A Test-Driven Approach to Documenting RESTful APIs with Spring REST Docs

Jenn Strater
@codeJENNerator

OBJECT PARTNERS

Follow Along

https://github.com/jlstrater/groovy-spring-boot-restdocs-example

https://github.com/jlstrater/grails-spring-restdocs-example

https://github.com/ratpack/example-books

About Me

About Me

- Senior Consultant at Object Partners, Inc.

About Me

- Senior Consultant at Object Partners, Inc.
- Co-Founder of Gr8Ladies

About Me

- Senior Consultant at Object Partners, Inc.
- Co-Founder of Gr8Ladies
- 2016 - 2017 Fulbright US Student Program Selectee to Denmark

Background

Background

- Creating RESTful APIs with Groovy

Background

- Creating RESTful APIs with Groovy
  - Spring Boot

Background

- Creating RESTful APIs with Groovy
  - Spring Boot
  - Grails

Background

- Creating RESTful APIs with Groovy
  - Spring Boot
  - Grails
  - Ratpack

Background

- Creating RESTful APIs with Groovy
  - Spring Boot
  - Grails
  - Ratpack
- Documentation

Background

- Creating RESTful APIs with Groovy
  - Spring Boot
  - Grails
  - Ratpack
- Documentation
  - Swagger

Background

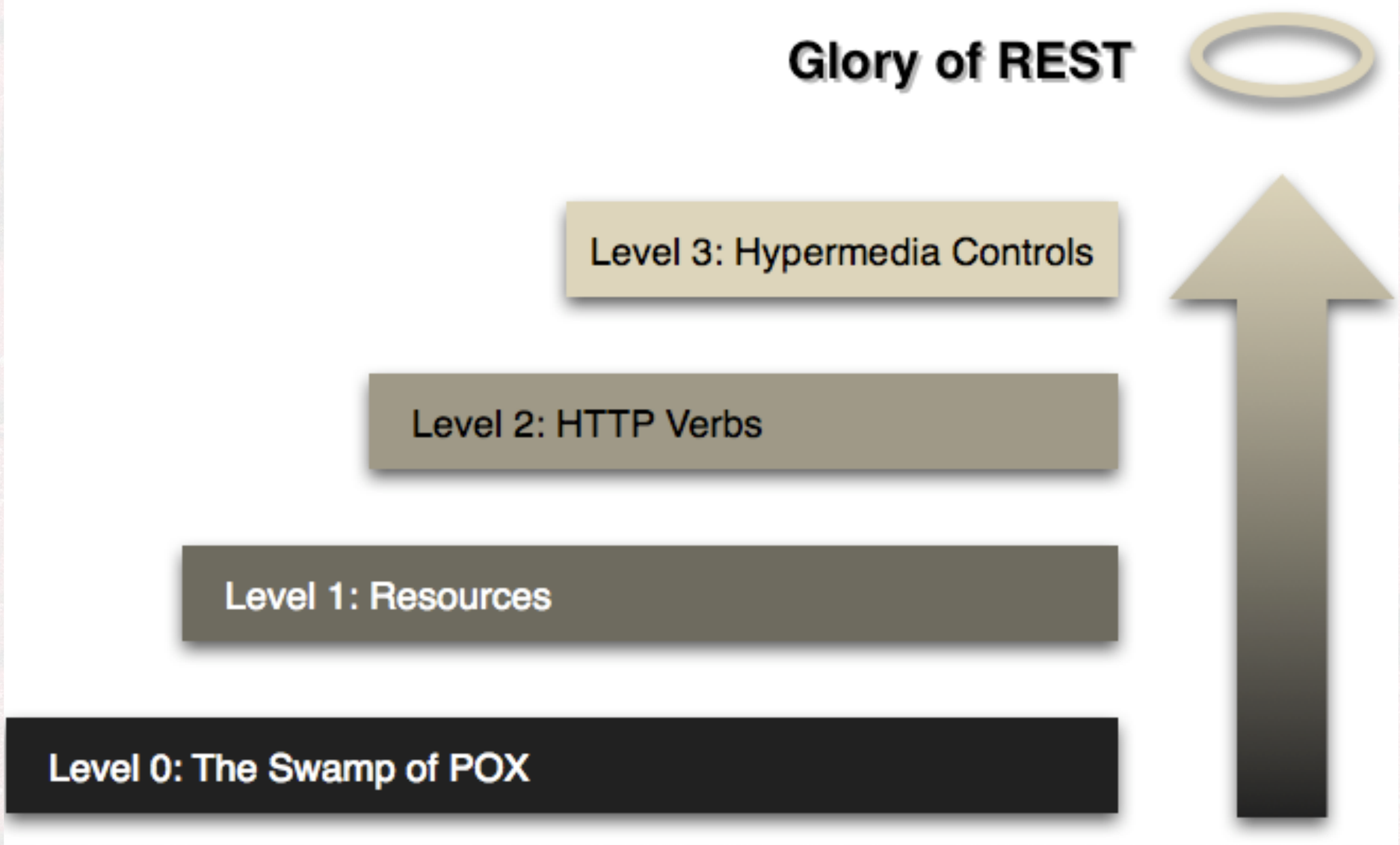
- Creating RESTful APIs with Groovy
  - Spring Boot
  - Grails
  - Ratpack
- Documentation
  - Swagger
  - Asciidoctor

img src: https://flic.kr/p/rehEf5

img src: https://flic.kr/p/rehEf5

I hate writing documentation!

Factors in Choosing a Documentation Solution

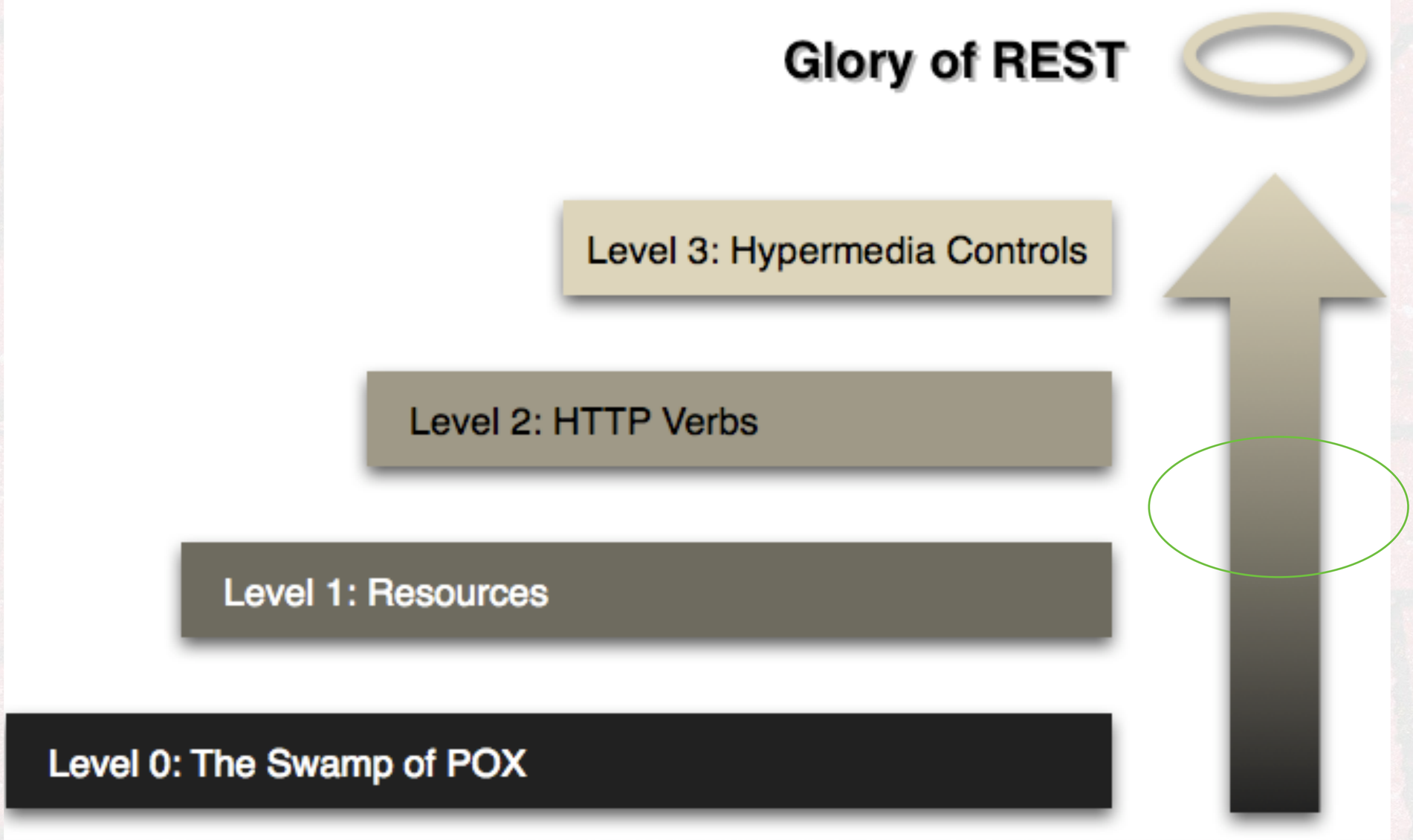


REST Maturity Model

OBJECT PARTNERS

img src: http://martinfowler.com/articles/richardsonMaturityModel.html

@codeJENNerator



REST Maturity Model

OBJECT PARTNERS

img src: http://martinfowler.com/articles/richardsonMaturityModel.html

@codeJENNerator

Endpoint Vs Resource Design

api : get available API versions

GET	/api

api/v1 : API at /api/v1

POST	/api/v1/namespaces/{namespace}/bindings
GET	/api/v1/componentstatuses
GET	/api/v1/componentstatuses/{name}
GET	/api/v1/namespaces/{namespace}/endpoints
POST	/api/v1/namespaces/{namespace}/endpoints
GET	/api/v1/watch/namespaces/{namespace}/endpoints
DELETE	/api/v1/namespaces/{namespace}/endpoints/{name}
GET	/api/v1/namespaces/{namespace}/endpoints/{name}
PATCH	/api/v1/namespaces/{namespace}/endpoints/{name}
PUT	/api/v1/namespaces/{namespace}/endpoints/{name}
GET	/api/v1/watch/namespaces/{namespace}/endpoints/{name}
GET	/api/v1/endpoints
GET	/api/v1/watch/endpoints
GET	/api/v1/namespaces/{namespace}/events

Endpoint Vs Resource Design

api : get available API versions

GET	/api

api/v1 : API at /api/v1

POST	/api/v1/namespaces/{namespace}/bindings
GET	/api/v1/componentstatuses
GET	/api/v1/componentstatuses/{name}
GET	/api/v1/namespaces/{namespace}/endpoints
POST	/api/v1/namespaces/{namespace}/endpoints
GET	/api/v1/watch/namespaces/{namespace}/endpoints
DELETE	/api/v1/namespaces/{namespace}/endpoints/{name}
GET	/api/v1/namespaces/{namespace}/endpoints/{name}
PATCH	/api/v1/namespaces/{namespace}/endpoints/{name}
PUT	/api/v1/namespaces/{namespace}/endpoints/{name}
GET	/api/v1/watch/namespaces/{namespace}/endpoints/{name}
GET	/api/v1/endpoints
GET	/api/v1/watch/endpoints
GET	/api/v1/namespaces/{namespace}/events

VS

Table of Contents

Overview
HTTP verbs
HTTP status codes
Headers
Errors
Hypermedia
Resources
Index
Accessing the index
Response structure
ample response
ks
Listing notes
Response structure
Example request
Example response
Creating a note
Request structure
Example request
Example response
Tags
Listing tags
Response structure
Example request

RESTful N

Andy Wilkinson

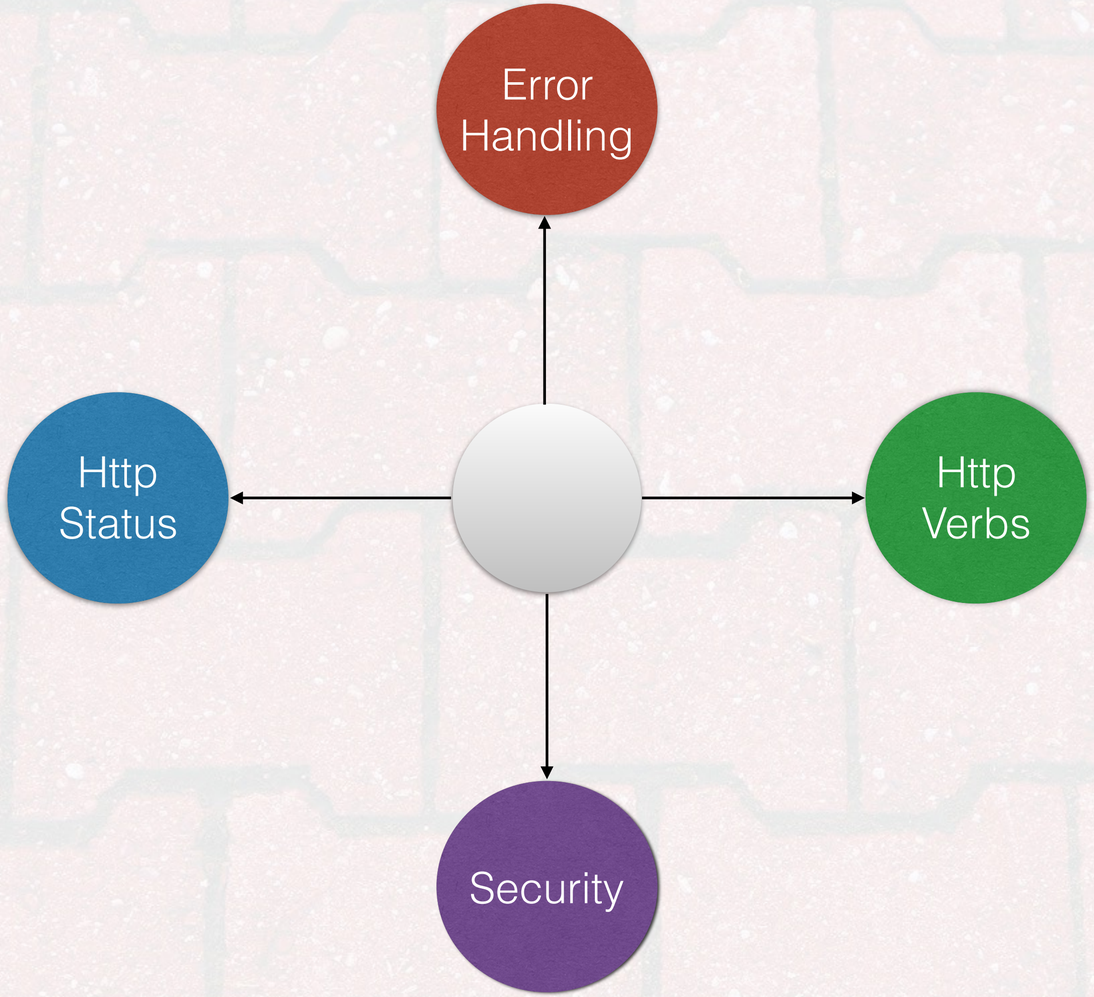
Overview

HTTP verbs

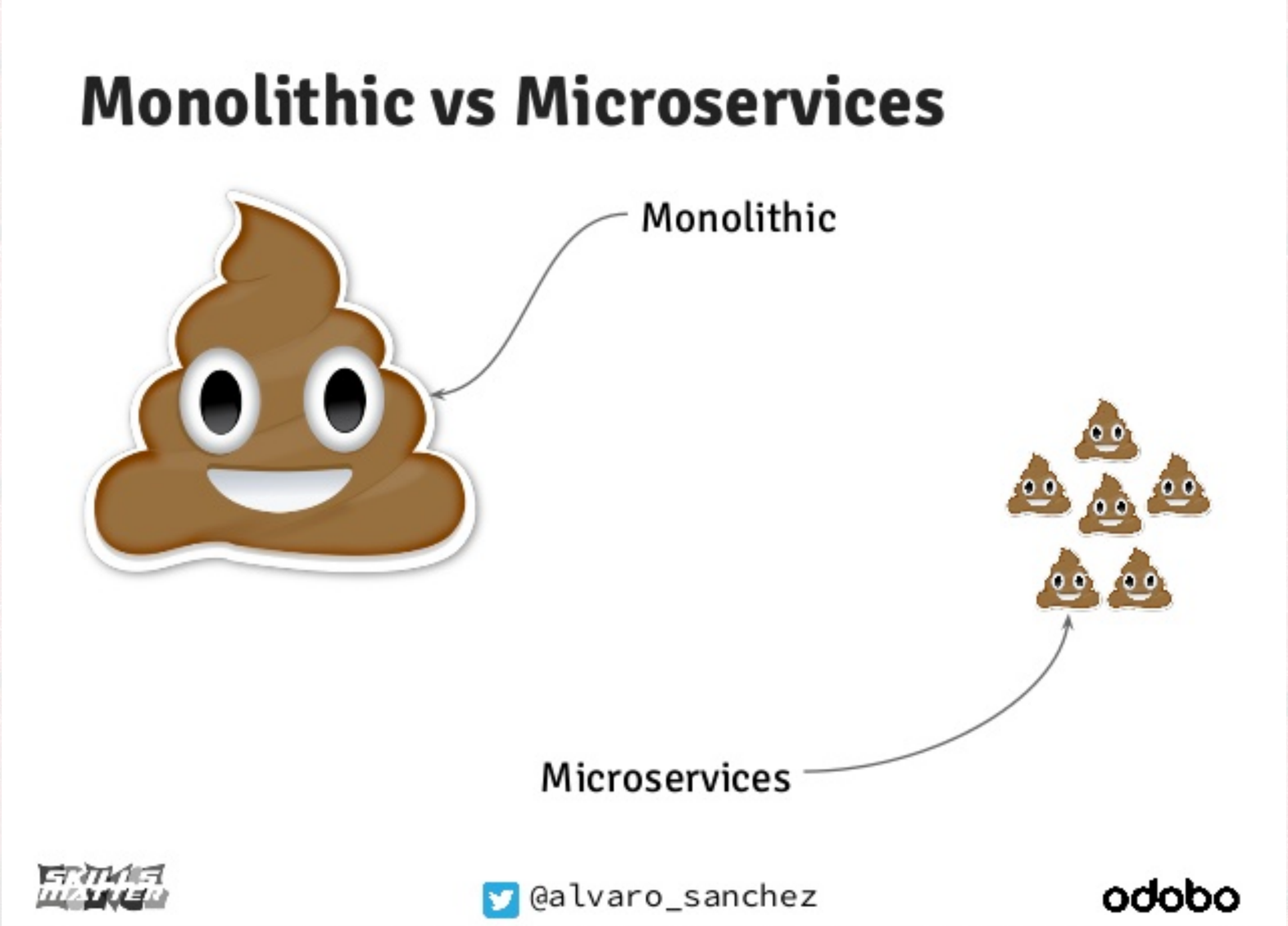
RESTful notes tries to a

Verb
GET
POST
PATCH
DELETE

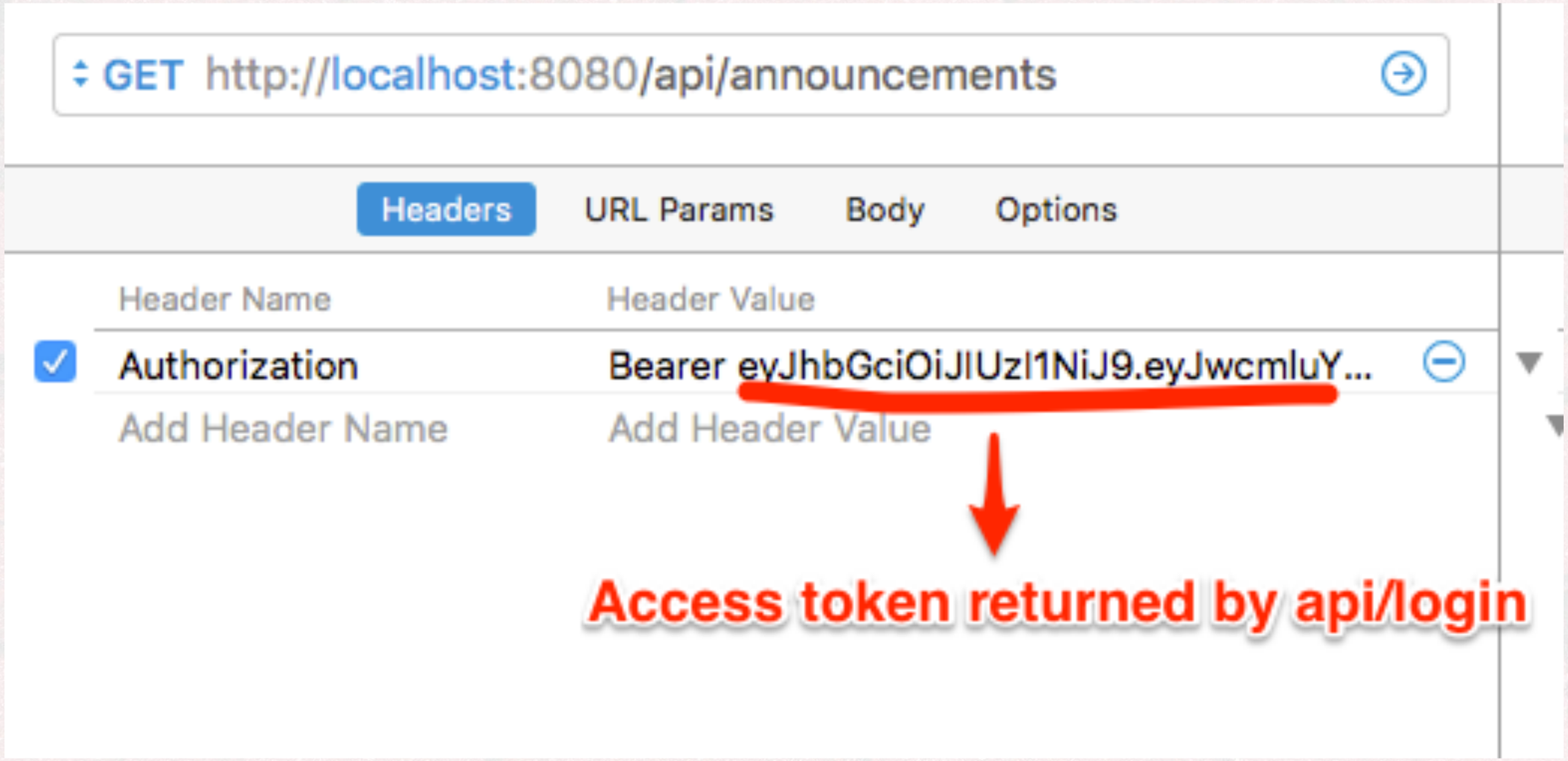
HTTP statu



Central Information



Multiple Services



img src: http://sergiodelamo.es/how-to-secure-your-grails-3-api-with-spring-security-rest-for-grails/

Headers

I'M A NO EXCUSES TYPE OF Chick
2015
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
NOVEMBER
DECEMBER
JANUARY
FEBRUARY
eos

Swagger

Swagger Approaches

SpringFox

OBJECT PARTNERS

img src: https://www.flickr.com/photos/24874528@N04/17125924230

@codeJENNerator

Super Easy Install

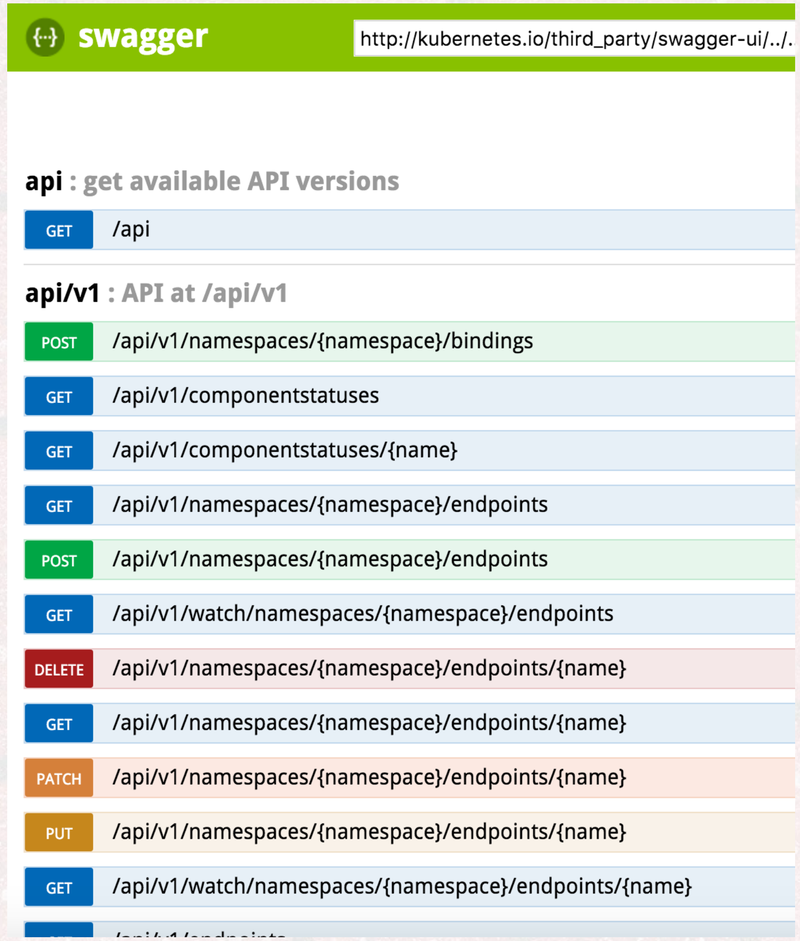
- `io.springfox:springfox-swagger2`
- `io.springfox:springfox-swagger-ui`

```
{
  "swagger": "2.0",
  "info": {
    "version": "1",
    "title": "My Service",
    "contact": {
      "name": "Company Name"
    },
    "license": {}
  },
  "host": "example.com",
  "basepath": "/docs",
  "tags": [
    {
      "name": "group name",
      "description": "what this group of api endpoints is for",
    }
  ],
  "paths": {
    "/api/v1/groupname/resource":
    .
    .
    .
}
```

Considerations

Swagger UI

- Springfox generated UI
- Copy static files and customize



```
1  @Secured('ROLE_ALLOWED_TO_PERFORM_ACTION')
2  @RequestMapping(value = '/v1/serviceName/actionName', method =
3    RequestMethod.POST)
4  @ApiOperation(value = '/actionName',
5    notes = 'Enables or disables setting via "1" or "0", respectively')
6  @ApiResponses(value = [
7    @ApiResponse(code = 200, response = CustomSettingResponse, message =
8      'Successful setting update'),
9    @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
10     user input'),
11   @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
12     server error')
13 ])
14 CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
15   SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
16     settingsValue.fieldOne,
17       [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
18         new Double(settingsValue.value))]
19   )
20   api.saveUpdatedSetting(request)
21 }
```

```
1  @Secured('ROLE_ALLOWED_TO_PERFORM_ACTION')
2  @RequestMapping(value = '/v1/serviceName/actionName', method =
3    RequestMethod.POST)
4  @ApiOperation(value = '/actionName',
5    notes = 'Enables or disables setting via "1" or "0", respectively')
6  @ApiResponses(value = [
7    @ApiResponse(code = 200, response = CustomSettingResponse, message =
8      'Successful setting update'),
9    @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
10     user input'),
11   @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
12     server error')
13 ])
14 CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
15   SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
16     settingsValue.fieldOne,
17       [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
18         new Double(settingsValue.value))]
19   )
20   api.saveUpdatedSetting(request)
21 }
```

Annotation Hell

```
1  @Secured('ROLE_ALLOWED_TO_PERFORM_ACTION')
2  @RequestMapping(value = '/v1/serviceName/actionName', method =
3    RequestMethod.POST)
4  @ApiOperation(value = '/actionName',
5    notes = 'Enables or disables setting via "1" or "0", respectively')
6  @ApiResponses(value = [
7    @ApiResponse(code = 200, response = CustomSettingResponse, message =
8      'Successful setting update'),
9    @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
10     user input'),
11   @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
12     server error')
13 ])
14 CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
15   SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
16     settingsValue.fieldOne,
17       [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
18         new Double(settingsValue.value))]
19   )
20   api.saveUpdatedSetting(request)
21 }
```

Annotation Hell

```
1  @Secured('ROLE_ALLOWED_TO_PERFORM_ACTION')
2  @RequestMapping(value = '/v1/serviceName/actionName', method =
3    RequestMethod.POST)
4  @ApiOperation(value = '/actionName',
5    notes = 'Enables or disables setting via "1" or "0", respectively')
6  @ApiResponses(value = [
7    @ApiResponse(code = 200, response = CustomSettingResponse, message =
8      'Successful setting update'),
9    @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
10     user input'),
11   @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
12     server error')
13 ])
14 CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
15   SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
16     settingsValue.fieldOne,
17       [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
18         new Double(settingsValue.value))]
19   )
20   api.saveUpdatedSetting(request)
21 }
```

Annotation Hell

```
1  @Secured('ROLE_ALLOWED_TO_PERFORM_ACTION')
2  @RequestMapping(value = '/v1/serviceName/actionName', method =
3    RequestMethod.POST)
4  CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
5   SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
6     settingsValue.fieldOne,
7       [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
8         new Double(settingsValue.value))]
9   )
10   api.saveUpdatedSetting(request)
11 }
```

```
@ApiModel(value = "Model", description = "Model description")
public class Model {
        @ApiModelProperty(value = "variable0", dataType = "java.lang.String", required = tru
        private String variable0;
        @ApiModelProperty(value = "variable1", dataType = "java.lang.String", required = tru
        private String variable1;
        //required not set because default = false
        @ApiModelProperty(value = "variable2", dataType = "java.lang.String")
        private String variable2;
        @ApiModelProperty(value = "variable3", dataType = "java.lang.Integer", required = tr
        private Integer variable3;
...
}
```

img src: https://github.com/springfox/springfox/issues/281

Object Mapping

Advantages

Response Class (Status 200)

Model | Model Schema

```
{
  "message": "string",
  "status": "OK"
}
```

Response Content Type */*

Parameters

Parameter	Value	Description	Parameter Type	Data Type
settingsValue	Parameter content type: application/json	settingsValue	body	Model | Model Schema { "dsn": 0, "value": 0 } Click to set as parameter value

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

Swagger UI

"Try-It" Button

The Alternative

API Console Tool

Public API

- Uploading Media
- The Search API
- The Search API: Tweets by Place
- Working with Timelines
- API Rate Limits
- API Rate Limits: Chart
- GET statuses/mentions_timeline
- GET statuses/user_timeline
- GET statuses/home_timeline
- GET statuses/retweets_of_me
- GET statuses/retweets/:id
- GET statuses/show/:id
- POST statuses/destroy/:id
- POST statuses/update
- POST statuses/retweet/:id

REST APIs

Run in Postman

The REST APIs provide programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

If your intention is to monitor or process Tweets in real-time, consider using the Streaming API instead.

Overview

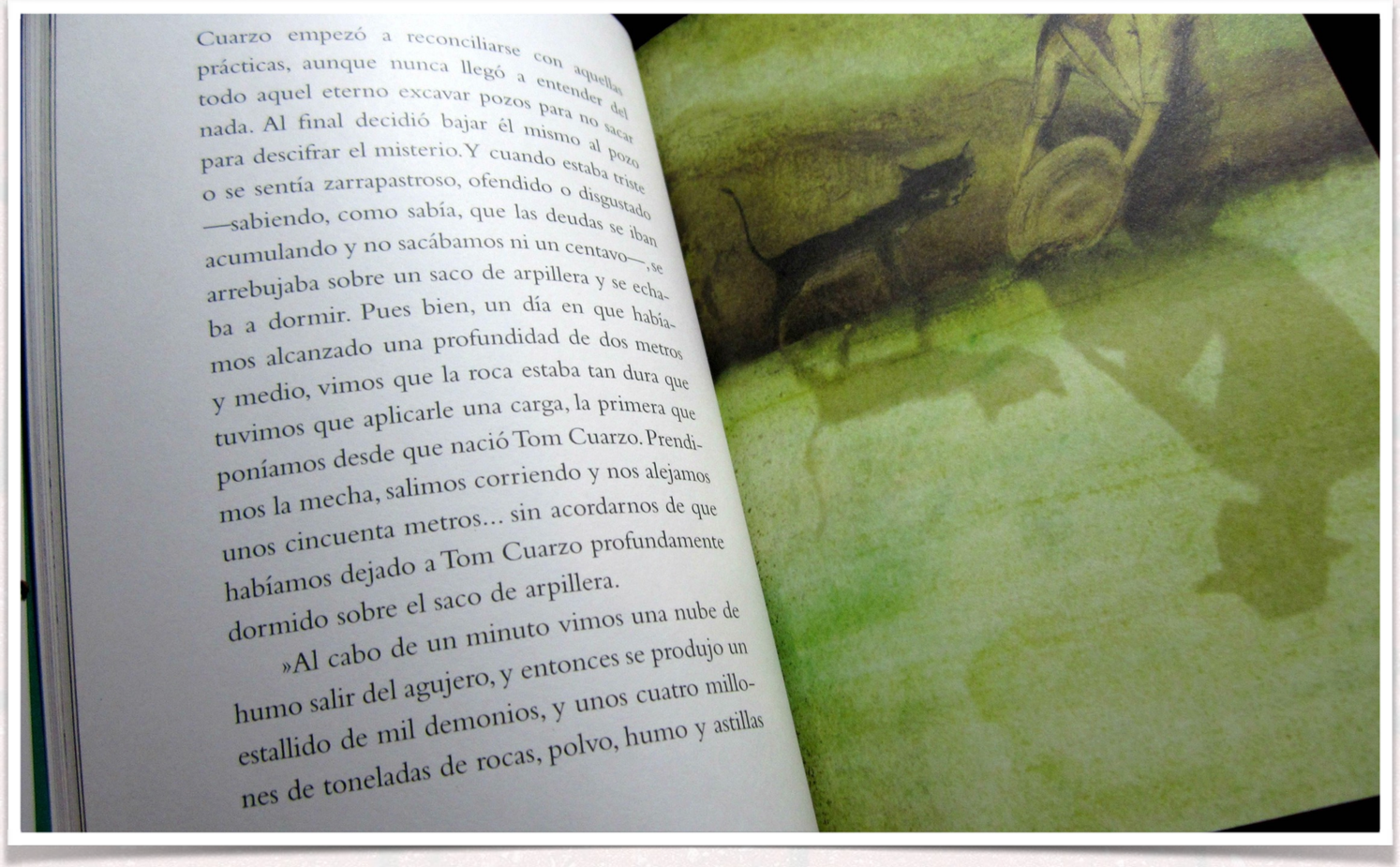
Below are the documents that will help you get going with the REST APIs as quickly as possible

- API Rate Limiting
- API Rate Limits
- Working with Timelines
- Using the Twitter Search API
- Uploading Media
- Multiple Media Entities in Statuses
- Finding Tweets about Places

Postman Run Button

src: https://www.getpostman.com/img/v1/docs/run_btn_ux/run_btn_ux_1.png

Mixing Spring REST Docs and Swagger



Cuarzo empezó a reconciliarse con aquellas prácticas, aunque nunca llegó a entender del todo aquel eterno excavar pozos para no sacar nada. Al final decidió bajar él mismo al pozo para descifrar el misterio. Y cuando estaba triste o se sentía zarrapastroso, ofendido o disgustado —sabiendo, como sabía, que las deudas se iban acumulando y no sacábamos ni un centavo—, se arrebujaba sobre un saco de arpillera y se echaba a dormir. Pues bien, un día en que habíamos alcanzado una profundidad de dos metros y medio, vimos que la roca estaba tan dura que tuvimos que aplicarle una carga, la primera que poníamos desde que nació Tom Cuarzo. Prendimos la mecha, salimos corriendo y nos alejamos unos cincuenta metros... sin acordarnos de que habíamos dejado a Tom Cuarzo profundamente dormido sobre el saco de arpillera.

»Al cabo de un minuto vimos una nube de humo salir del agujero, y entonces se produjo un estallido de mil demonios, y unos cuatro millones de toneladas de rocas, polvo, humo y astillas

Swagger2Markup

https://swagger2markup.readme.io/

Table of Contents

1. General notes
2. Authentication and authorization
3. API User Guide
   3.1. URL syntax
   3.2. HTTP Verbs
   3.3. HTTP Redirects
   3.4. Simple query parameters
   3.5. Extended query parameters
   3.6. Array parameter notation in queries
   3.7. Error Handling
   3.8. HTTP Headers
4. Paths
   4.1. Get contacts
   4.2. Create a contact
   4.3. Delete contacts
   4.4. Delete address book
   4.5. Delete all contacts
   4.6. Get birthdays of contacts
   4.7. Get all backups
   4.8. Create a backup
   4.9. Get a backup
   4.10. Delete a backup
   4.11. Restore a backup
   4.12. Check duplicates
   4.13. Export all contacts as embedded CSV
   4.14. Export all contacts as embedded VCard
   4.15. Get frequently used contacts
   4.16. Get all contact groups
   4.17. Create a contact group
   4.18. Delete all contact groups
   4.19. Get a contact group
   4.20. Delete a contact group

4.2. Create a contact

```
PUT [base-url]/rest/contacts/v1
```

4.2.1. Description

Creates a new contact and adds it to the user's address book. If the user has no address book yet, an empty
will be created first.

4.2.2. Parameters

Type	Name	Description	Required	Schema	De
BodyParameter	contact	The contact to create and add to the user's address book.	true	Contact	

4.2.3. Responses

HTTP Code	Description	Schema
200	OK	ContactResponse
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

4.2.4. Consumes

- application/xml
- application/json

Table of Contents

1. General notes
2. Authentication and authorization
3. API User Guide
   3.1. URL syntax
   3.2. HTTP Verbs
   3.3. HTTP Redirects
   3.4. Simple query parameters
   3.5. Extended query parameters
   3.6. Array parameter notation in queries
   3.7. Error Handling
   3.8. HTTP Headers
4. Paths
   4.1. Get contacts
   4.2. Create a contact
   4.3. Delete contacts
   4.4. Delete address book
   4.5. Delete all contacts
   4.6. Get birthdays of contacts
   4.7. Get all backups
   4.8. Create a backup
   4.9. Get a backup
   4.10. Delete a backup
   4.11. Restore a backup
   4.12. Check duplicates
   4.13. Export all contacts as embedded CSV
   4.14. Export all contacts as embedded VCard
   4.15. Get frequently used contacts
   4.16. Get all contact groups
   4.17. Create a contact group
   4.18. Delete all contact groups
   4.19. Get a contact group
   4.20. Delete a contact group

4.2. Create a contact

```
PUT [base-url]/rest/contacts/v1
```

4.2.1. Description

Creates a new contact and adds it to the user's address book. If the user has no address book yet, an empty
will be created first.

4.2.2. Parameters

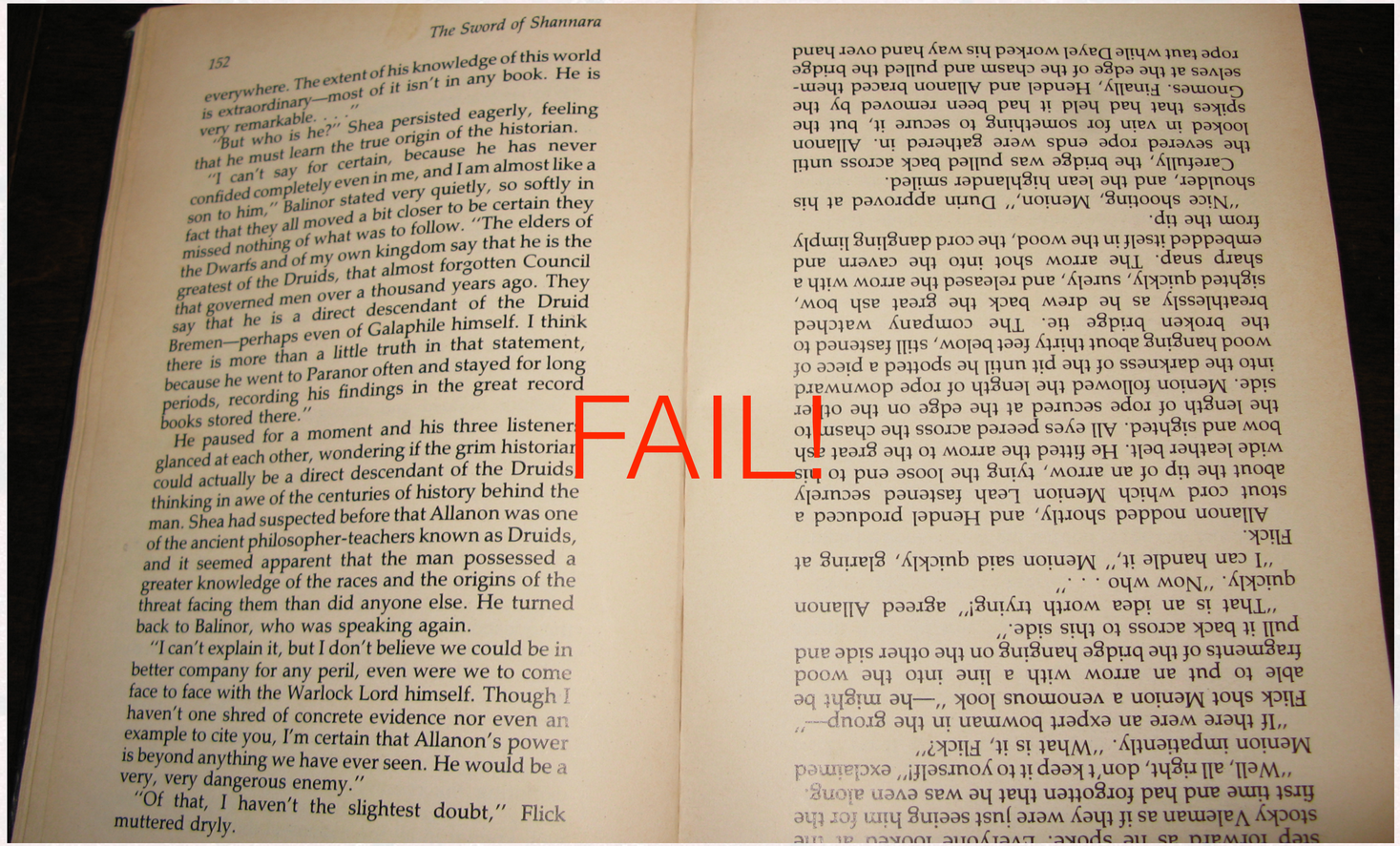
Type	Name	Description	Required	Schema	De
BodyParameter	contact	The contact to create and add to the user's address book.	true	Contact	

4.2.3. Responses

HTTP Code	Description	Schema
200	OK	ContactResponse
201	Created	No Content
401	Unauthorized	No Content
403	Forbidden	No Content
404	Not Found	No Content

4.2.4. Consumes

- application/xml
- application/json



everywhere. The extent of his knowledge of this world is extraordinary—most of it isn't in any book. He is very remarkable. . . . ''

''But who is he?'' Shea persisted eagerly, feeling that he must learn the true origin of the historian.

''I can't say for certain, because he has never confided completely even in me, and I am almost like a son to him,'' Balinor stated very quietly, so softly in fact that they all moved a bit closer to be certain they missed nothing of what was to follow. ''The elders of the Dwarfs and of my own kingdom say that he is the greatest of the Druids, that almost forgotten Council that governed men over a thousand years ago. They say that he is a direct descendant of the Druid Bremen—perhaps even of Galaphile himself. I think there is more than a little truth in that statement, because he went to Paranor often and stayed for long periods, recording his findings in the great record books stored there.''

He paused for a moment and his three listeners glanced at each other, wondering if the grim historian could actually be a direct descendant of the Druids, thinking in awe of the centuries of history behind the man. Shea had suspected before that Allanon was one of the ancient philosopher-teachers known as Druids, and it seemed apparent that the man possessed a greater knowledge of the races and the origins of the threat facing them than did anyone else. He turned back to Balinor, who was speaking again.

''I can't explain it, but I don't believe we could be in better company for any peril, even were we to come face to face with the Warlock Lord himself. Though I haven't one shred of concrete evidence nor even an example to cite you, I'm certain that Allanon's power is beyond anything we have ever seen. He would be a very, very dangerous enemy.''

''Of that, I haven't the slightest doubt,'' Flick muttered dryly.

FAIL!

step forward as he spoke. Everyone looked at the stocky Valeman as if they were just seeing him for the first time and had forgotten that he was even along.

''Well, all right, don't keep it to yourself!'' exclaimed Menion impatiently. ''What is it, Flick?''

''If there were an expert bowman in the group—'' Flick shot Menion a venomous look ''—he might be able to put an arrow with a line into the wood fragments of the bridge hanging on the other side and pull it back across to this side.''

''That is an idea worth trying!'' agreed Allanon quickly. ''Now who . . .''

''I can handle it,'' Menion said quickly, glaring at Flick.

Allanon nodded shortly, and Hendel produced a stout cord which Menion Leah fastened securely about the tip of an arrow, tying the loose end to his wide leather belt. He fitted the arrow to the great ash bow and sighted. All eyes peered across the chasm to the length of rope secured at the edge on the other side. Menion followed the length of rope downward into the darkness of the pit until he spotted a piece of wood hanging about thirty feet below, still fastened to the broken bridge tie. The company watched breathlessly as he drew back the great ash bow, sighted quickly, surely, and released the arrow with a sharp snap. The arrow shot into the cavern and embedded itself in the wood, the cord dangling limply from the tip.

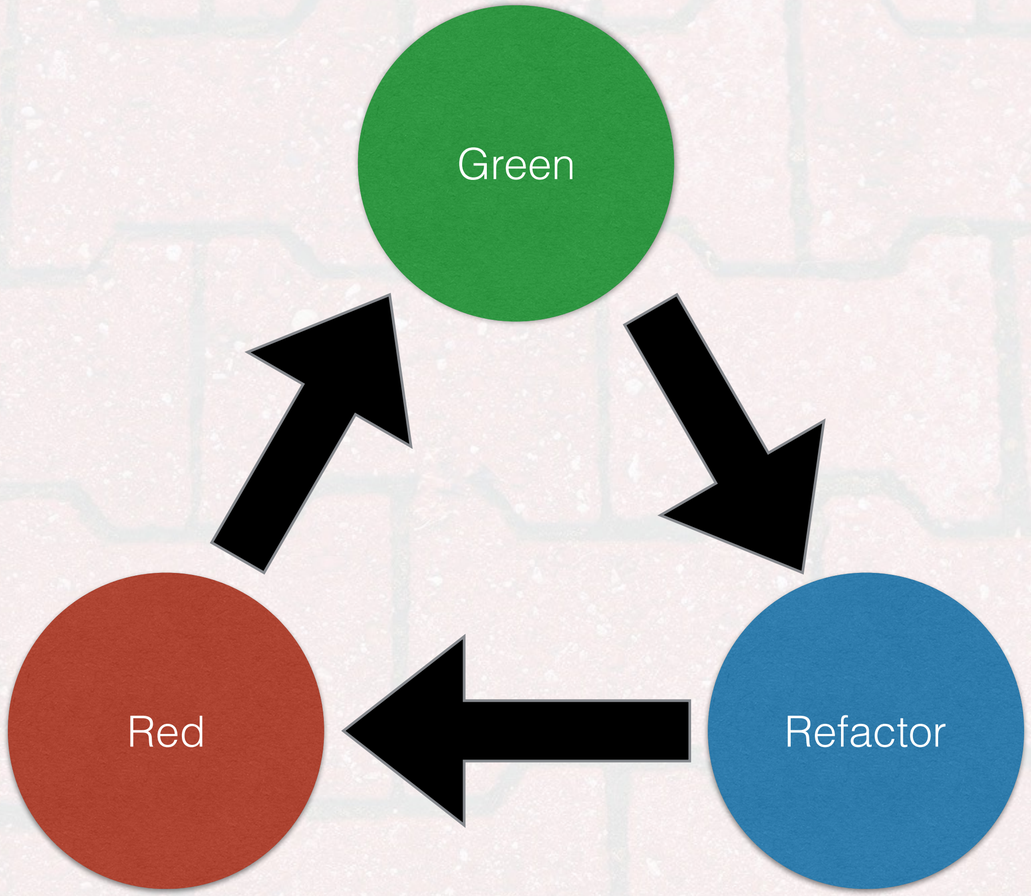
''Nice shooting, Menion,'' Dunin approved at his shoulder, and the lean highlander smiled.

Carefully, the bridge was pulled back across until the severed rope ends were gathered in. Allanon looked in vain for something to secure it, but the spikes that had held it had been removed by the Gnomes. Finally, Hendel and Allanon braced themselves at the edge of the chasm and pulled the bridge rope taut while Dayel worked his way hand over hand

img src: http://www.elvenspirit.com/elf/wp-content/uploads/2011/10/IMG_3013.jpg

AssertJ-Swagger

https://github.com/RobWin/assertj-swagger



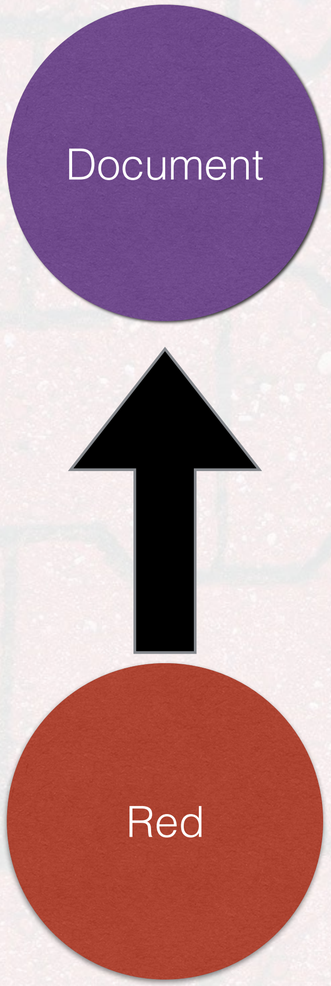
Test-Driven Documentation

Test-Driven Documentation

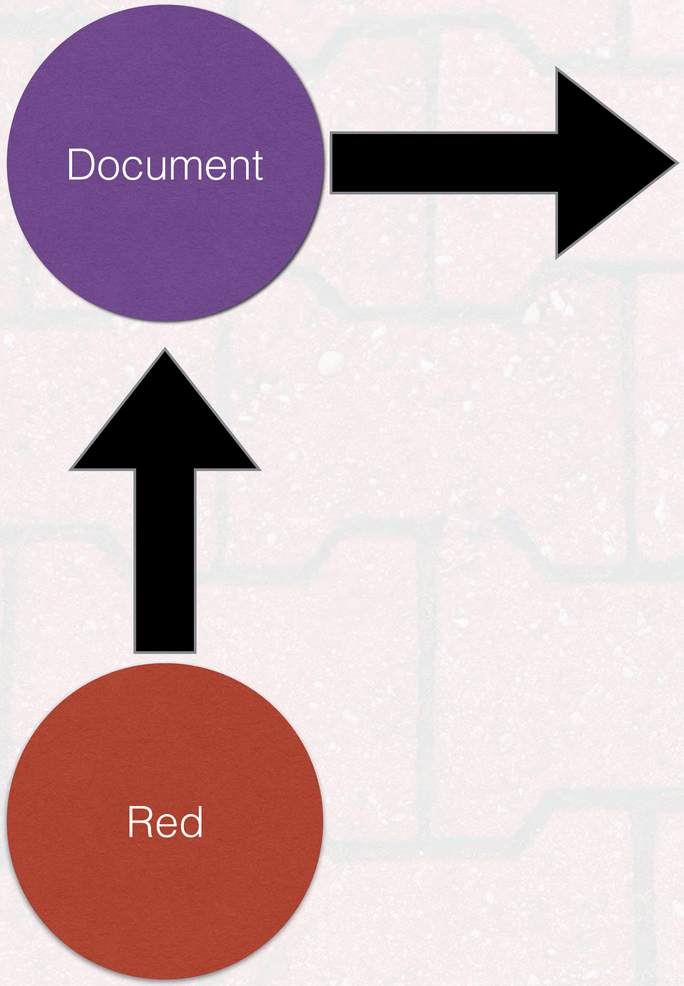
Red

Test-Driven Documentation

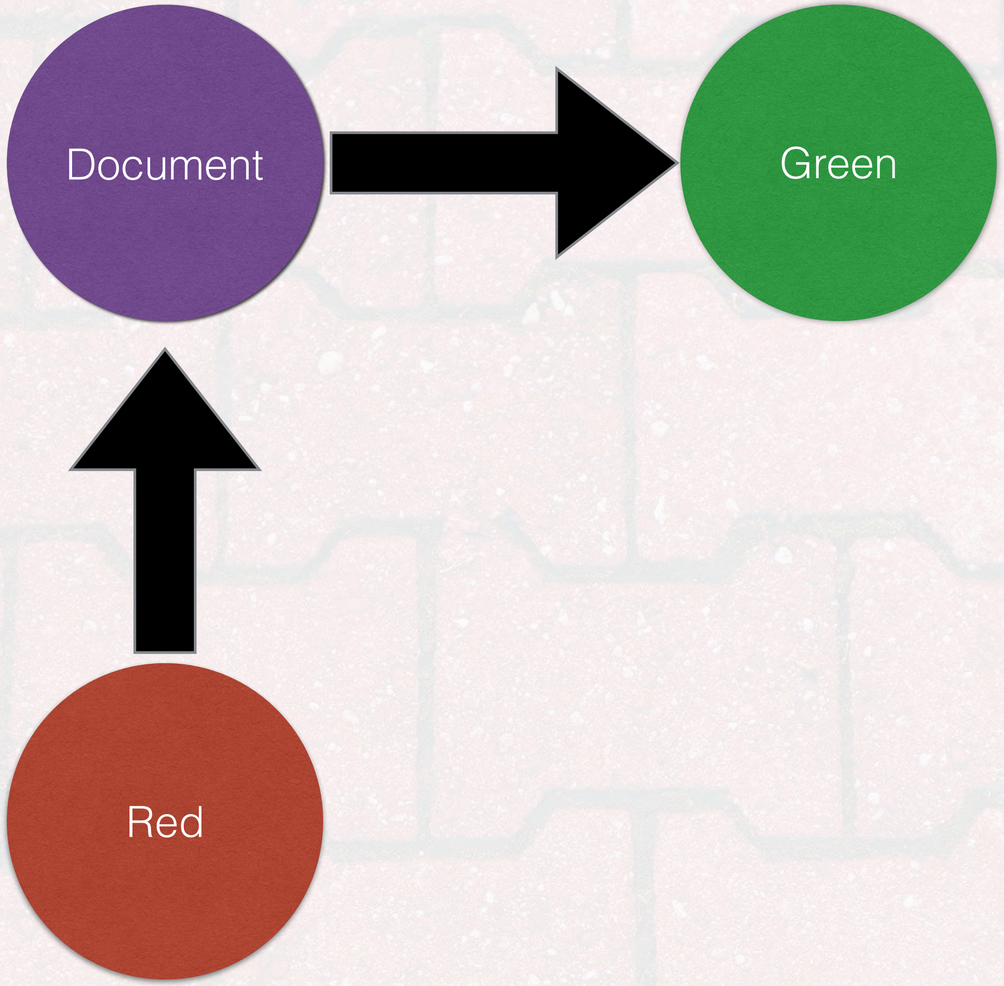
Test-Driven Documentation



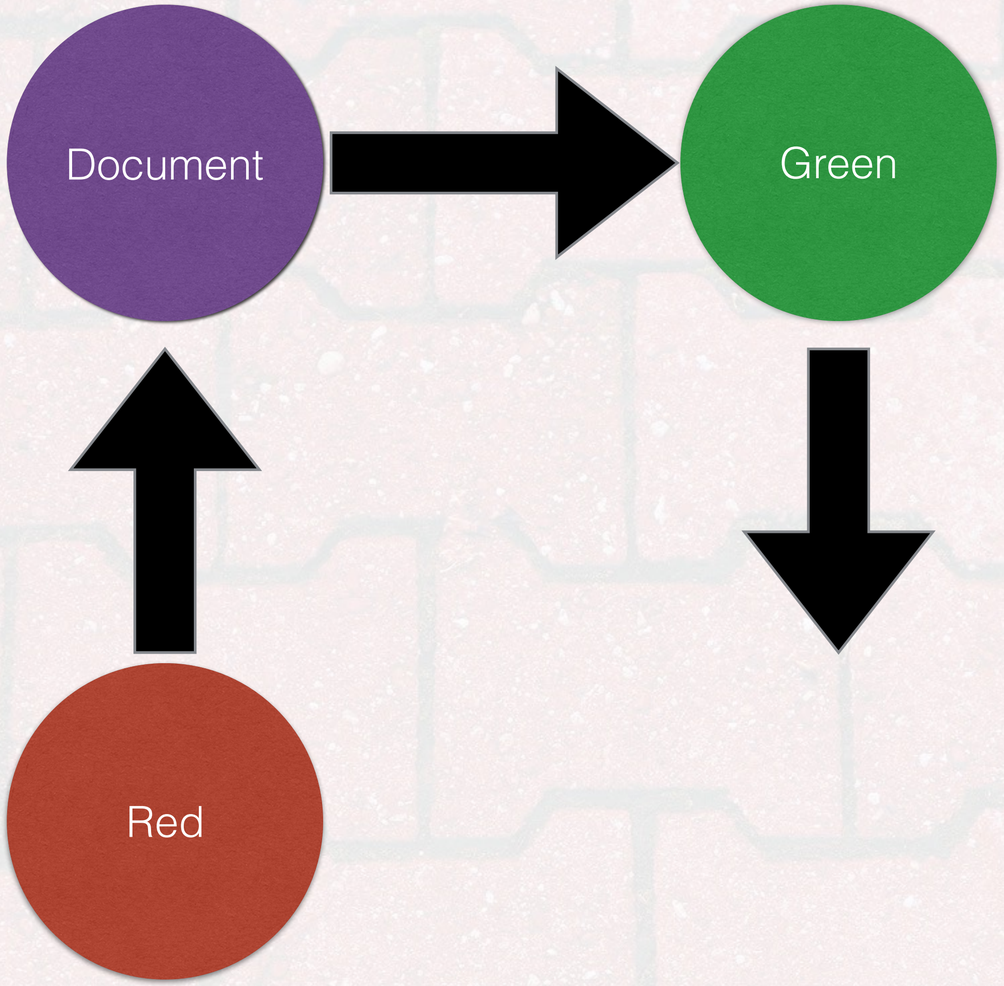
Test-Driven Documentation



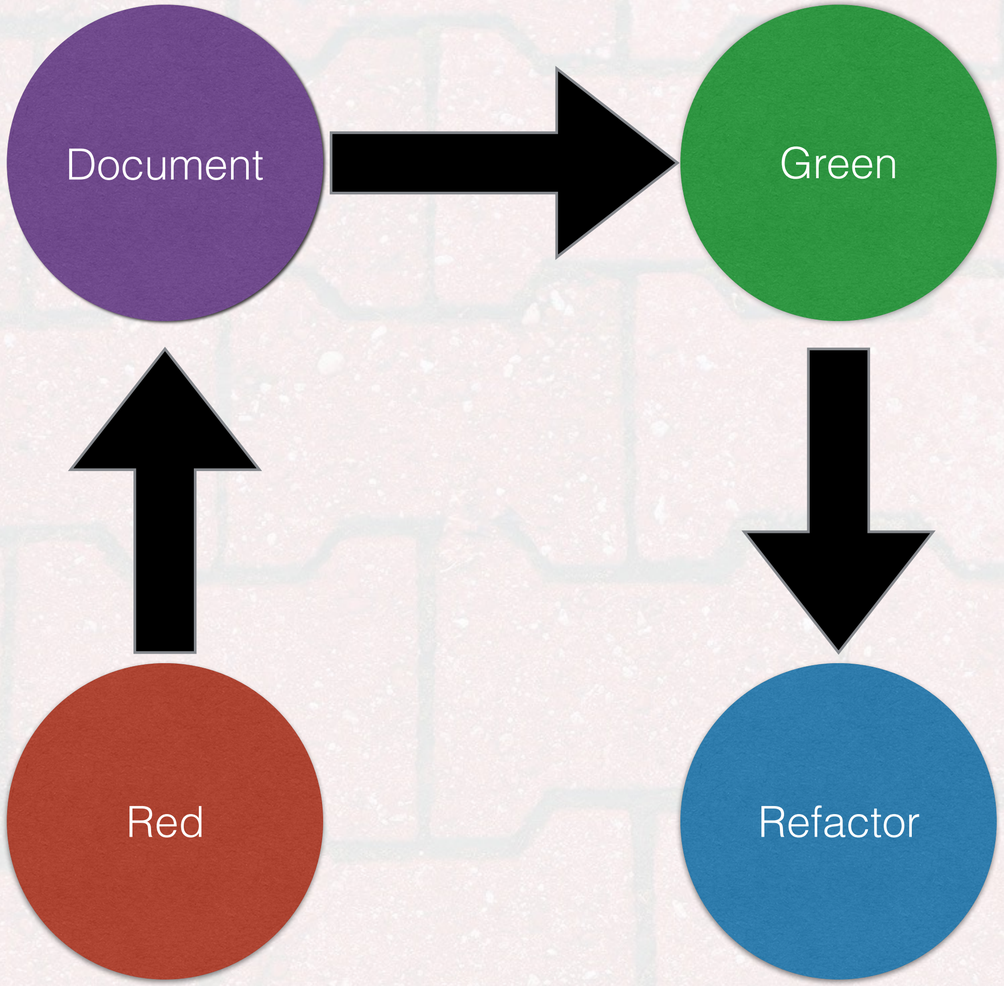
Test-Driven Documentation



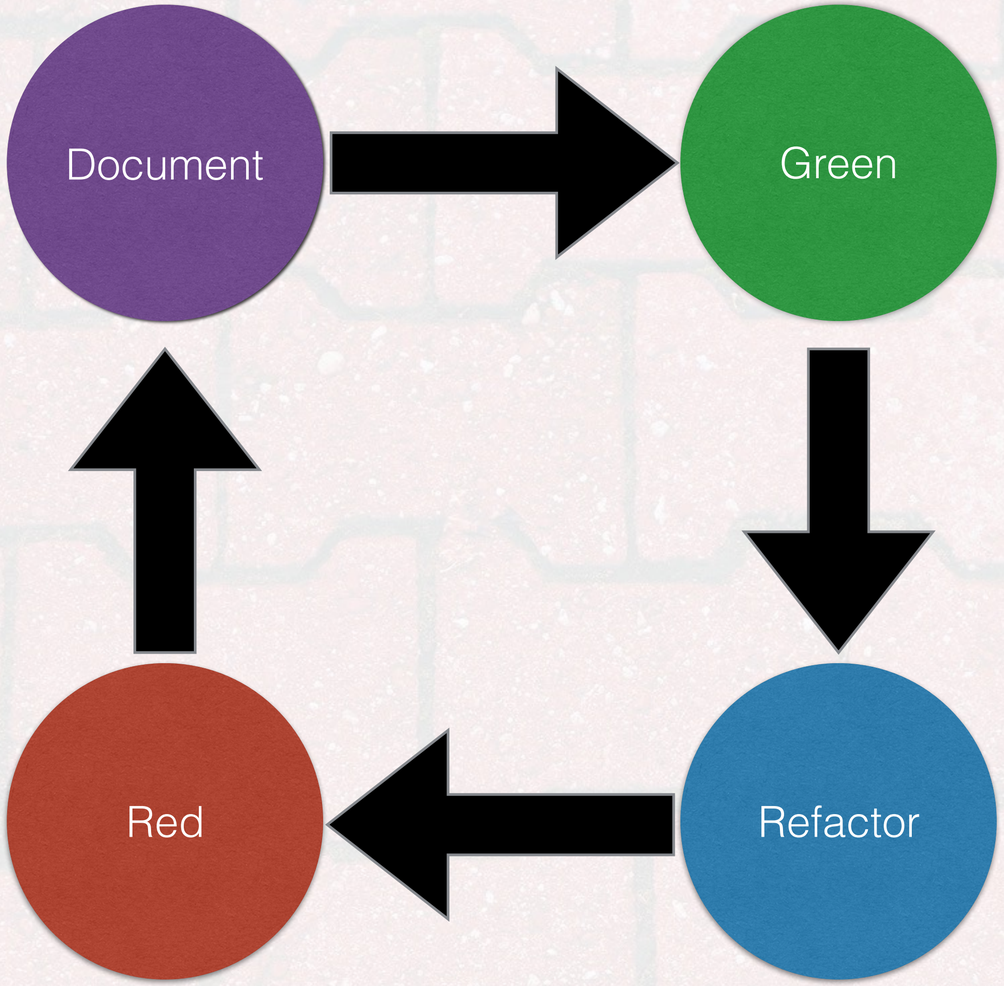
Test-Driven Documentation



Test-Driven Documentation



Test-Driven Documentation



Test-Driven Documentation

Advantages

https://flic.kr/p/81RP5v

Advantages

- Ensure documentation matches implementation

https://flic.kr/p/81RP5v

Advantages

- Ensure documentation matches implementation
- Encourages writing more tests

https://flic.kr/p/81RP5v

Advantages

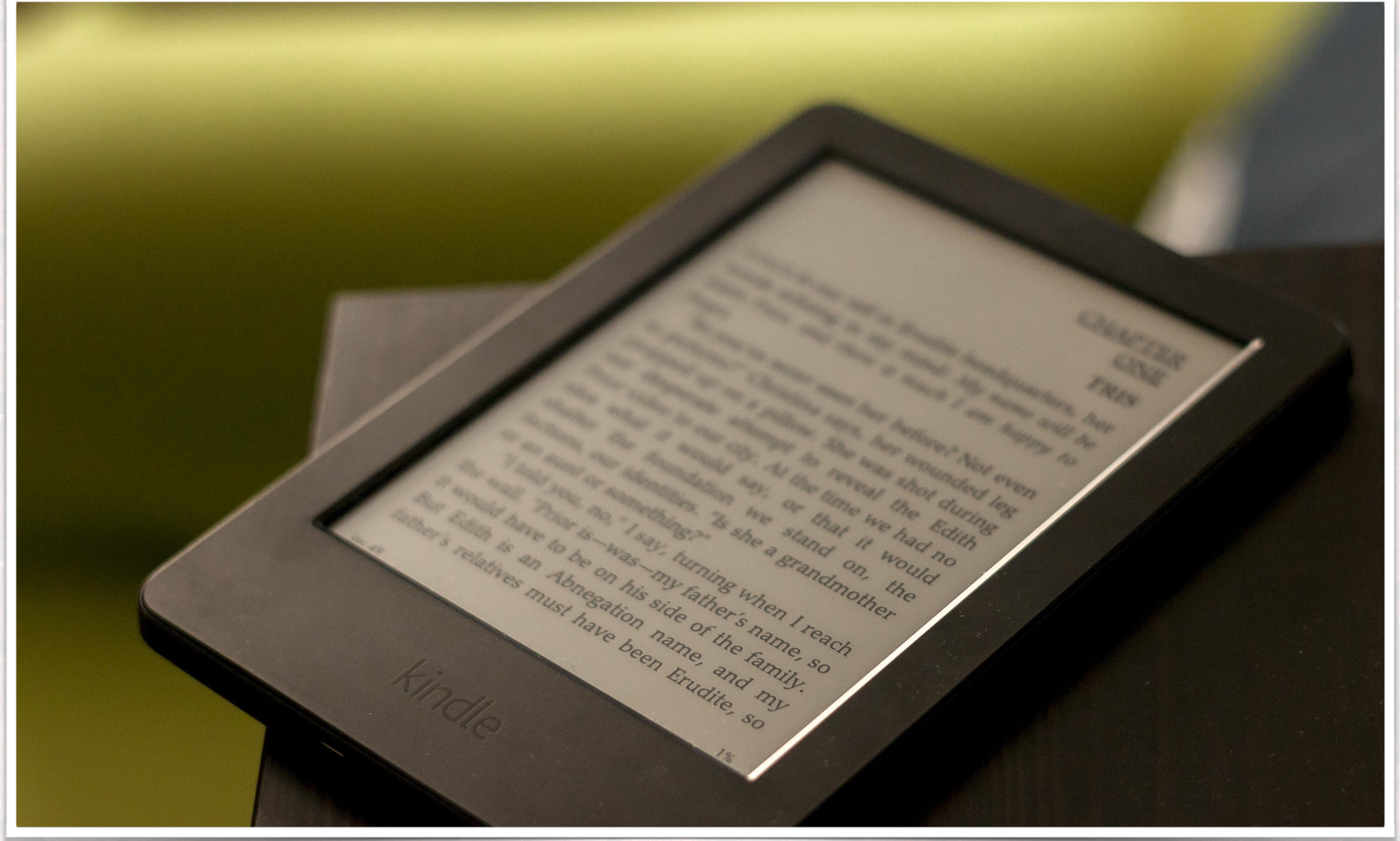
- Ensure documentation matches implementation
- Encourages writing more tests
- Reduces duplication

https://flic.kr/p/81RP5v

Advantages

- Ensure documentation matches implementation
- Encourages writing more tests
- Reduces duplication
- Removes annotations from source

https://flic.kr/p/81RP5v



Spring REST Docs

OBJECT PARTNERS

@codeJENNerator

Overview

projects.spring.io/spring-restdocs

Overview

- Sponsored by Pivotal

projects.spring.io/spring-restdocs

Overview

- Sponsored by Pivotal
- Project Lead
  - Andy Wilkinson

projects.spring.io/spring-restdocs

Overview

- Sponsored by Pivotal
- Project Lead
  - Andy Wilkinson
- Current Version
  - 1.1.1

projects.spring.io/spring-restdocs

Overview

- Sponsored by Pivotal
- Project Lead
  - Andy Wilkinson
- Current Version
  - 1.1.1

NEW

projects.spring.io/spring-restdocs

Game Changers

https://flic.kr/p/9Tiv3U

- Generated code snippets
- Tests fail when documentation is missing or out-of-date
- Can't Use Swagger
  - Level III Rest APIs — Hypermedia
  - Ratpack

Getting Started

- Documentation - Spring Projects
- Documenting RESTful Apis - SpringOne2GX 2015
  Andy Wilkinson
- Spring REST Docs - Documenting RESTful APIs using your tests - Devoxx Belgium 2015
  Anders Evers

Groovier Spring REST docs

- Spring Boot
- Grails
- Ratpack

Groovier Spring REST docs Example I

Groovier Spring REST docs Example I

- Groovy Spring Boot Project
  - Level I/II Rest API

Groovier Spring REST docs Example I

- Groovy Spring Boot Project
  - Level I/II Rest API

+ Asciidoctor Gradle plugin

Groovier Spring REST docs Example I

- Groovy Spring Boot Project
  - Level I/II Rest API

+ Asciidoctor Gradle plugin

+ **MockMVC** and documentation to Spock tests

Groovier Spring REST docs Example I

- Groovy Spring Boot Project
  - Level I/II Rest API

+ Asciidoctor Gradle plugin

+ **MockMVC** and documentation to Spock tests

+ Add to static assets during build

Groovy Spring Boot App

- Start with lazybones spring boot app
- Add mock endpoints for example

https://github.com/jlstrater/groovy-spring-boot-restdocs-example

Endpoints

```
@CompileStatic
@RestController
@RequestMapping('/hello')
@Slf4j
class ExampleController {

    @RequestMapping(method = RequestMethod.GET, produces =
'application/json', consumes = 'application/json')
    Example list() {
        new Example(name: 'World')
    }

    @RequestMapping(method = RequestMethod.POST, produces =
'application/json', consumes = 'application/json')
    Example list(@RequestBody Example example) {
        example
    }
}
```

Asciidoctor Gradle Plugin

Install

```
classpath 'org.asciidoctor:asciidoctor-gradle-plugin:1.5.3'

apply plugin: 'org.asciidoctor.convert'


asciidoctor {
    sourceDir = file('src/docs')
    outputDir "$projectDir/src/main/resources/public"
    backends 'html5'
    attributes 'source-highlighter' : 'prettify',
            'imagesdir':'images',
            'toc':'left',
            'icons': 'font',
            'setanchors':'true',
            'idprefix':'',
            'idseparator':'-',
            'docinfo1':'true'
}
```

Example AsciiDoc

```
= Gr8Data API Guide
Jenn Strater;
:doctype: book
:icons: font
:source-highlighter: highlightjs
:toc: left
:toclevels: 4
:sectlinks:

[introduction]
= Introduction

The Gr8Data API is a RESTful web service for aggregating and displaying gender
ratios from
various companies across the world.  This document outlines how to submit data from
your company or team and
how to access the aggregate data.

[[overview-http-verbs]]
== HTTP verbs

The Gr8Data API tries to adhere as closely as possible to standard HTTP and REST
conventions in its
use of HTTP verbs.
```

Table of Contents

- Introduction
  - HTTP verbs
  - HTTP status codes
  - Headers
  - Errors
- Resources
  - Aggregate Data
    - Listing aggregates
      - Response structure
      - Example request
      - Example response
  - Companies
    - Listing companies
      - Response structure
      - Example request
      - Example response
    - Submitting Company Data
      - Request structure
      - Example request
      - Example response
  - Company
    - Retrieve A Company Resource
      - Response structure
      - Example request

Gr8Data API Guide

Jenn Strater

Introduction

The Gr8Data API is a RESTful web service for aggregating and displaying gender ratios from various co
world. This document outlines how to submit data from your company or team and how to access the a

HTTP verbs

The Gr8Data API tries to adhere as closely as possible to standard HTTP and REST conventions in its use

Verb	Usage
GET	Used to retrieve a resource
POST	Used to create a new resource
PUT	Used to update an existing resource, overw
PATCH	Used for partial updates to an existing reso
DELETE	Used to delete an existing resource

Converted to HTML

MockMvc Test

src: http://www.javacodegeeks.com/2013/04/spring-mvc-introduction-in-testing.html

Stand Alone Setup

```
class ExampleControllerSpec extends Specification {

   protected MockMvc mockMvc

   void setup() {
      this.mockMvc = MockMvcBuilders.standaloneSetup(
         new ExampleController()).build()
   }

   void 'test and document get with example endpoint'() {

   when:
   ResultActions result = this.mockMvc.perform(
   get('/hello').contentType(MediaType.APPLICATION_JSON))

  then:
  result
     .andExpect(status().isOk())
     .andExpect(jsonPath("name").value("World"))
     .andExpect(jsonPath("message").value("Hello, World!”))
}
```

Web Context Setup

```
void setup() {
    this.mockMvc = MockMvcBuilders
        .webAppContextSetup(this.context)
        .build()
}
```

Web Context Setup

```
void setup() {
    this.mockMvc = MockMvcBuilders
        .webAppContextSetup(this.context)
        .build()
}
```

If context is null, remember to add spock-spring!!

Spring REST docs

Gradle

```
ext {
    snippetsDir = file('src/docs/generated-snippets')
}

testCompile “org.springframework.restdocs:spring-restdocs-mockmvc:1.0.1.RELEASE”
```

Gradle

```
asciidoctor {
    dependsOn test
    sourceDir = file('src/docs')
    outputDir "$projectDir/src/main/resources/public"
+   inputs.dir snippetsDir
    backends 'html5'
    attributes 'source-highlighter' : 'prettify',
            'imagesdir':'images',
            'toc':'left',
            'icons': 'font',
            'setanchors':'true',
            'idprefix':'',
            'idseparator':'-',
            'docinfo1':'true',
+           'snippets': snippetsDir
}
```

```
class ExampleControllerSpec extends Specification {

    @Rule
    RestDocumentation restDocumentation = new RestDocumentation('src/docs/generated-snippets')

    protected MockMvc mockMvc

    void setup() {
        this.mockMvc = MockMvcBuilders.standaloneSetup(new ExampleController())
                .apply(documentationConfiguration(this.restDocumentation))
                .build()
    }
```

```
class ExampleControllerSpec extends Specification {

    @Rule
    RestDocumentation restDocumentation = new RestDocumentation('src/docs/generated-snippets')

    protected MockMvc mockMvc

    void setup() {
        this.mockMvc = MockMvcBuilders.standaloneSetup(new ExampleController())
                .apply(documentationConfiguration(this.restDocumentation))
                .build()
    }
```

```
class ExampleControllerSpec extends Specification {

    @Rule
    RestDocumentation restDocumentation = new RestDocumentation('src/docs/generated-snippets')

    protected MockMvc mockMvc

    void setup() {
        this.mockMvc = MockMvcBuilders.standaloneSetup(new ExampleController())
                .apply(documentationConfiguration(this.restDocumentation))
                .build()
    }


    void 'test and document get with example endpoint'() {
        when:
        ResultActions result = this.mockMvc.perform(get('/hello')
                .contentType(MediaType.APPLICATION_JSON))
```

```
class ExampleControllerSpec extends Specification {

    @Rule
    RestDocumentation restDocumentation = new RestDocumentation('src/docs/generated-snippets')

    protected MockMvc mockMvc

    void setup() {
        this.mockMvc = MockMvcBuilders.standaloneSetup(new ExampleController())
                .apply(documentationConfiguration(this.restDocumentation))
                .build()
    }


    void 'test and document get with example endpoint'() {
        when:
        ResultActions result = this.mockMvc.perform(get('/hello')


        then:
        result
           .andExpect(status().isOk())
           .andExpect(jsonPath('name').value('World'))
           .andExpect(jsonPath('message').value('Hello, World!'))
           .andDo(document('hello-get-example', preprocessResponse(prettyPrint()),
           responseFields(
              fieldWithPath('name').type(JsonFieldType.STRING).description('name'),
              fieldWithPath('message').type(JsonFieldType.STRING).description('hello world'))
        ))
    }
}
```

```
class ExampleControllerSpec extends Specification {

    @Rule
    RestDocumentation restDocumentation = new RestDocumentation('src/docs/generated-snippets')

    protected MockMvc mockMvc

    void setup() {
        this.mockMvc = MockMvcBuilders.standaloneSetup(new ExampleController())
                .apply(documentationConfiguration(this.restDocumentation))
                .build()
    }


    void 'test and document get with example endpoint'() {
        when:
        ResultActions result = this.mockMvc.perform(get('/hello')


        then:
        result
           .andExpect(status().isOk())
           .andExpect(jsonPath('name').value('World'))
           .andExpect(jsonPath('message').value('Hello, World!'))
           .andDo(document('hello-get-example', preprocessResponse(prettyPrint()),
           responseFields(
              fieldWithPath('name').type(JsonFieldType.STRING).description('name'),
              fieldWithPath('message').type(JsonFieldType.STRING).description('hello world'))
        ))
    }
}
```

Default Example

A `GET` request with no parameters with return 'Hello, World!'

Example request

```
$ curl 'http://localhost:8080/hello' -i -H 'Content-Type: application/json'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 53

{
  "name" : "World",
  "message" : "Hello, World!"
}
```

Response structure

Path	Type	Description
name	String	name
message	String	message

POST

POST

```
void 'test and document post with example endpoint and custom name'() {
    when:
    ResultActions result = this.mockMvc.perform(post('/hello')
        .content(new ObjectMapper()
            .writeValueAsString(new Example(name: 'mockmvc test’))
         .contentType(MediaType.APPLICATION_JSON))
```

POST

```
void 'test and document post with example endpoint and custom name'() {
    when:
    ResultActions result = this.mockMvc.perform(post('/hello')
        .content(new ObjectMapper()
            .writeValueAsString(new Example(name: 'mockmvc test'))
         .contentType(MediaType.APPLICATION_JSON))
    then:
    result
        .andExpect(status().isOk())
        .andExpect(jsonPath('name').value('mockmvc test'))
        .andExpect(jsonPath('message').value('Hello, mockmvc test!'))
        .andDo(document('hello-post-example',
            preprocessResponse(prettyPrint()),
            responseFields(
                fieldWithPath('name').type(JsonFieldType.STRING)
                                .description('name'),
                fieldWithPath('message').type(JsonFieldType.STRING)
                                .description('hello world'))
        ))
}
```

Custom Name Example

A `POST` request with a name parameter will return 'Hello, $name!'

Example request

```
$ curl 'http://localhost:8080/hello' -i -X POST -H 'Content-Type: application/json' -d
'{"name":"mockmvc test","message":"Hello, mockmvc test!"}'
```

Example response

```http
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 67

{
  "name" : "mockmvc test",
  "message" : "Hello, mockmvc test!"
}
```

Response structure

Path	Type	Description
name	String	name
message	String	message

List Example

```
void 'test and document get of a list from example endpoint'() {
    when:
    ResultActions result = this.mockMvc.perform(get('/hello/list')
        .contentType(MediaType.APPLICATION_JSON))

    then:
    result
        .andExpect(status().isOk())
        .andDo(document('hello-list-example',
           preprocessResponse(prettyPrint()),
        responseFields(
            fieldWithPath('[].message').type(JsonFieldType.STRING)
                    .description('message'))
    ))
}
```

List Example

```
void 'test and document get of a list from example endpoint'() {
    when:
    ResultActions result = this.mockMvc.perform(get('/hello/list')
        .contentType(MediaType.APPLICATION_JSON))

    then:
    result
        .andExpect(status().isOk())
        .andDo(document('hello-list-example',
           preprocessResponse(prettyPrint()),
        responseFields(
            fieldWithPath('[].message').type(JsonFieldType.STRING)
                    .description('message'))
    ))
}
```

```
org.springframework.restdocs.snippet.SnippetException: The following parts of the payload were not documented:
{
  "name" : "mockmvc test"
}
Fields with the following paths were not found in the payload: [username]
        at org.springframework.restdocs.payload.AbstractFieldsSnippet.validateFieldDocumentation(AbstractFields
        at org.springframework.restdocs.payload.AbstractFieldsSnippet.createModel(AbstractFieldsSnippet.java:73
        at org.springframework.restdocs.snippet.TemplatedSnippet.document(TemplatedSnippet.java:64)
        at org.springframework.restdocs.mockmvc.RestDocumentationResultHandler.handle(RestDocumentationResultHa
        at org.springframework.test.web.servlet.MockMvc$1.andDo(MockMvc.java:177)
        at sample.web.ExampleControllerSpec.test and document post with example endpoint and custom name(Exampl
```

Failing Tests

```
org.springframework.restdocs.snippet.SnippetException: The following parts of the payload were not documented:
{
  "name" : "mockmvc test"
}
Fields with the following paths were not found in the payload: [username]
        at org.springframework.restdocs.payload.AbstractFieldsSnippet.validateFieldDocumentation(AbstractFields
        at org.springframework.restdocs.payload.AbstractFieldsSnippet.createModel(AbstractFieldsSnippet.java:73
        at org.springframework.restdocs.snippet.TemplatedSnippet.document(TemplatedSnippet.java:64)
        at org.springframework.restdocs.mockmvc.RestDocumentationResultHandler.handle(RestDocumentationResultHa
        at org.springframework.test.web.servlet.MockMvc$1.andDo(MockMvc.java:177)
        at sample.web.ExampleControllerSpec.test and document post with example endpoint and custom name(Exampl
```

```
g.springframework.restdocs.snippet.SnippetException: Fields with the following paths were not found in the payload: [[].name]
      at org.springframework.restdocs.payload.AbstractFieldsSnippet.validateFieldDocumentation(AbstractFieldsSnippet.java:133)
      at org.springframework.restdocs.payload.AbstractFieldsSnippet.createModel(AbstractFieldsSnippet.java:73)
      at org.springframework.restdocs.snippet.TemplatedSnippet.document(TemplatedSnippet.java:64)
      at org.springframework.restdocs.mockmvc.RestDocumentationResultHandler.handle(RestDocumentationResultHandler.java:101)
      at org.springframework.test.web.servlet.MockMvc$1.andDo(MockMvc.java:177)
      at sample.web.ExampleControllerSpec.test and document get of a list from example endpoint(ExampleControllerSpec.groovy:7
```

Failing Tests

Generated Snippets

{example-name}

- curl-request.adoc
- http-request.adoc
- httpie-request.adoc
- http-response.adoc
- response-fields.adoc
- request-parameters.adoc
- request-parts.adoc

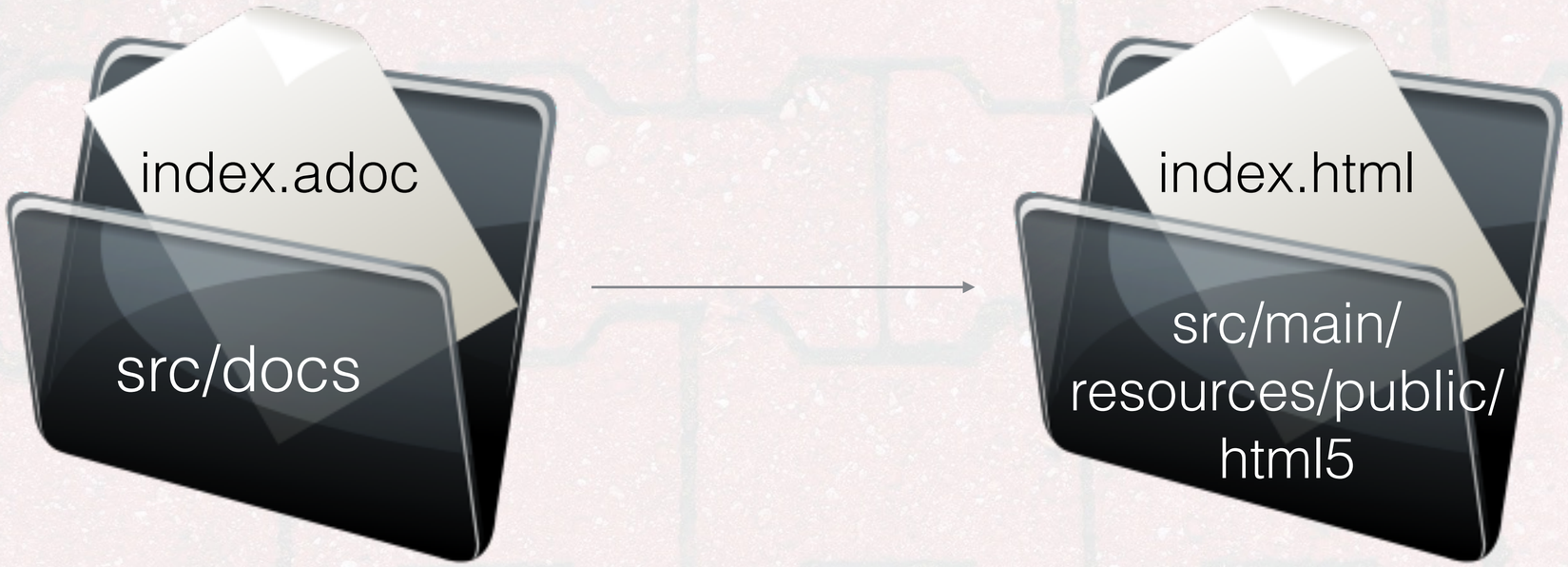
Generated Snippets

{example-name}

- curl-request.adoc
- http-request.adoc
- NEW httpie-request.adoc
- http-response.adoc
- response-fields.adoc
- request-parameters.adoc
- NEW request-parts.adoc

Add Snippets to AsciiDoc

```
== Errors

Whenever an error response (status code >= 400) is returned, the
body will contain a JSON object
that describes the problem. The error object has the following
structure:

include::{snippets}/error-example/response-fields.adoc[]

For example, a request that attempts to apply a non-existent tag
to a note will produce a
`400 Bad Request` response:

include::{snippets}/error-example/http-response.adoc[]

[[resources]]
= Resources

include::resources/example.adoc[]
```


index.adoc
src/docs
index.html
src/main/
resources/public/
html5

Build Docs

Table of Contents

- Introduction
  - HTTP verbs
  - HTTP status codes
  - Errors
- Resources
  - Hello
    - Default Example
      - Example request
      - Example response
      - Response structure
    - Custom Name Example
      - Example request
      - Example response
      - Response structure

Example API Guide

Jenn Strater

Introduction

The Example API is a RESTful web service that shows how Spring REST docs works.

HTTP verbs

The Example API tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs.

Verb	Usage
`GET`	Used to retrieve a resource
`POST`	Used to create a new resource
`PUT`	Used to update an existing resource, overwrites all fields

http://jlstrater.github.io/groovy-spring-boot-restdocs-example

Support for Rest Assured

OBJECT PARTNERS

@codeJENNerator

img src: https://www.flickr.com/photos/dskley/15558668690

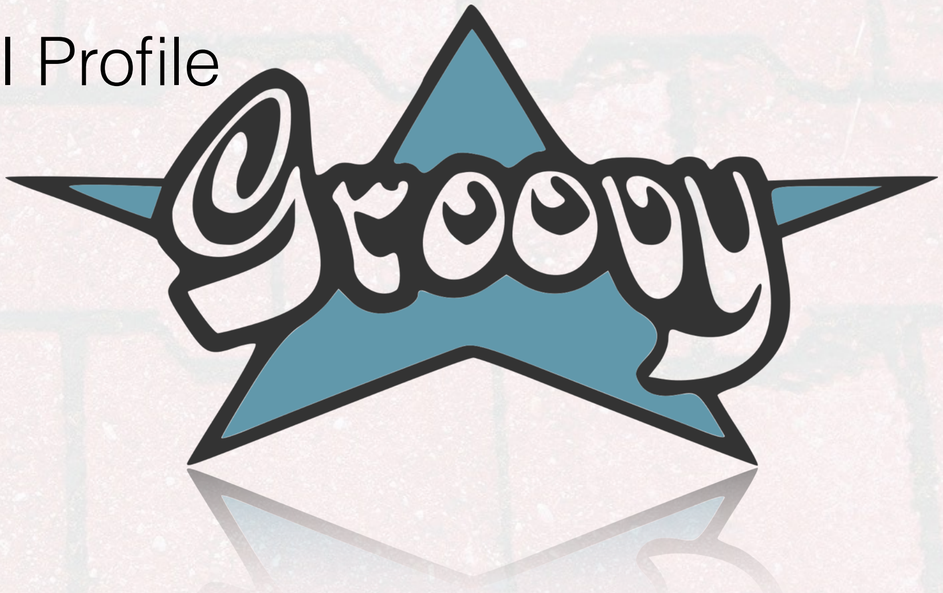
Groovier Spring REST docs

- Grails
- Ratpack

Groovier Spring REST docs Example II

Groovier Spring REST docs Example II

- Grails 3.0.15 with Web API Profile
  - Level II Rest API



Groovier Spring REST docs Example II

- Grails 3.0.15 with Web API Profile
  - Level II Rest API

+ Asciidoctor Gradle plugin

Groovier Spring REST docs Example II

- Grails 3.0.15 with Web API Profile
  - Level II Rest API

+ Asciidoctor Gradle plugin

+ Spring REST docs 1.1.0.RELEASE - **RestAssured**

Groovier Spring REST docs Example II

- Grails 3.0.15 with Web API Profile
  - Level II Rest API

+ Asciidoctor Gradle plugin

+ Spring REST docs 1.1.0.RELEASE - **RestAssured**

+ Publish to Github Pages

Simple Grails App

Simple Grails App

```
->grails create-app –profile=web-api –inplace

grails> create-domain-resource com.example.Note
| Created grails-app/domain/com/example/Note.groovy
| Created src/test/groovy/com/example/NoteSpec.groovy

grails> create-domain-resource com.example.Tag
| Created grails-app/domain/com/example/Tag.groovy
| Created src/test/groovy/com/example/TagSpec.groovy
```

```
package com.example

import grails.rest.Resource

+@Resource(uri='/notes', readOnly = false, formats = ['json', 'xml'])
class Note {
+    Long id
+    String title
+    String body

+    static hasMany = [tags: Tag]
+    static mapping = {
+        tags joinTable: [name: "mm_notes_tags", key: 'mm_note_id' ]
+    }
}
```

```
package com.example

import grails.rest.Resource

+@Resource(uri='/tags', readOnly = false, formats = ['json', 'xml'])
class Tag {
+    Long id
+    String name

+    static hasMany = [notes: Note]
+    static belongsTo = Note
+    static mapping = {
+        notes joinTable: [name: "mm_notes_tags", key: 'mm_tag_id']
+    }
}
```

Asciidoctor Gradle Plugin

Spring REST docs
&
REST-assured

https://github.com/jayway/rest-assured

Grails Specific Setup

```
+ testCompile “org.springframework.restdocs:spring-restdocs-
restassured:1.1.0.RELEASE”

asciidoctor {
  …
+ mustRunAfter integrationTest
  …
}
```

```
@Integration
@Rollback
class ApiDocumentationSpec extends Specification {
    @Rule
    JUnitRestDocumentation restDocumentation =
       new JUnitRestDocumentation('src/docs/generated-snippets')

    protected RequestSpecification documentationSpec

    void setup() {
        this.documentationSpec = new RequestSpecBuilder()
                .addFilter(documentationConfiguration(restDocumentation))
                .build()
    }
}
```

```
@Integration
@Rollback
class ApiDocumentationSpec extends Specification {
    @Rule
    JUnitRestDocumentation restDocumentation =
       new JUnitRestDocumentation('src/docs/generated-snippets')

    protected RequestSpecification documentationSpec

    void setup() {
        this.documentationSpec = new RequestSpecBuilder()
                .addFilter(documentationConfiguration(restDocumentation))
                .build()
    }
}
```

```
@Integration
@Rollback
class ApiDocumentationSpec extends Specification {
    @Rule
    JUnitRestDocumentation restDocumentation =
       new JUnitRestDocumentation('src/docs/generated-snippets')

    protected RequestSpecification documentationSpec

    void setup() {
        this.documentationSpec = new RequestSpecBuilder()
                .addFilter(documentationConfiguration(restDocumentation))
                .build()
    }
}
```

```
void 'test and document notes list request'() {
    expect:
    given(this.documentationSpec)
            .accept(MediaType.APPLICATION_JSON.toString())
            .filter(document('notes-list-example',
            preprocessRequest(modifyUris()
                    .host('api.example.com')
                    .removePort()),
            preprocessResponse(prettyPrint()),
            responseFields(
                fieldWithPath('[].class').description('the class of the resource'),
                fieldWithPath('[].id').description('the id of the note'),
                fieldWithPath('[].title').description('the title of the note'),
                fieldWithPath('[].body').description('the body of the note'),
                fieldWithPath('[].tags').type(JsonFieldType.ARRAY)
                    .description('the list of tags associated with the note'),
            )))
            .when()
            .port(8080)
            .get('/notes')
            .then()
            .assertThat()
            .statusCode(is(200))
}
```

```
void 'test and document notes list request'() {
    expect:
    given(this.documentationSpec)
            .accept(MediaType.APPLICATION_JSON.toString())
            .filter(document('notes-list-example',
            preprocessRequest(modifyUris()
                    .host('api.example.com')
                    .removePort()),
            preprocessResponse(prettyPrint()),
            responseFields(
                fieldWithPath('[].class').description('the class of the resource'),
                fieldWithPath('[].id').description('the id of the note'),
                fieldWithPath('[].title').description('the title of the note'),
                fieldWithPath('[].body').description('the body of the note'),
                fieldWithPath('[].tags').type(JsonFieldType.ARRAY)
                    .description('the list of tags associated with the note'),
            )))
            .when()
            .port(8080)
            .get('/notes')
            .then()
            .assertThat()
            .statusCode(is(200))
}
```

```
void 'test and document create new note'() {
    expect:
    given(this.documentationSpec)
            .accept(MediaType.APPLICATION_JSON.toString())
            .contentType(MediaType.APPLICATION_JSON.toString())
            .filter(document('notes-create-example',
            preprocessRequest(modifyUris()
                    .host('api.example.com')
                    .removePort()),
            preprocessResponse(prettyPrint()),
            requestFields(
                    fieldWithPath('title').description('the title of the note'),
                    fieldWithPath('body').description('the body of the note'),
                    fieldWithPath('tags').type(JsonFieldType.ARRAY)
                       .description('a list of tags associated to the note')
            ),
            responseFields(
                fieldWithPath('class').description('the class of the resource'),
                fieldWithPath('id').description('the id of the note'),
                fieldWithPath('title').description('the title of the note'),
                fieldWithPath('body').description('the body of the note'),
                fieldWithPath('tags').type(JsonFieldType.ARRAY)
                   .description('the list of tags associated with the note'),
            )))
            .body('{ "body": "My test example", "title": "Eureka!",
                     "tags": [{"name": "testing123"}] }')
            .when()
            .port(8080)
            .post('/notes')
            .then()
            .assertThat()
            .statusCode(is(201))
}
```

```
void 'test and document create new note'() {
    expect:
    given(this.documentationSpec)
            .accept(MediaType.APPLICATION_JSON.toString())
            .contentType(MediaType.APPLICATION_JSON.toString())
            .filter(document('notes-create-example',
            preprocessRequest(modifyUris()
                    .host('api.example.com')
                    .removePort()),
            preprocessResponse(prettyPrint()),
            requestFields(
                    fieldWithPath('title').description('the title of the note'),
                    fieldWithPath('body').description('the body of the note'),
                    fieldWithPath('tags').type(JsonFieldType.ARRAY)
                       .description('a list of tags associated to the note')
            ),
            responseFields(
                fieldWithPath('class').description('the class of the resource'),
                fieldWithPath('id').description('the id of the note'),
                fieldWithPath('title').description('the title of the note'),
                fieldWithPath('body').description('the body of the note'),
                fieldWithPath('tags').type(JsonFieldType.ARRAY)
                   .description('the list of tags associated with the note'),
            )))
            .body('{ "body": "My test example", "title": "Eureka!",
                     "tags": [{"name": "testing123"}] }')
            .when()
            .port(8080)
            .post('/notes')
            .then()
            .assertThat()
            .statusCode(is(201))
}
```

Publish Docs

publish.gradle

```
buildscript {
    repositories {
        jcenter()
    }

    dependencies {
        classpath 'org.ajoberstar:gradle-git:1.1.0'
    }
}

apply plugin: 'org.ajoberstar.github-pages'

githubPages {
    repoUri = 'git@github.com:jlstrater/grails-spring-boot-restdocs-example.git'
    pages {
        from(file('build/docs'))
    }
}
```

Table of Contents

Overview
HTTP verbs
HTTP status codes
Resources
Index
Accessing the index
Example request
Response structure
Example response
Notes
Listing notes
Response structure
Example request
Example response
Creating a note
Request structure
Example request
Example response
Retrieve a note
Response structure
Example request
Example response

Creating a note

A `POST` request is used to create a note

Request structure

Path	Type	Description
title	String	the title of the note
body	String	the body of the note
tags	Array	a list of tags associated to the note

Example request

```
$ curl 'http://api.example.com/notes' -i -X POST -H 'Accept: application/json' -H 'Content-Type:
application/json; charset=UTF-8' -d '{ "body": "My test example", "title": "Eureka!", "tags":
[{"name": "testing123"}] }'
```

Example response

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
X-Application-Context: application:test
```

https://jlstrater.github.io/grails-spring-restdocs-example

Groovier Spring REST docs Example III

- Ratpack Example Project
  - https://github.com/ratpack/example-books
- Spring RESTdocs **RestAssured**
  - https://github.com/ratpack/example-books/pull/25

restdocs.gradle

```
dependencies {
    testCompile ‘org.springframework.restdocs:spring-restdocs-restassured:1.1.1.RELEASE'
}

ext {
    snippetsDir = file('src/docs/generated-snippets')
}

task cleanTempDirs(type: Delete) {
    delete fileTree(dir: 'src/docs/generated-snippets')
    delete fileTree(dir: 'src/ratpack/public/docs')
}

test {
    dependsOn cleanTempDirs
    outputs.dir snippetsDir
}

asciidoctor {
    mustRunAfter test
    inputs.dir snippetsDir
    sourceDir = file('src/docs')
    separateOutputDirs = false
    outputDir "$projectDir/src/ratpack/public/docs"
    attributes 'snippets': snippetsDir,
            'source-highlighter': 'prettify',
            'imagesdir': 'images',
            'toc': 'left',
            'icons': 'font',
            'setanchors': 'true',
            'idprefix': '',
            'idseparator': '-',
            'docinfo1': 'true'
}

build.dependsOn asciidoctor
```

build.gradle

```
plugins {
    id "io.ratpack.ratpack-groovy" version "1.2.0"
    .
    .
    .
+   id 'org.asciidoctor.convert' version '1.5.3'
}

repositories {
    jcenter()
   .
   .
   .
+    maven { url 'https://repo.spring.io/libs-snapshot' }
}


//some CI config
apply from: "gradle/ci.gradle"
+ apply from: "gradle/restdocs.gradle"
```

```
path(":isbn") {
    def isbn = pathTokens["isbn"]

    byMethod {
        get {
            bookService.find(isbn).
                single().
                subscribe { Book book ->
                    if (book == null) {
                        clientError 404
                    } else {
                        render book
                    }
                }
        }
        ...
    }
}
```

```
byMethod {
   get {
      bookService.all().
         toList().
            subscribe { List<Book> books ->
               render json(books)
            }
   }
   post {
      parse(jsonNode()).
         observe().
         flatMap { input ->
            bookService.insert(

               input.get("isbn").asText(),

               input.get("quantity").asLong(),

               input.get("price").asDouble()
            )
         }.
         single().
         flatMap {
            bookService.find(it)
         }.
         single().
         subscribe { Book createdBook ->
            render createdBook
         }
    }
}
```

```
path(":isbn") {
    def isbn = pathTokens["isbn"]

    byMethod {
        get {
            bookService.find(isbn).
                single()
                                    ->
                                    {
                    } else {
                        render book
                    }
                }
        }
        ...
    }
}
```

Get a book by ISBN
/api/books/1932394842

```
byMethod {
   get {
      bookService.all().
         toList().
            subscribe { List<Book> books ->
               render json(books)
            }
   }
   post {
      parse(jsonNode()).
         observe().
         flatMap { input ->
            bookService.insert(

               input.get("isbn").asText(),

               input.get("quantity").asLong(),

               input.get("price").asDouble()
            )
         }.
         single().
         flatMap {
            bookService.find(it)
         }.
         single().
         subscribe { Book createdBook ->
            render createdBook
         }
    }
}
```

```
path(":isbn") {
    def isbn = pathTokens["isbn"]

    byMethod {
        get {
            bookService.find(isbn).
                single()
                                    ->
                    } else {
                        render book
                    }
                }
        }
        ...
    }
}
```

Get a book by ISBN
/api/books/1932394842

```
byMethod {
    get {
        bookSe
            toL
                                    >
                }
    }
    post {
        parse(jsonNode()).
            observe().
            flatMap { input ->
                bookService.insert(

                    input.get("isbn").asText(),

                    input.get("quantity").asLong(),

                    input.get("price").asDouble()
                )
            }.
            single().
            flatMap {
                bookService.find(it)
            }.
            single().
            subscribe { Book createdBook ->
                render createdBook
            }
    }
}
```

Get all books
/api/books

```
path(":isbn") {
    def isbn = pathTokens["isbn"]

    byMethod {
        get {
            bookService.find(isbn).
                single()
                    ->
                    } else {
                        render book
                    }
                }
        }
        ...
    }
}
```

Get a book by ISBN
/api/books/1932394842

```
byMethod {
    get {
        bookSe
            toL
                >
            }
    }
    post {
        parse(jsonNode()).
            observe().
            flatMap { input ->
                bookService.insert(
                    ),
                    input.get("price").asDouble()
                )
            }.
            single().
            flatMap {
                bookService.find(it)
            }.
            single().
            subscribe { Book createdBook ->
                render createdBook
            }
    }
}
```

Get all books
/api/books

Post to create a new book
/api/books

```java
import static org.springframework.restdocs.restassured.RestAssuredRestDocumentation.documentationConfiguration

import com.jayway.restassured.builder.RequestSpecBuilder
import com.jayway.restassured.specification.RequestSpecification
import org.junit.Rule
import org.springframework.restdocs.JUnitRestDocumentation
import ratpack.examples.book.fixture.ExampleBooksApplicationUnderTest
import ratpack.test.ApplicationUnderTest
import spock.lang.Shared
import spock.lang.Specification

abstract class BaseDocumentationSpec extends Specification {

    @Shared
    ApplicationUnderTest aut = new ExampleBooksApplicationUnderTest()

    @Rule
    JUnitRestDocumentation restDocumentation = new JUnitRestDocumentation('src/docs/generated-snippets')

    protected RequestSpecification documentationSpec

    void setup() {
        this.documentationSpec = new RequestSpecBuilder()
                .addFilter(documentationConfiguration(restDocumentation))
                .build()
    }
}
```

```
class BookDocumentationSpec extends BaseDocumentationSpec {

    @Shared
    EmbeddedApp isbndb = GroovyEmbeddedApp.of {
        handlers {
            all {
                render '{"data" : [{"title" : "Learning Ratpack", "publisher_name" : "O\'Reilly Media", "author_data" : [{"id" :
"dan_woods", "name" : "Dan Woods"}]}]}'
            }
        }
    }

    @Delegate
    TestHttpClient client = aut.httpClient
    RemoteControl remote = new RemoteControl(aut)


    def setupSpec() {
        System.setProperty('eb.isbndb.host', "http://${isbndb.address.host}:${isbndb.address.port}")
        System.setProperty('eb.isbndb.apikey', "fakeapikey")
    }

    def cleanupSpec() {
        System.clearProperty('eb.isbndb.host')
    }

    def setupTestBook() {
        requestSpec { RequestSpec requestSpec ->
            requestSpec.body.type("application/json")
            requestSpec.body.text(JsonOutput.toJson([isbn: "1932394842", quantity: 0, price: 22.34]))
        }
        post("api/book")
    }

    def cleanup() {
        remote.exec {
            get(Sql).execute("delete from books")
        }
    }
.
.
.
```

```
void 'test and document list books'() {
    setup:
    setupTestBook()

    expect:
    given(this.documentationSpec)
            .contentType('application/json')
            .accept('application/json')
            .port(aut.address.port)
            .filter(document('books-list-example',
            preprocessRequest(modifyUris()
                    .host('books.example.com')
                    .removePort()),
            preprocessResponse(prettyPrint()),
            responseFields(
                    fieldWithPath('[].isbn').description('The ISBN of the book'),
                    fieldWithPath('[].quantity').description("The quantity of the book that is available"),
                    fieldWithPath('[].price').description("The current price of the book"),
                    fieldWithPath('[].title').description("The title of the book"),
                    fieldWithPath('[].author').description('The author of the book'),
                    fieldWithPath('[].publisher').description('The publisher of the book')
            )))
            .when()
            .get('/api/book')
            .then()
            .assertThat()
            .statusCode(is(200))
}
```

Reusable Snippets

```
protected final Snippet getBookFields() {
    responseFields(
       fieldWithPath('isbn').description('The ISBN of the book'),
       fieldWithPath('quantity').description("The quantity of the book that is available"),
       fieldWithPath('price').description("The current price of the book"),
       fieldWithPath('title').description("The title of the book"),
       fieldWithPath('author').description('The author of the book’),
       fieldWithPath('publisher').description('The publisher of the book’)
    )
}
```

```
def "test and document create book"() {
    given:
    def setup = given(this.documentationSpec)
            .body('{"isbn": "1234567890", "quantity": 10, "price": 22.34}')
            .contentType('application/json')
            .accept('application/json')
            .port(aut.address.port)
            .filter(document('books-create-example',
            preprocessRequest(prettyPrint(),
                    modifyUris()
                            .host('books.example.com')
                            .removePort()),
            preprocessResponse(prettyPrint()),
            bookFields,
            requestFields(
                    fieldWithPath('isbn').type(JsonFieldType.STRING).description('book ISBN id'),
                    fieldWithPath('quantity').type(JsonFieldType.NUMBER).description('quanity available'),
                    fieldWithPath('price').type(JsonFieldType.NUMBER)
                            .description('price of the item as a number without currency')
            ),))
    when:
    def result = setup
            .when()
            .post("api/book")
    then:
    result
            .then()
            .assertThat()
            .statusCode(is(200))
}
```

```
def "test and document create book"() {
    given:
    def setup = given(this.documentationSpec)
            .body('{"isbn": "1234567890", "quantity": 10, "price": 22.34}')
            .contentType('application/json')
            .accept('application/json')
            .port(aut.address.port)
            .filter(document('books-create-example',
            preprocessRequest(prettyPrint(),
                    modifyUris()
                            .host('books.example.com')
                            .removePort()),
            preprocessResponse(prettyPrint()),
            bookFields,
            requestFields(
                    fieldWithPath('isbn').type(JsonFieldType.STRING).description('book ISBN id'),
                    fieldWithPath('quantity').type(JsonFieldType.NUMBER).description('quanity available'),
                    fieldWithPath('price').type(JsonFieldType.NUMBER)
                            .description('price of the item as a number without currency')
            ),))
    when:
    def result = setup
            .when()
            .post("api/book")
    then:
    result
            .then()
            .assertThat()
            .statusCode(is(200))
}
```

```
void 'test and document get individual book'() {
    setup:
    setupTestBook()

    expect:
    given(this.documentationSpec)
            .contentType('application/json')
            .accept('application/json')
            .port(aut.address.port)
            .filter(document('books-get-example',
            preprocessRequest(modifyUris()
                    .host('books.example.com')
                    .removePort()),
            preprocessResponse(prettyPrint()),
            bookFields))
            .when()
            .get("/api/book/1932394842")
            .then()
            .assertThat()
            .statusCode(is(200))
}
```

Read the docs for more on..

- Adding Security and Headers
- Documenting Constraints
- Hypermedia Support
- Using Markdown instead of Asciidoc

http://projects.spring.io/spring-restdocs/

Conclusion

- API documentation is complex

- API documentation is complex
- Choosing the right tool for the job not just about the easiest one to setup

- API documentation is complex
- Choosing the right tool for the job not just about the easiest one to setup
- Spring REST Docs is a promising tool to enforce good testing and documentation practices without muddying source code.

- API documentation is complex
- Choosing the right tool for the job not just about the easiest one to setup
- Spring REST Docs is a promising tool to enforce good testing and documentation practices without muddying source code.
- I still hate writing documentation, but at least it's a little less painful now.

Questions?

https://flic.kr/p/5DeuzB

https://github.com/jlstrater/groovy-spring-boot-restdocs-example

https://github.com/jlstrater/grails-spring-restdocs-example

https://github.com/ratpack/example-books