

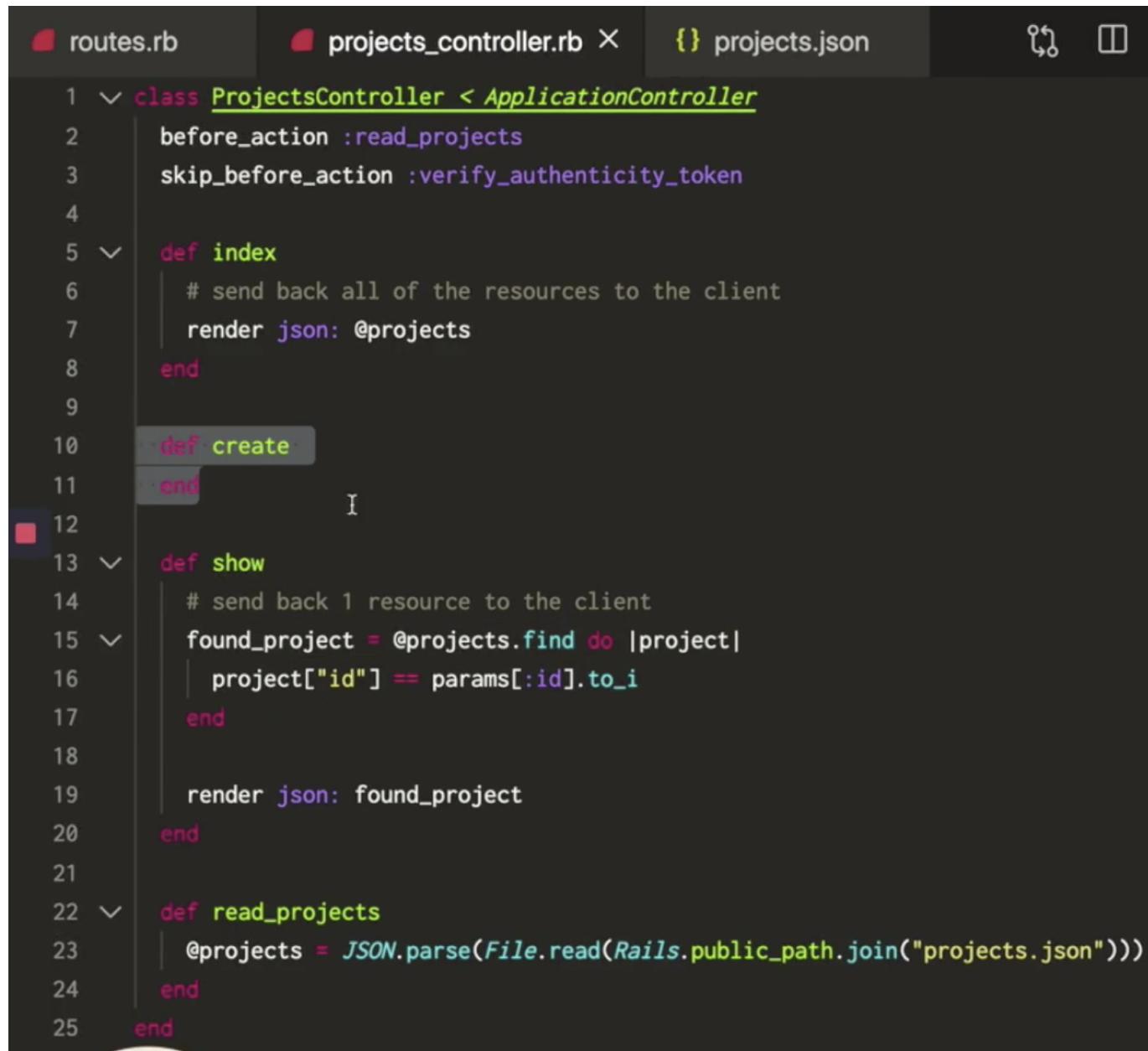
Rails Controllers: Query String

Link to lesson

- A link to the lesson can be found [here](#).

Create action

- Previously we set up a **Create action**



```
routes.rb          projects_controller.rb X      {} projects.json
1  class ProjectsController < ApplicationController
2    before_action :read_projects
3    skip_before_action :verify_authenticity_token
4
5    def index
6      # send back all of the resources to the client
7      render json: @projects
8    end
9
10   def create
11   end
12
13   def show
14     # send back 1 resource to the client
15     found_project = @projects.find do |project|
16       project["id"] == params[:id].to_i
17     end
18
19     render json: found_project
20   end
21
22   def read_projects
23     @projects = JSON.parse(File.read(Rails.public_path.join("projects.json")))
24   end
25 end
```

- And a **route** that **connected** to the **create action** through a **post request**.

```

routes.rb      X  projects_controller.rb  {} projects.json

1   Rails.application.routes.draw do
2     # localhost:3000/projects
3     get "/projects", to: "projects#index"
4     # localhost:3000/projects
5     post "/projects", to: "projects#create" | I
6     # localhost:3000/projects/1
7     get "/projects/:id", to: "projects#show"
8   end
9

```

- We found out we **couldn't test** this out in the **browser**, we had to test this out in **another app (postman)**:
- In order to send a **post request** to our Rails app.

Sending data from client side to web sever/rails app

- Often you would do this through a **form like a html forms**
- We are currently NOT working with forms because we don't yet have a **view**.
- What we DO have instead is the **actual URL**.
 - We can send some data through the URL
 - Like sending an **ID in our params** to show one project.
- Before we were sending through a param

localhost:3000/projects/1

We will send our data differently this time.

- Instead we will use the **query string** to send some data from our **client side** to our Rails app.

Query String

- Query string looks like this (see image below):

POST http://localhost:3000/projects



...

Untitled Request

POST



http://localhost:3000/projects

Params ▾

- The route is:

```
/projects
```

- After the route we can add a question mark

```
/projects?
```

- The question mark is the **start** of the **query string**.
- Then we can **define key value pairs** in our **query string**.
- For example: We can send across an id:

```
/projects?id
```

- And we want that id to be equal to 3.

```
/projects?id=3
```

- We can also send across another key value pair
 - Key value pairs need to be separated with an &

```
http://localhost:3000/projects?id=3&
```

- The second key value pair will be the name of the project

```
http://localhost:3000/projects?id=3&name=portfolio app
```

- Then we want to send through the GitHub status

```
http://localhost:3000/projects?id=3&name=portfolio app&github_status
```

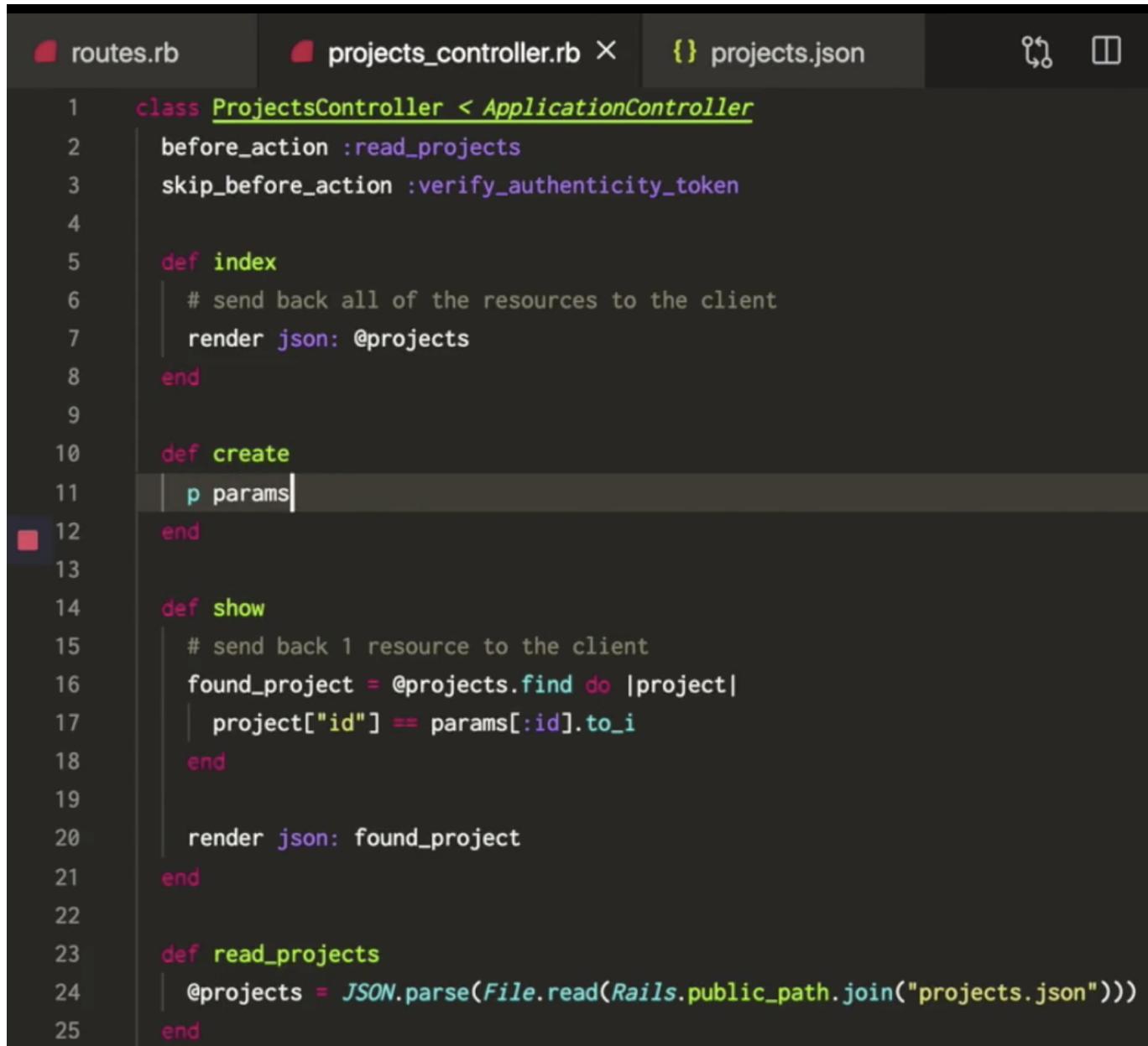
- And lets set that to true

```
http://localhost:3000/projects?id=3&name=portfolio  
app&github_status=true
```

Back to Create Action

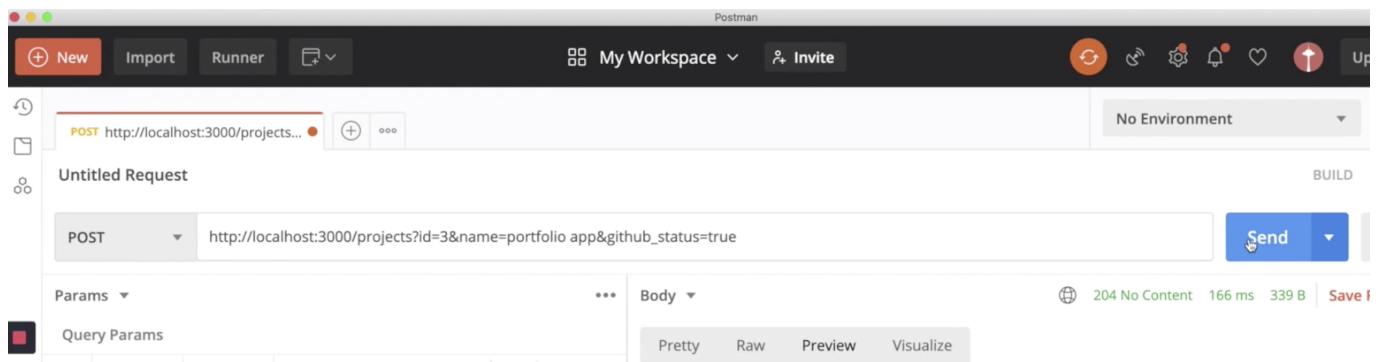
- Lets go back to our create action and *p params*:

```
def create  
  p params  
end
```



```
routes.rb          projects_controller.rb X      {} projects.json
1 class ProjectsController < ApplicationController
2   before_action :read_projects
3   skip_before_action :verify_authenticity_token
4
5   def index
6     # send back all of the resources to the client
7     render json: @projects
8   end
9
10  def create
11    p params
12  end
13
14  def show
15    # send back 1 resource to the client
16    found_project = @projects.find do |project|
17      project["id"] == params[:id].to_i
18    end
19
20    render json: found_project
21  end
22
23  def read_projects
24    @projects = JSON.parse(File.read(Rails.public_path.join("projects.json")))
25  end
```

Now we can send a post request on postman:



We will still get back a no content:

BUILD

The screenshot shows a browser developer tools interface with a network tab. A specific request is selected, displaying the following details:

- Icon: A globe icon.
- Status: 204 No Content
- Time: 94 ms
- Size: 339 B
- Save R... (button)

A tooltip for "204 No Content" is displayed below the status:

204 No Content

The server successfully processed the request, but is not returning any content.

But if we look at our logs and where we are printed our params, we will see that:

- We HAVE got some extra information in parameters now.

```
Started POST "/projects?id=3&name=portfolio%20app&github_status=true" for ::1 at 2020-09-11 12:36:21 +1000
(0.1ms)  SELECT sqlite_version(*)
Processing by ProjectsController#create as */
  Parameters: {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true"}
<ActionController::Parameters {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true", "controller"=>"projects", "action"=>"create"} permitted: false>
No template found for ProjectsController#create, rendering head :no_content
Completed 204 No Content in 4ms (ActiveRecord: 0.0ms | Allocations: 1835)
```

We now have:

- The id

```
status"=>"true"}
<ActionController::Parameters {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true", "controller"=>"projects", "action"=>"create"} permitted: false>
```

- The name key value pair

```
status"=>"true"}
<ActionController::Parameters {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true", "controller"=>"projects", "action"=>"create"} permitted: false>
```

- And we got the GitHub status

```
Processing by ProjectsController#create as */
Parameters: {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true"}
<ActionController::Parameters {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true", "controller"=>"projects", "action"=>"create"} permitted: false>
```

So ALL OF THE INFORMATION sent through on the query string:

```
http://localhost:3000/projects?id=3&name=portfolio app&github_status=true
```

- Can be accessed through params

```
def create
  p params
end
```

Accessing name in the project from params

```
def create
  p params[:name]
end
```

- To double check that works we can send a post request through postman:

The screenshot shows the Postman interface with an 'Untitled Request' named 'Untitled Request'. The method is set to 'POST' and the URL is 'http://localhost:3000/projects?id=3&name=portfolio app&github_status=true'. The 'Params' tab is selected. The response status is 204 No Content, with a duration of 166 ms and a body size of 339 B.

Then take a look at our log and we can see that we are getting the name of the project (portfolio app)

```
Started POST "/projects?id=3&name=portfolio%20app&github_status=true" for ::1 at 2020-09-11 12:37:10 +1000
  (0.1ms)  SELECT sqlite_version(*)
Processing by ProjectsController#create as */
  Parameters: {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true"}
  "portfolio app"
No template found for ProjectsController#create, rendering head :no_content
Completed 204 No Content in 1ms ( ActiveRecord: 0.0ms | Allo
```

Collect information from Param and store in a variable

```
def create
  p params[:name]
end
```

```
def create
  new_project =
end
```

And we are going to pass a hash with the id that we get from params:

```
def create
  new_project = {id: params[:id] }
end
```

And we are also going to get params name of the project

```
def create
  new_project = {id: params[:id], name: params[:name] }
end
```

And we are also going to get our GitHub status

```
def create
  new_project = {id: params[:id], name: params[:name], github_status:
params[:githubstatus] }
end
```

```
routes.rb      projects_controller.rb X  projects.json
1 class ProjectsController < ApplicationController
2   before_action :read_projects
3   skip_before_action :verify_authenticity_token
4
5   def index
6     # send back all of the resources to the client
7     render json: @projects
8   end
9
10  def create
11    new_project = { id: params[:id], name: params[:name], github_status: params[:githubstatus] }
12  end
13
14  def show
15    # send back 1 resource to the client
16    found_project = @projects.find do |project|
17      project["id"] == params[:id].to_i
18    end
19
20    render json: found_project
21  end
22
23  def read_projects
24    @projects = JSON.parse(File.read(Rails.public_path.join("projects.json")))
25  end
```

NOW LETS PRINT NEW_PROJECT

```
def create
  new_project = {id: params[:id], name: params[:name], github_status:
  params[:githubstatus]}
  p new_project
end
```

THEN SEND ANOTHER POST REQUEST

- With the same information in the query string.

```
Processing by ProjectsController#create as */
Parameters: {"id"=>"3", "name"=>"portfolio app", "github_status"=>"true"}
{> id=>"3", :name=>"portfolio app", :github_status=>"true"}
No template found for ProjectsController#create, rendering head
:no_content
Completed 204 No Content in 1ms (ActiveRecord: 0.0ms | Allocati
```

- We will see in our logs we have got:
 - A hash with ALL of the values we have sent through as parameters:

Now that we have this nicely formatted hash we can insert it into our projects array of hashes

```
def index
  # send back all of the
  render json: @projects
end
```

- We have access to the @projects instance variable in our create method because....

```
def create
  new_project = {id: params[:id], name: params[:name],
github_status: params[:githubstatus]}
  p new_project
end
```

THE BEFORE ACTION IS RUNNING before our create method!

```
before_action :read_projects
```

The screenshot shows a code editor with three tabs: `routes.rb`, `projects_controller.rb`, and `projects.json`. The `projects_controller.rb` tab is active, displaying the following code:

```
1 class ProjectsController < ApplicationController
2   before_action :read_projects
3   skip_before_action :verify_authenticity_token
4
5   def index
6     # send back all of the resources to the client
7     render json: @projects
8   end
```

The `before_action` line is highlighted in purple, indicating it is being executed before the `index` action.

And it is reading in the JSON file:

```

5   def index
6     # send back all of the resources to the client
7     render json: @projects
8   end
9
10  def create
11    new_project = { id: params[:id], name: params[:name], github_status: params[:github_status] }
12    p new_project
13  end
14
15  def show
16    # send back 1 resource to the client
17    found_project = @projects.find do |project|
18      project["id"] == params[:id].to_i
19    end
20
21    render json: found_project
22  end
23
24  def read_projects
25    @projects = JSON.parse(File.read(Rails.public_path.join("projects.json")))
26  end
27 end

```

Pushing new project into our array of hashes:

- In our **create** method:

```

def create
  new_project = {id: params[:id], name: params[:name],
github_status: params[:githubstatus]}
  p new_project
end

```

- We can get rid of `p new_project` and push `new_project` into our array of hashes (JSON)

```
def create new_project = {id: params[:id], name: params[:name], github_status:
params[:githubstatus]} @projects << new_project end
```

- Now we can print `@projects`

```

def create
  new_project = {id: params[:id], name: params[:name],
github_status: params[:githubstatus]}
  @projects << new_project

```

```
p projects
end
```

- Lets restart our rails server so we can actually see what's happening:

```
rails s
```

- Send a post request

The screenshot shows the Postman application interface. A new request is being created for a POST method to the URL `http://localhost:3000/projects`. The request body contains the following parameters: `id=3`, `name=portfolio app`, and `github_status=true`. The response status is 204 No Content with a duration of 166 ms and a size of 339 B.

- In our terminal response we can now see we have 3 hashes
- including our newly created hash

```
Processing by ProjectsController#create as */*
Parameters: {"id=>"3", "name=>"portfolio app", "github_status=>"true"}
[{"id=>1, "name=>"rails project", "github_status=>false"}, {"id=>2, "name=>"terminal app", "github_status=>true"}, {:id=>"3", :name=>"portfolio app", :github_status=>"true"}]
```

Write the newly created array to JSON file

- In order to write this new array into our JSON file we are going to define a new method

Note: Our array is @projects

```
def write_projects(projects)
end
```

- The method will take projects as an argument (which is our newly created array)

- The write method is going to be similar to the read JSON method.
- Instead of File.read we are going to use File.write

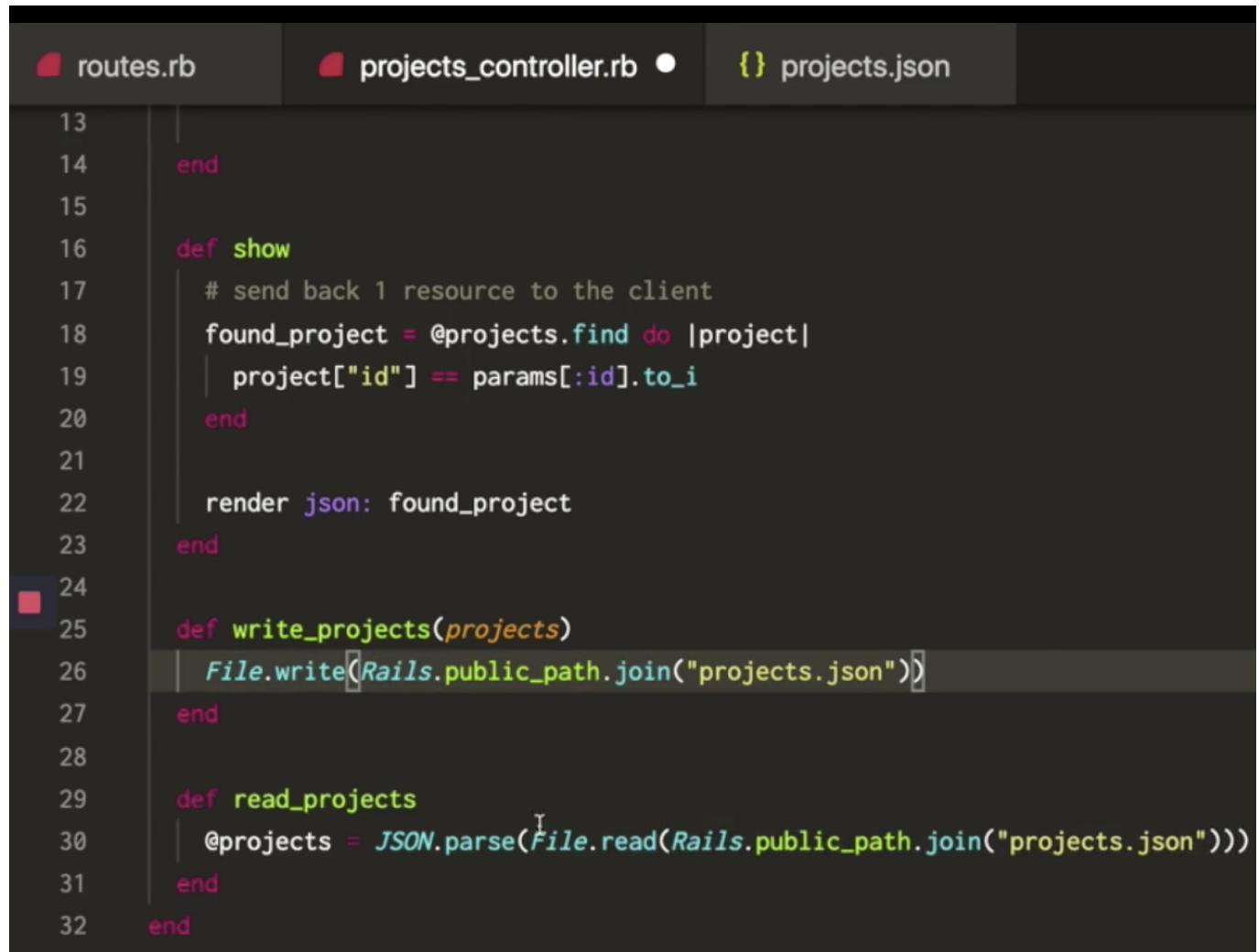
```
File.write
```

- And write takes TWO arguments.

ARGUMENT ONE:

- It takes the file path (which is the same as what we had with File.read)
- Using the public_path.join notation

```
File.write(Rails.public_path.join("projects.json"))
```



```
routes.rb          projects_controller.rb ●          projects.json
13
14      end
15
16      def show
17          # send back 1 resource to the client
18          found_project = @projects.find do |project|
19              project["id"] == params[:id].to_i
20          end
21
22          render json: found_project
23      end
24
25      def write_projects(projects)
26          File.write(Rails.public_path.join("projects.json"))
27      end
28
29      def read_projects
30          @projects = JSON.parse(File.read(Rails.public_path.join("projects.json")))
31      end
32  end
```

ARGUMENT TWO:

- Which is whatever you want to actually write to (that particular file - this in case it is projects).

```
File.write(Rails.public_path.join("projects.json")), projects)
```

The screenshot shows a code editor with three tabs: `routes.rb`, `projects_controller.rb` (which is the active tab), and `projects.json`. The `projects_controller.rb` file contains the following code:

```
13
14     end
15
16     def show
17         # send back 1 resource to the client
18         found_project = @projects.find do |project|
19             project["id"] == params[:id].to_i
20         end
21
22         render json: found_project
23     end
24
25     def write_projects(projects)
26         File.write(Rails.public_path.join("projects.json"), projects)
27     end
28
29     def read_projects
30         @projects = JSON.parse(File.read(Rails.public_path.join("projects.json")))
31     end
32 end
```

- NOTE `projects` is an array of hashes

Calling the `write_projects` method

- Let's call our `write_projects` method inside of our `create` method.

```
def create
    new_project = {id: params[:id], name: params[:name],
github_status: params[:githubstatus]}
    @projects << new_project
    write_projects(@project)
end
```

```

routes.rb
projects_controller.rb
{} projects.json

2 before_action :read_projects
3 skip_before_action :verify_authenticity_token
4
5 def index
6   # send back all of the resources to the client
7   render json: @projects
8 end
9
10 def create
11   new_project = { id: params[:id], name: params[:name], github_status: params[:github_status] }
12   @projects << new_project
13   write_projects(@projects)
14 end
15
16 def show
17   # send back 1 resource to the client
18   found_project = @projects.find do |project|
19     project["id"] == params[:id].to_i
20   end
21
22   render json: found_project
23 end
24
25 def write_projects(projects)
26   File.write(Rails.public_path.join("projects.json"), projects)

```

So in our create method we are: pushing the new project into our array of hashes:

```
@projects << new_project
```

- Then we are parsing that particular array of hashes (@projects):
- To our write_projects method as an argument

```
write_projects(@projects)
```

One more thing to do to @projects

- Currently @projects is just an array of hashes.
- But that won't work with JSON format.

Formatting @projects

- We need to call the json.generate method in our write projects method

```
def write_projects(projects)
  File.write(Rails.public_path.join("projects.json")),
  JSON.generate(projects)
end
```

```
def create
  new_project = { id: params[:id], name: params[:name], github_status: params[:github_status] }
  @projects << new_project
  write_projects(@projects)
end
```

Render plain text

- Lets go to our create method and render some plain text

```
def create
  new_project = {id: params[:id], name: params[:name],
  github_status: params[:githubstatus]}
  @projects << new_project
  write_project(@project)
  render plain "successfully added to projects!"
end
```

```
def create
  new_project = { id: params[:id], name: params[:name], github_status: params[:github_status] }
  @projects << new_project
  write_projects(@projects)
  render plain: "successfully added to projects!"
end
```

Run this again!

- Send our post request and will receive a response.

Postman interface showing a successful POST request to `http://localhost:3000/projects?id=3&name=portfolio app&github_status=true`. The response status is 200 OK, 98 ms, 502 B, and the response body is "successfully added to projects!".

Double checking it added to our json file

- lets go to our projects.json file
- We can see that it our projects.json has been written to and received this object/this hash with:
 - The id
 - The name
 - The GitHub status

```
[{"id":1,"name":"rails project","github_status":false},{ "id":2,"name":"terminal app","github_status":true},{ "id":3,"name":"portfolio app","github_status":true}]
```

- If we go back to the browser and refresh
- We are now getting the project we just created:

```
[{"id":1,"name":"rails project","github_status":false},{ "id":2,"name":"terminal app","github_status":true},{ "id":3,"name":"portfolio app","github_status":true}]
```

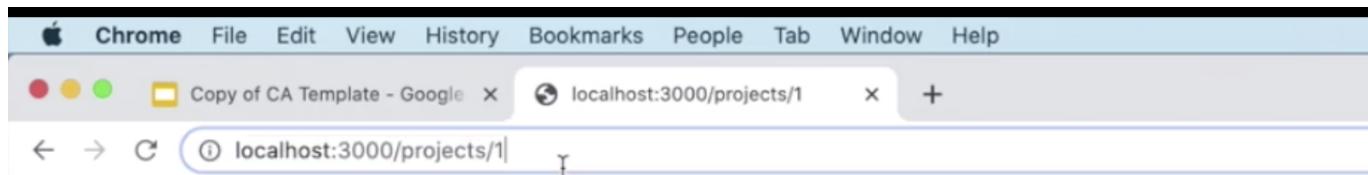
- Because it is a part of the json file, it is now included in this response.

Individual projects

- We can find projects 1

localhost:3000/projects/1

- And we will get the first project



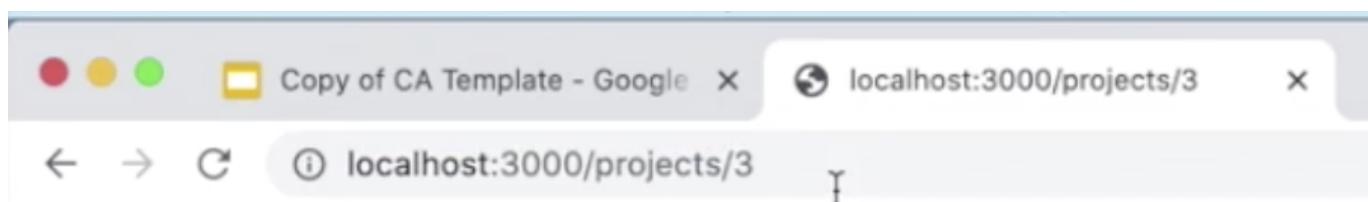
```
{"id":1, "name": "rails project", "github_status": false}
```

- We can do the same for the second project

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

- BUT we get null when we try the third project

```
localhost:3000/projects/3
```



null

Why null?

- The problem is we are writing the id to our json file as a string.

```
[{"id":1, "name": "rails project", "github_status": false}, {"id":2, "name": "terminal app", "github_status": true}, {"id": "3", "name": "portfolio app", "github_status": "true"}]
```

- To overcome this, let's first manually delete this project from our json file.

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

Fixing null?

- And lets go back to our controller (projects_controller.rb)

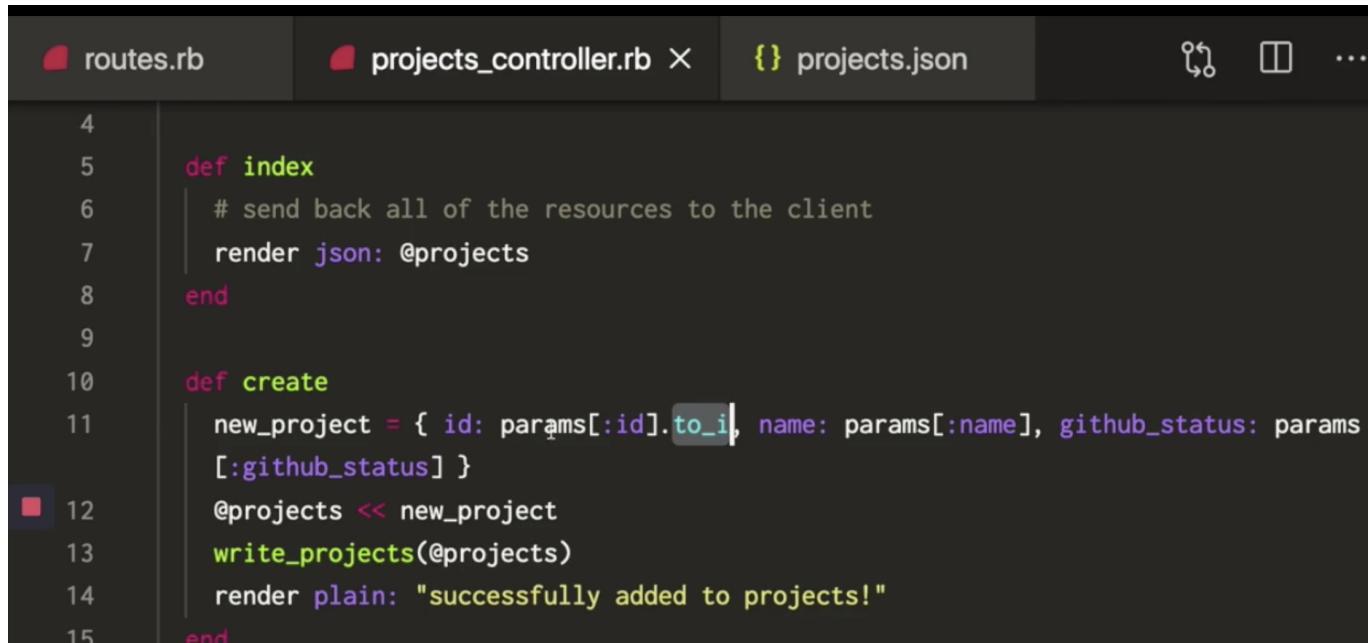
```
routes.rb
projects_controller.rb ×
{} projects.json

1 render json: @projects
2 end
3
4 def create
5   new_project = { id: params[:id], name: params[:name], github_status: params
6     [:github_status] }
7   @projects << new_project
8   write_projects(@projects)
9   render plain: "successfully added to projects!"
10  end
11
12
13  def show
14    # send back 1 resource to the client
15    found_project = @projects.find do |project|
16      project["id"] == params[:id].to_i
17    end
18
19    render json: found_project
20
21  end
22
23
24  def write_projects(projects)
25
26
```

What we are going to have to do is:

- Convert the id that is passed through params to an integer

```
def create
  new_project = {id: params[:id].to_i, name: params[:name],
github_status: params[:githubstatus]}
  @projects << new_project
  write_project(@project)
end
```



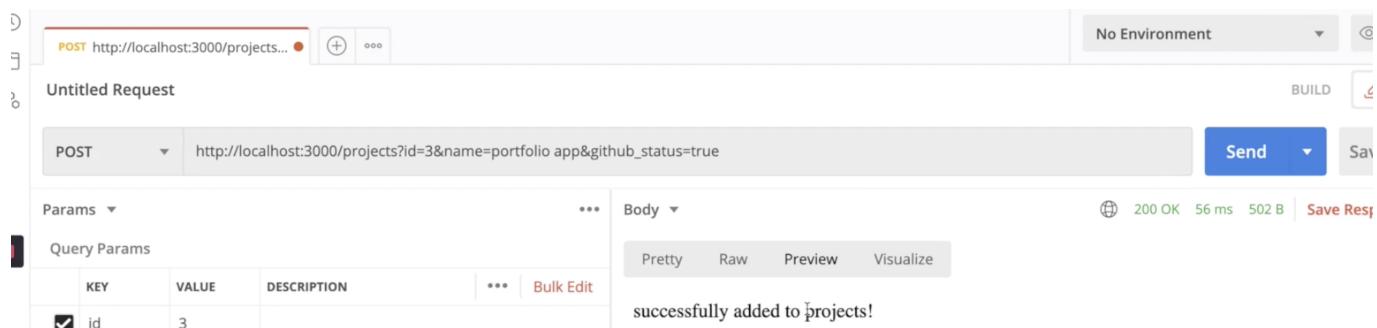
```

4
5      def index
6          # send back all of the resources to the client
7          render json: @projects
8      end
9
10     def create
11         new_project = { id: params[:id].to_i, name: params[:name], github_status: params[:github_status] }
12         @projects << new_project
13         write_projects(@projects)
14         render plain: "successfully added to projects!"
15     end

```

Save and send post request again to check for reading project 3

- Save project and send a post request
- Will see we have a response "successfully added to projects!"



Untitled Request

POST http://localhost:3000/projects?&id=3&&name=portfolio app&&github_status=true

Params

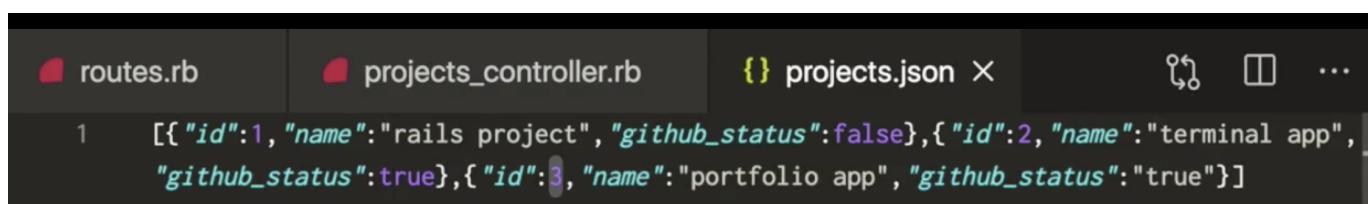
KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	3	

Body

Pretty Raw Preview Visualize

successfully added to projects!

- Now lets look at the json file
- And id is now an integer



```

1      [{"id":1,"name":"rails project","github_status":false},{ "id":2,"name":"terminal app","github_status":true},{ "id":3,"name":"portfolio app","github_status":true}]

```

- Then we can come back to the browser and check if we get project 3

localhost:3000/projects/3

- And you can now access project 3



A screenshot of a Google Chrome browser window. The title bar shows "Chrome" and the menu items "File", "Edit", "View", "History", "Bookmarks", "People", "Tab", "Window", and "Help". There are two tabs open: "Copy of CA Template - Google" and "localhost:3000/projects/3". The address bar shows "localhost:3000/projects/3". The main content area displays the following JSON object:

```
{"id":3,"name":"portfolio app","github_status":"true"}
```