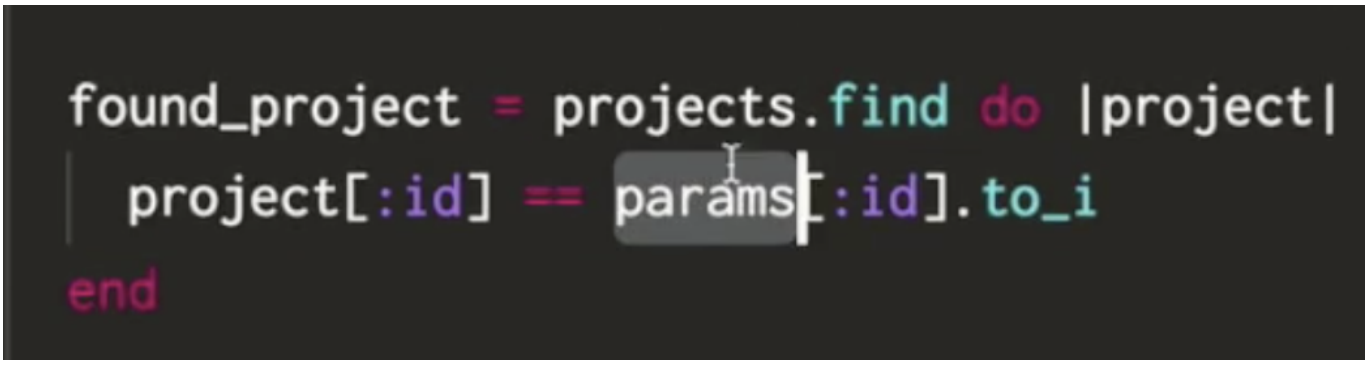
- If we pass in 3 we will get a return of null:



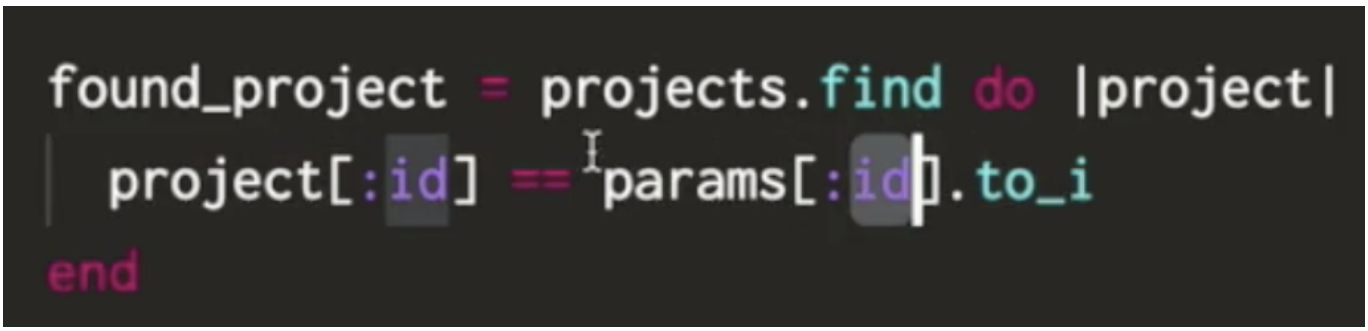
- This is because we don't have a project with an id of 3.
- We only have projects with an id of 1 and 2.


routes.rb
projects_controller.rb
18
19 def show
20 # show is for sending back only 1 resource
21 projects = [
22 {
23 id: 1,
24 name: "rails project",
25 github_status: false
26 },
27 {
28 id: 2,
29 name: "terminal app",
30 github_status: true
31 }
32]
33
34 found_project = projects.find do |project|
35 project[:id] == params[:id].to_i
36 end
37
38 render json: found_project

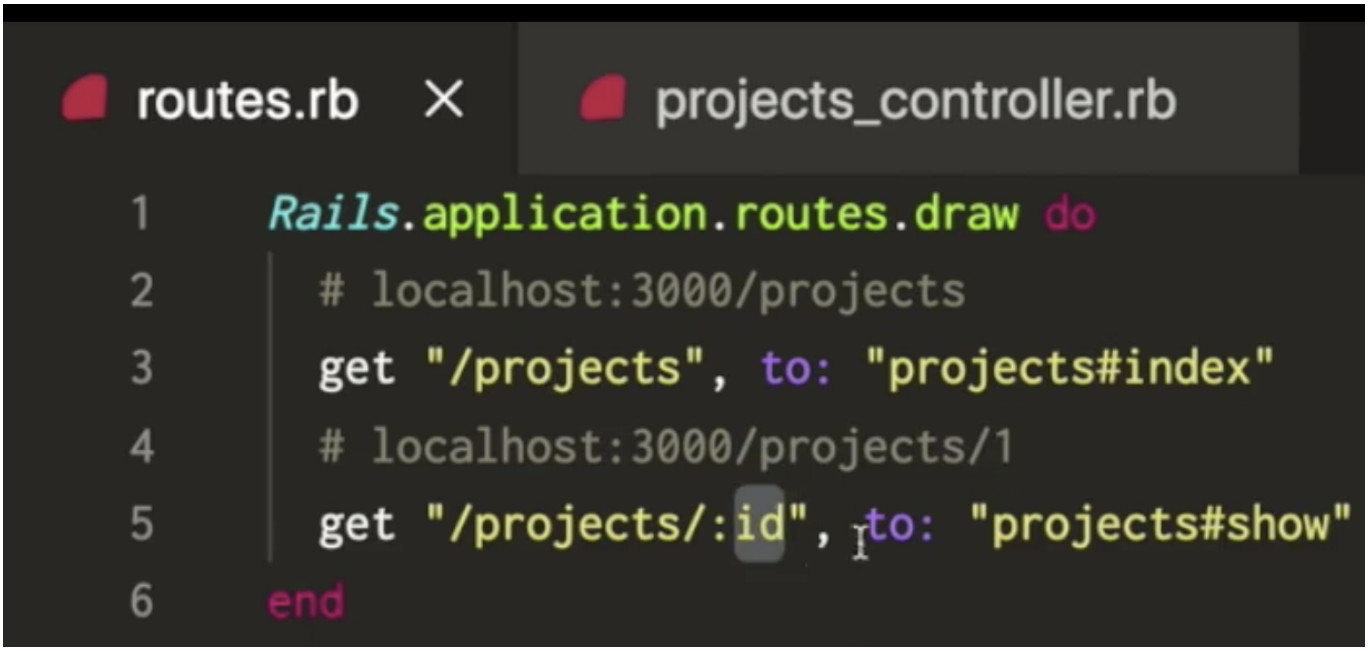
Params



- Params is a special method that we have access to in each controller action
- The id represents a key in our params.



- The name of this key depends on what we have named it in our routes.rb folder.



- Therefore the id in our routes folder is very important because:
    - Whatever we have in our params depends on whatever name we use in our route.

For example:

- We could name it elephant:

routes.rb × projects_controller.rb

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

- We then change the name in our projects_controller.rb folder:

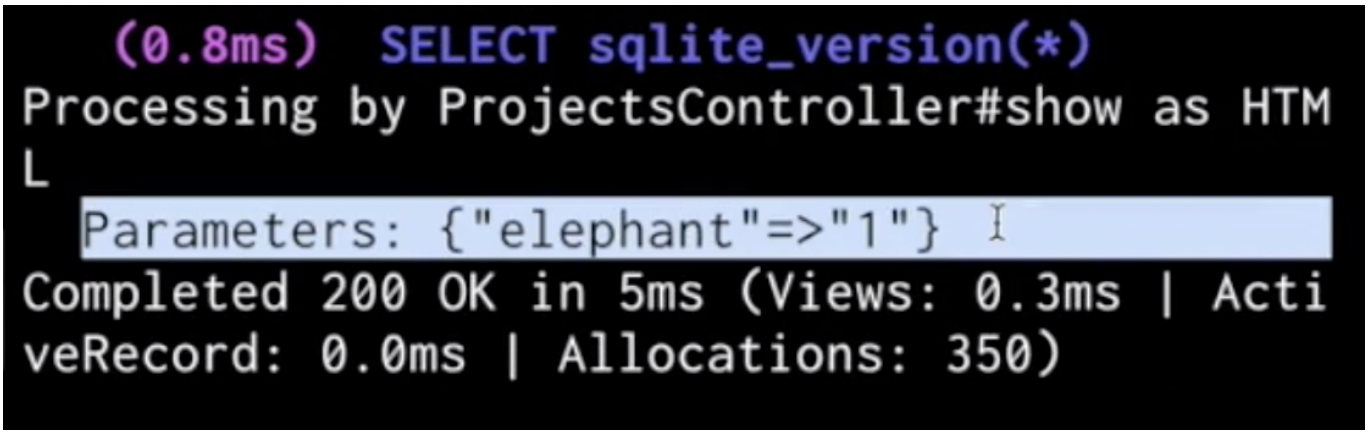
```
found_project = projects.find do |project|
  project[:id] == params[:elephant].to_i
end
```

- We can then run our rails server:

```
rails s
```

- The send a request again with a valid project id (1 in this case)
- And we can see that things still work.

- We can see this is updated in our terminal too:



- Therefore the key that we have access to in params is dependant on the name that we use in our routes.