Module 3
HTTP, RESTful APIs

CS W169A: Software Engineering

John Yang | Summer 2020

1 Github Routes

Read through and try to answer the following questions. Work in pairs if you can!

1.1 Requests

Without looking at any documentation, discuss what the following HTTP requests might do (no specific answer format required, just a general idea). *Note*: These links do not work, but determine what the request type and arguments would do if they did exist.

- GET https://github.com/orgs/cs169/repos
  Retrieves all the repositories for the organization "cs169"
- GET https://github.com/repos/cs169/homework2
  Retrieves repository called "homework2" for the organization "cs169". (Food for thought: How should this API respond if the repository doesn't exist?)
- POST https://github.com/orgs/cs169/repos
  Publish/create a repository for organization "cs169". (Food for thought: If this repository is to be created, what other information should be sent? How might we send this information with the post request? If we use JSON, how could we format the necessary information? No right answers, but important questions to ask!)

1.2 API Documentation

Read through GitHub's API documentation, which can be found at https://developer.github.com/v3. Find answers to the following questions.

- How can I get publicly available information about a user?
  Github Documentation: https://developer.github.com/v3/users/get-a-single-user
  Example Request: curl https://api.github.com/users/cs169
  Behavior: Should grab the information of the user with the name "cs169".
- How can I list all pull requests of a repository?
  Github Documentation: https://developer.github.com/v3/pulls/list-pull-requests
  Example Request: curl https://api.github.com/repose/rails/rails/pulls
  Behavior: Should list all PRs of the "rails" repo owned by the "rails" organization/account
- How can I get all closed pull requests of a repository?
  Github Documentation: https://developer.github.com/v3/pulls/list-pull-requests
  Example Request: curl https://api.github.com/repos/rails/rails/pulls?state=closed
  Behavior: Should list all closed PRs of "rails" repo owned by the "rails" organization/account.
- How can I create a new pull request?
  Github Documentation: https://developer.github.com/v3/pulls/create-a-pull-request

Example Request: This request is different from the previous ones in that it is a POST, not GET request (since we're trying to create something). Therefore, we must modify the curl request by 1. Changing the request type to POST and 2. Sending the needed information as arguments in the URL or as a JSON (depends on the API). In this case, it might look like:
curl -X POST https://api.github.com/repos/:owner/:repo/pulls

- Which input values are required to create a new pull request?
  From the previous answer's Github Documentation hyperlink, it looks like the required parameters are: title, head, and base

2 API Implementation

Finish the following ruby implementation for a simple todo API. Assume you have access to a params hash with any necessary query parameters, along with a format_as_json(object) function. Return results as JSON when applicable.

```ruby
class User
  attr_reader :id
  attr_accessor :todo

  def initialize
    @id = app.get_new_user_id
    @todo = Todo.new
  end
end

class Todo
  def initialize
    @items = Array.new
  end

  def add item
    if not @items.include? item
      @items << item
    end
  end

  def delete item
    @items.delete item
  end

  def view_item item_id
    # assume each item instance has an accessible id field,
    # and all items/ids are unique
    item = @item.find {|i| i.id = item_id } # SOLUTION
    format_as_json item unless item.nil?
  end
end
```

```ruby
# Route Handling (Sinatra)

get '/user/:id/todo' do
  user = get_user_by_id params[:id]
  user.todo.to_json # SOLUTION
end

get '/user/:id/todo/:item_id' do
  user = get_user_by_id params[:id]
  todo = user.todo # SOLUTION
  todo.view_item params[:item_id] # SOLUTION
end

post '/user/:id/todo' do
  user = get_user_by_id params[:id]
  request.body.rewind
  raw_item = JSON.parse request.body.read
  user.todo.add raw_item # SOLUTION
  201 # SOLUTION (Return 201 to indicate resource created)
end

delete '/user/:id/todo/:item' do
  user = get_user_by_id params[:id]
  user.todo.delete params[:item] # SOLUTION
  200 # SOLUTION (Return 200 to indicate resource deleted)
end
```