Faculty of Computers, Informatics and Microelectronics

Technical University of Moldova

PSI

Laboratory work # 4

Code Documentation

Author:
Petru Negrei

Supervisor:
A. Railean

November 2014

1 Introduction

1.1 Objective

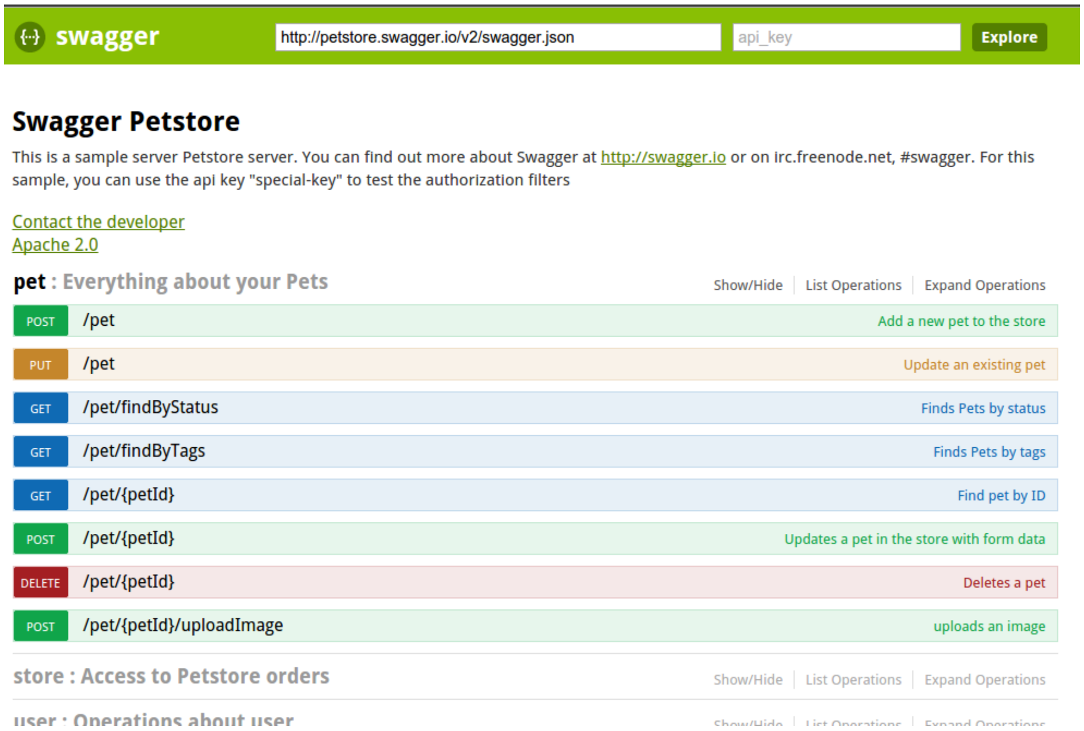
- Analyze and study available tools for code generation.
- Find advantages and disadvantages of using a certain tool.

2 Tools

2.1 API documentation

In this laboratory work I choose the Swagger API tool for generating the api documentation.

Swagger is a simple yet powerful representation of your RESTful API. With a Swagger-enabled API, you get interactive documentation, client SDK generation and discoverability.



Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key "special-key" to test the authorization filters

Contact the developer
Apache 2.0

pet : Everything about your Pets — Show/Hide | List Operations | Expand Operations

POST	/pet	Add a new pet to the store
PUT	/pet	Update an existing pet
GET	/pet/findByStatus	Finds Pets by status
GET	/pet/findByTags	Finds Pets by tags
GET	/pet/{petId}	Find pet by ID
POST	/pet/{petId}	Updates a pet in the store with form data
DELETE	/pet/{petId}	Deletes a pet
POST	/pet/{petId}/uploadImage	uploads an image

store : Access to Petstore orders — Show/Hide | List Operations | Expand Operations

user : Operations about user — Show/Hide | List Operations | Expand Operations

Minimum Configuration

```
Swagger::Docs::Config.base_api_controller = ActionController::API
Swagger::Docs::Config.register_apis({
 "1.0" => {
    controller_base_path: "",
    api_file_path: 'public',
    base_path: "http://localhost:3000"
 }
})
```

Example of documentation

```
# ...
class AppointmentsController < ApplicationController

  swagger_controller :appointments, 'Dates'

  # GET /dates
```

```
    swagger_api :index do
      summary 'Returns all dates'
      response :unauthorized
    end

    # GET /dates/1
    swagger_api :show do
      summary "Fetches a single date"
      param :path, :id, :integer, :required, "Date Id"
      response :unauthorized
      response :not_found
    end

    end
  end
```

To generate run:

```
  $ rake swagger:docs
```

3 Yard

YARD is a documentation generation tool for the Ruby programming language. It enables the user to generate consistent, usable documentation that can be exported to a number of formats very easily, and also supports extending for custom Ruby constructs such as custom class level definitions. Above is a highlight of the some of YARD's notable features. Update the rake and guard file.

```
   $ vim Gemfile
   gem "yard"
   $ bundle
   $ yardoc # to generate new documentation
   $ vim Rakefile
   require 'rdoc/task'
   ...
   YARD::Rake::YardocTask.new
   ...
   $ vim Guardfile

   guard rake, :task => 'yard' do
     watch(%r{^lib/(.+)\.rb)
     watch(%r{^(markdown|md|rdoc|)})
    end
```

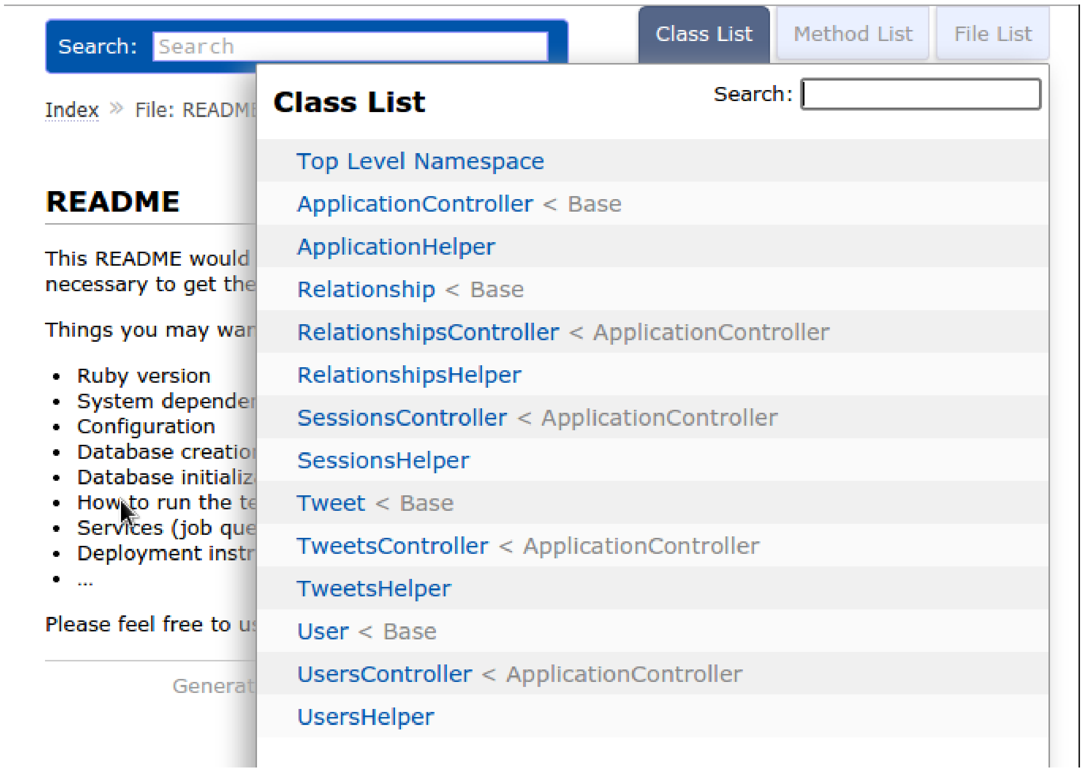
Examples of markup language used for creating documentation.

```
  # @author Yakuda Katz
  # @return [String] description
  # @param name [Conway::Cell] description
  # @param name [#method] description
  ...
  # @see Conway::World
  # @see Conway::World#method
  # @see #method
  ...
  # @example descrprion
  #  Cell.new(state: :dead),alive? #false
  ...
  # @yield [ cell ] cell description
  # @yieldparam cell [Conway::Cell] description
  # @note description
  ...
  # @!group Name
   ...
  # @!endgroup
```

```
# only first word count rest is show as descripiton
# ... {Word#method descripiton} ...
# {link text}
# {include:file:config/example.rdoc}
# {include:Conway::Utils::Serializer} # show the comment
```

In order to generate the documentation you need to run the following commands. The first one will also show how much your application is covered by documentation. The second one wiill start a server at port 8088 so you can see generated files.

```
$ yard
$ yard server
```



README

This README would
necessary to get the

Things you may war

- Ruby version
- System depende
- Configuration
- Database creatio
- Database initializ
- How to run the te
- Services (job que
- Deployment instr
- ...

Please feel free to u

Generat

Class List

- Top Level Namespace
- ApplicationController < Base
- ApplicationHelper
- Relationship < Base
- RelationshipsController < ApplicationController
- RelationshipsHelper
- SessionsController < ApplicationController
- SessionsHelper
- Tweet < Base
- TweetsController < ApplicationController
- TweetsHelper
- User < Base
- UsersController < ApplicationController
- UsersHelper

For more information about yard you can check their Startup Guide. Yardoc.

4 Conclussion

After analyzing the avalaible tools for generation documentation of your code I found out that most of them support additional features that will help to easily browse and read the existing code. You can add comments to your code using markdown syntax thus when the documentation will be generated it will be outputed in the format you desire. Most of these tool provide a lot functionality out of the box and provide an easy way to create documentation for your code.