RSpec API Documentation

Eric Oestrich
@ericoestrich

smartlogic

Who?

Zipmark - zipmark.com

Zipmark makes it simple and safe to quickly pay your rent, bills, and even your friends for that coffee they bought you.

SmartLogic Solutions - smartlogicsolutions.com

We build applications to support the our clients' unique visions.

What?

- Write tests that describe how your API should behave
- Automatically generate API documentation based on your tests

Keep your API docs in sync with your code

If your tests fail, your API is broken

Why?

- Eliminate time spent writing API docs
- Actually write docs
- Document API while under active development

How?

- RSpec Formatters
- RSpec Metadata
- Extending RSpec with a custom DSL

Formatters

- A user-defined observer-type class, thing that does a thing when something happens
- Hooks -
  - example_group_started
  - example_passed
  - example_failed
  - start
  - stop
- Has access to metadata

Formatters

```
module RspecApiDocumentation
  class ApiFormatter < RSpec::Core::Formatters::BaseFormatter
    def example_passed(example)
      super

      output.puts "     * #{example.description}"

      RspecApiDocumentation.documentations.each do |documentation|
        documentation.document_example(example)
      end
    end

    def example_failed(example)
      super

      output.puts "     ! #{example.description} (FAILED)"
    end
  end
end
```

Formatters

Use by running with -

```
rspec spec/ --format \
   RspecApiDocumentation::ApiFormatter
```

Comes with a Railtie that defines -

```
rake docs:generate
```

Metadata

"You can attach user-defined metadata to any example group or example. Pass a hash as the last argument (before the block) to describe, context or it. RSpec supports many configuration options that apply only to certain examples or groups based on the metadata."

- Rspec Core Docs, Metadata

Metadata

```
describe "a group with user-defined metadata", :foo => 17 do
  it 'has access to the metadata in the example' do
    example.metadata[:foo].should eq(17)
  end

  it 'does not have access to metadata defined on sub-groups' do
    example.metadata.should_not include(:bar)
  end

  describe 'a sub-group with user-defined metadata', :bar => 12 do
    it 'has access to the sub-group metadata' do
      example.metadata[:foo].should eq(17)
    end

    it 'also has access to metadata defined on parent groups' do
      example.metadata[:bar].should eq(12)
    end
  end
end
```

Metadata

In the formatter, we can look at examples' metadata to write API documentation

For non-trivial metadata usage, it's cumbersome and error prone.

Metadata

```
it "Should create an order",
        :resource_name=>"Orders", :document=>true, :parameters=>[{:name=>"
        format", :description=>"Format of response", :required=>true}, {:
        name=>"name", :description=>"Name of order", :scope=>:order}, {:
        name=>"paid", :description=>"If the order has been paid for", :
        scope=>:order}, {:name=>"email", :description=>"Email of user
        that placed the order", :scope=>:order}], :method=>:post, :
        path=>"/orders.:format" do

    test_client.post("/orders.json", :email => "email@example.com")
end
```

Who wants to do that?

DSL

```
resource "Orders" do
  parameter :format, "Format of response"

  required_parameters :format

  let(:format) { :json }

  post "/orders.:format" do
    parameter :name, "Name of order"
    parameter :paid, "If the order has been paid for"
    parameter :email, "Email of user that placed the order"

    let(:email) { "email@example.com" }

    scope_parameters :order, [:name, :paid, :email]

    example "Creating an order" do
      do_request
    end
  end
end
```

DSL

- Rspec provides an API for extending its DSL
- Use metadata to augment test logic

Extending RSpec

- RSpec will include a module when metadata criteria are met

```
RSpec.configuration.include RspecApiDocumentation::DSL, :api_docs_dsl
=> true
```

Augment Test Logic

parameter

```
resource "Orders" do
  parameter :name

  let(:name) { "My Order" }
  # automatically sets params to {:name => "My Order"}
end
```

scope_parameter

```
resource "Orders" do
  parameter :name
  scope_parameters :order, [:name]

  let(:name) { "My Order" }
  # automatically sets params to {:order => {:name => "My Order"}}
end
```

delete, get, post, put

```
resource "Orders" do
  parameter :name
  let(:name) { "My Order" }

  post "/orders" do
    # when do_request is called it will POST to /orders
  end
end
```

do_request

```
resource "Orders" do
  parameter :name
  let(:name) { "My Order" }

  post "/orders" do
    example "Creating an order" do
      do_request
      # POST to /orders and send parameters
    end
  end
end
```

What does it look like?

API Documentation

Orders

Orders API

Getting a list of orders

Parameters

Name		Description
format	required	Format of response
page		Current page of orders

Request

```
Host: example.org
Cookie:
```

```
GET /orders.json?page=1
```

Query Parameters

```
page: 1
```

Response

```
Content-Type: application/json; charset=utf-8
X-Ua-Compatible: IE=Edge,chrome=1
E-Tag: "6cd5831e98b0090b64672b5604d9eb2a"
Cache-Control: max-age=0, private, must-revalidate
X-Runtime: 0.090505
Content-Length: 121
```

```
200 OK
```

```
[
  {
    "email": "email0@example.com",
    "name": "Order 0",
    "paid": true
  },
  {
    "email": "email1@example.com",
    "name": "Order 1",
    "paid": true
  }
]
```

Formats

- HTML
- JSON

Future Additions

- Document multiple requests/responses
- Don't pollute RSpec metadata

Where?

GitHub
github.com/zipmark/rspec_api_documentation