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

Jenn Strater
@codeJENNerator

Note For Those Viewing Slides Online

- Bulleted text like this indicates the key points mentioned on a previous slide. They may not have been included in the official presentation.

- This view does not support links, but the links will work in the pdf. Click the ‘download pdf’ button on the right.

Follow Along

https://speakerdeck.com/jlstrater/test-driven-docs-jfokus-2017

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

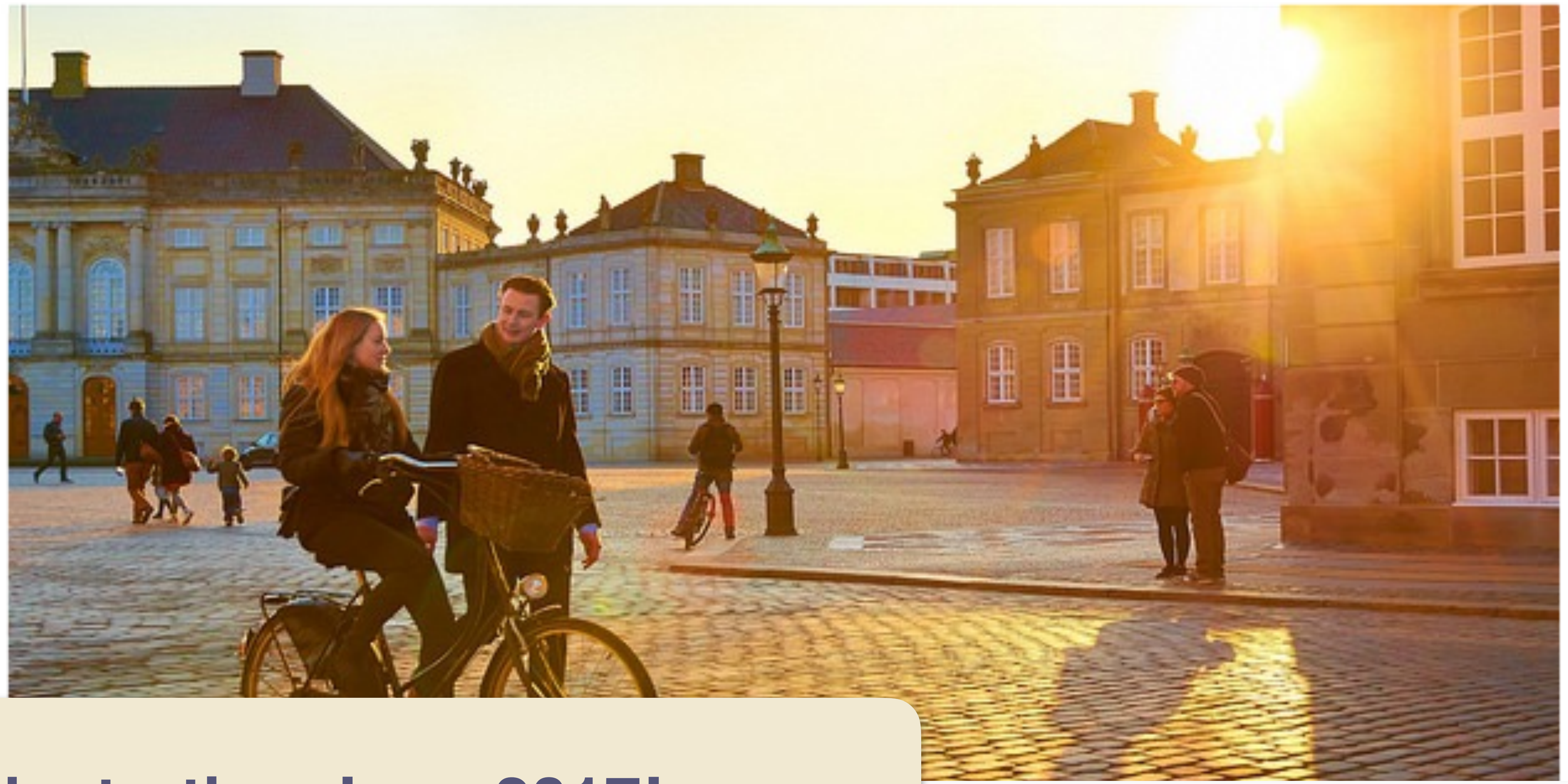
https://github.com/jlstrater/grails-spring-restdocs-example

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

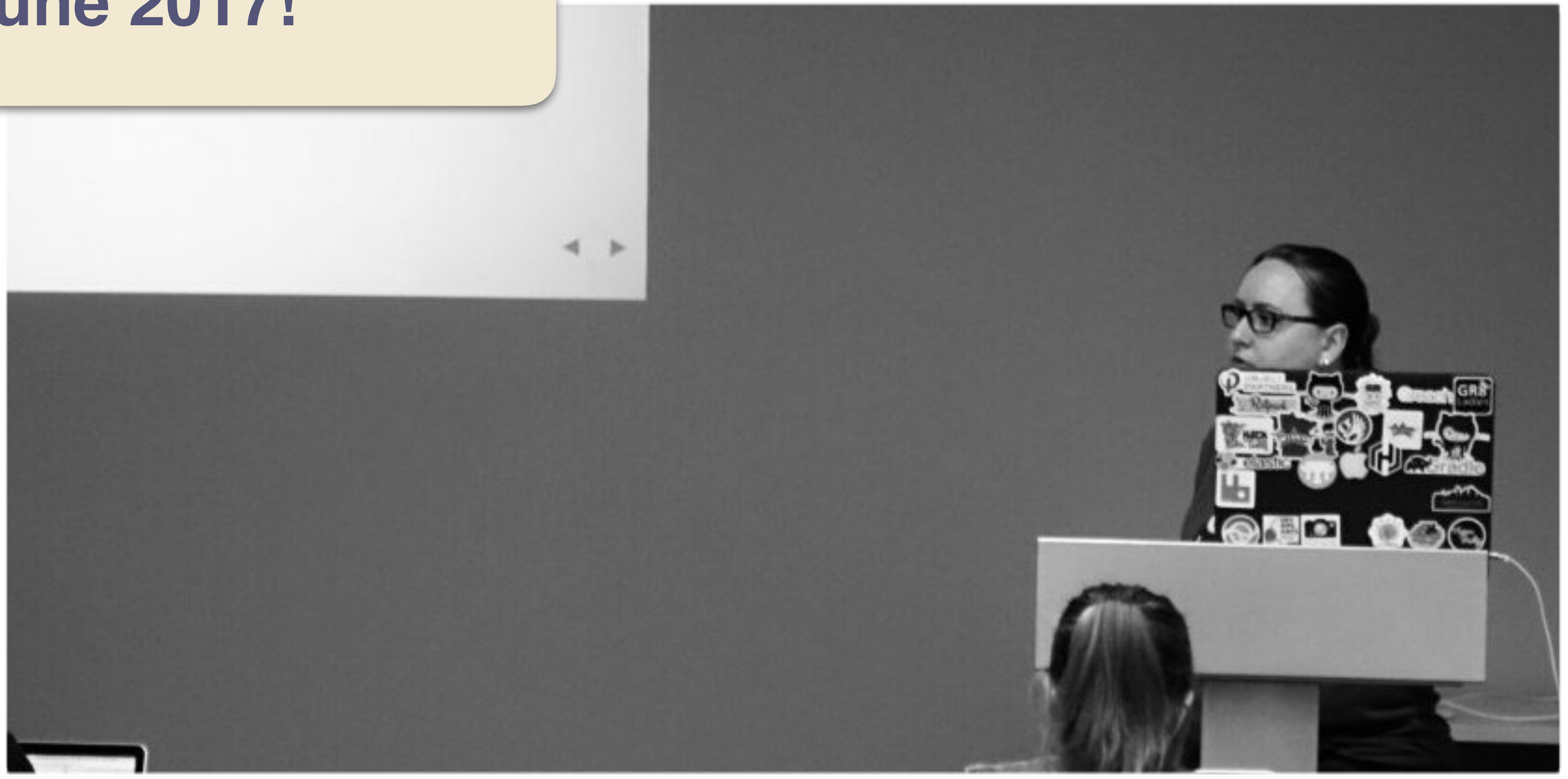
About Me

@codeJENNerator

About Me



Looking for work starting June 2017!



@codeJENNerator

About Me

- Taking classes at the Technical University of Denmark and working on a research project
- Also exploring Danish Culture with funding from the US Fulbright Grant program
- Prior to the Fulbright Grant, I was a senior consultant at Object Partners, Inc. in Minneapolis, MN, USA. My work there is the subject of this talk.
- Co-founder of Gr8Ladies and talk about women in the Groovy Community all over the world
- Passionate about bring new people into the Groovy community through free introductory workshops called Gr8Workshops.
- Looking for work starting June 2017 or later

Background

Background

- Creating RESTful APIs
  - Spring Boot
  - Grails
  - Ratpack

Background

- Creating RESTful APIs
  - Spring Boot
  - Grails
  - Ratpack

- Documentation
  - Swagger
  - Asciidoctor

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

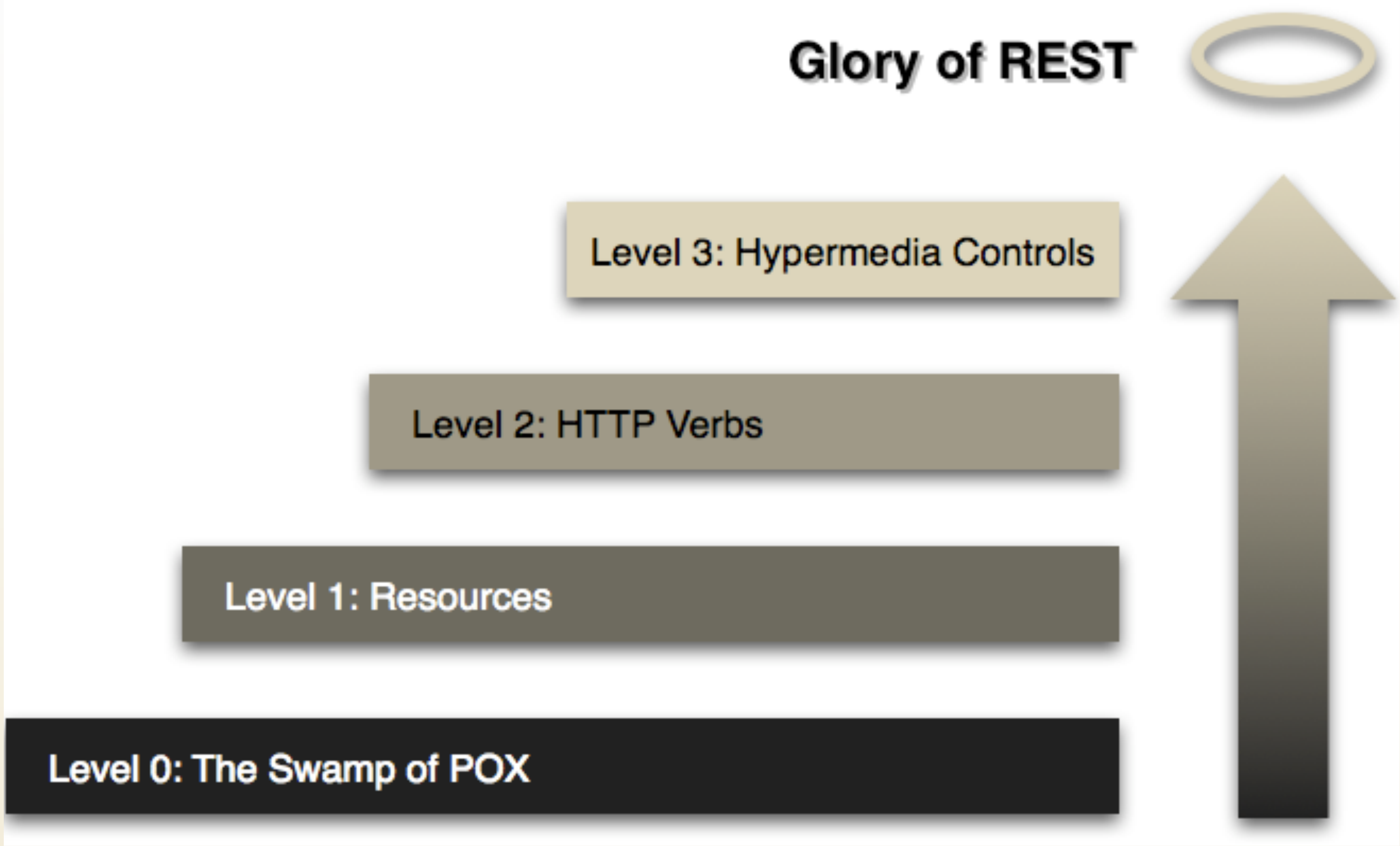
@codeJENNerator

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

I hate writing documentation!

@codeJENNerator

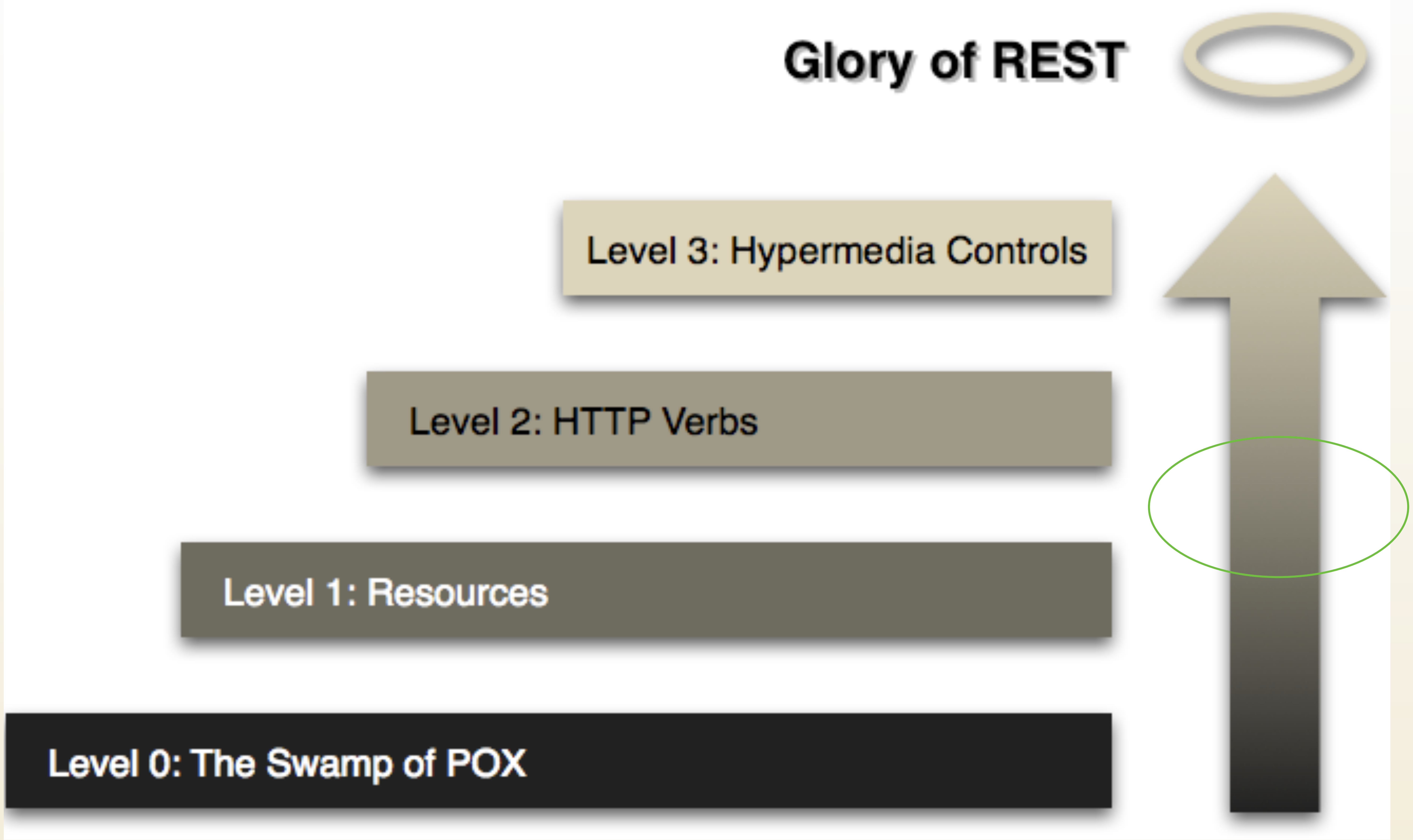
Factors in Choosing a Documentation Solution



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

REST Maturity Model

@codeJENNerator

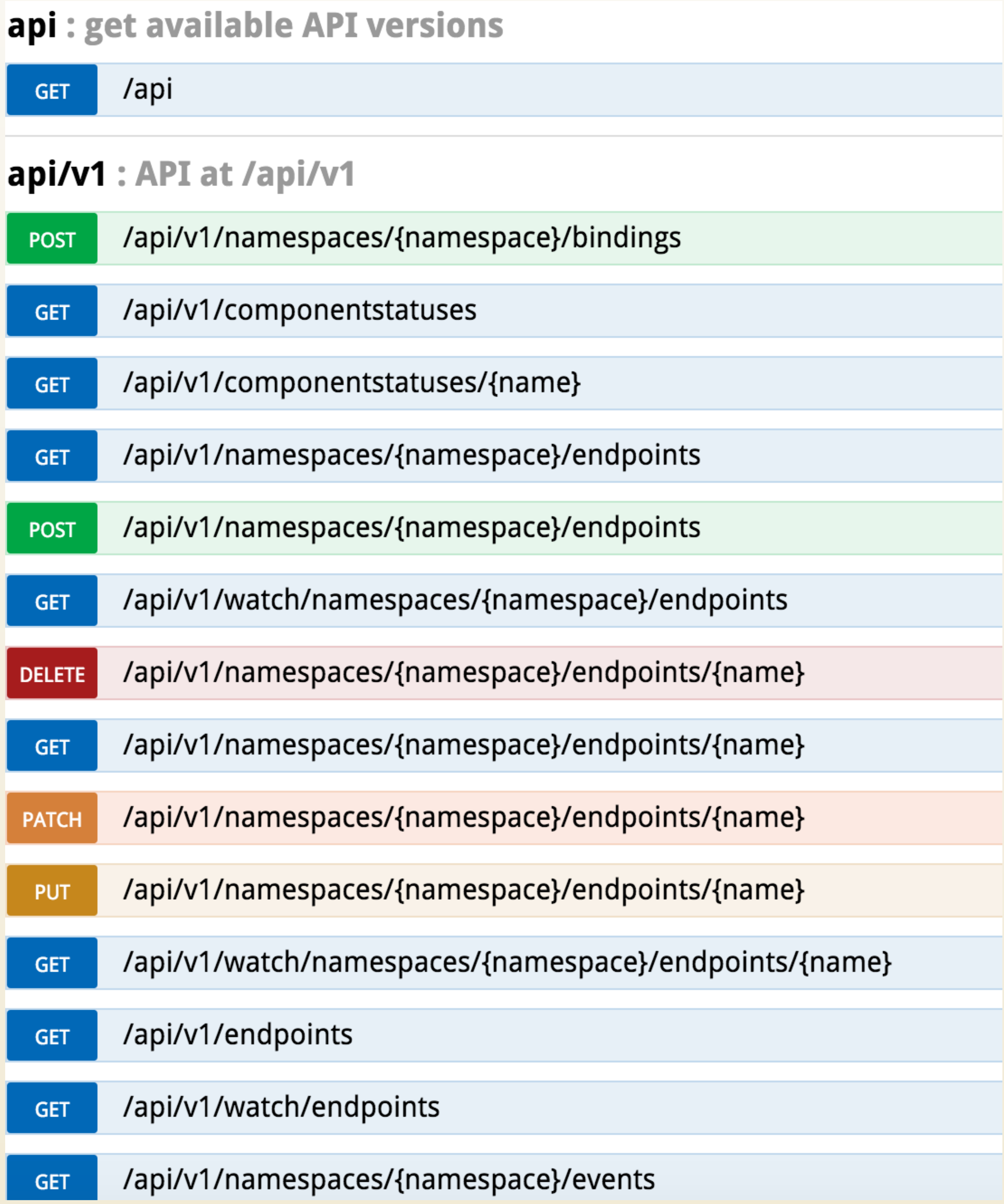


img src: http://martinfowler.com/articles/richardsonMaturityModel.html

REST Maturity Model

@codeJENNerator

Endpoint Vs Resource Design



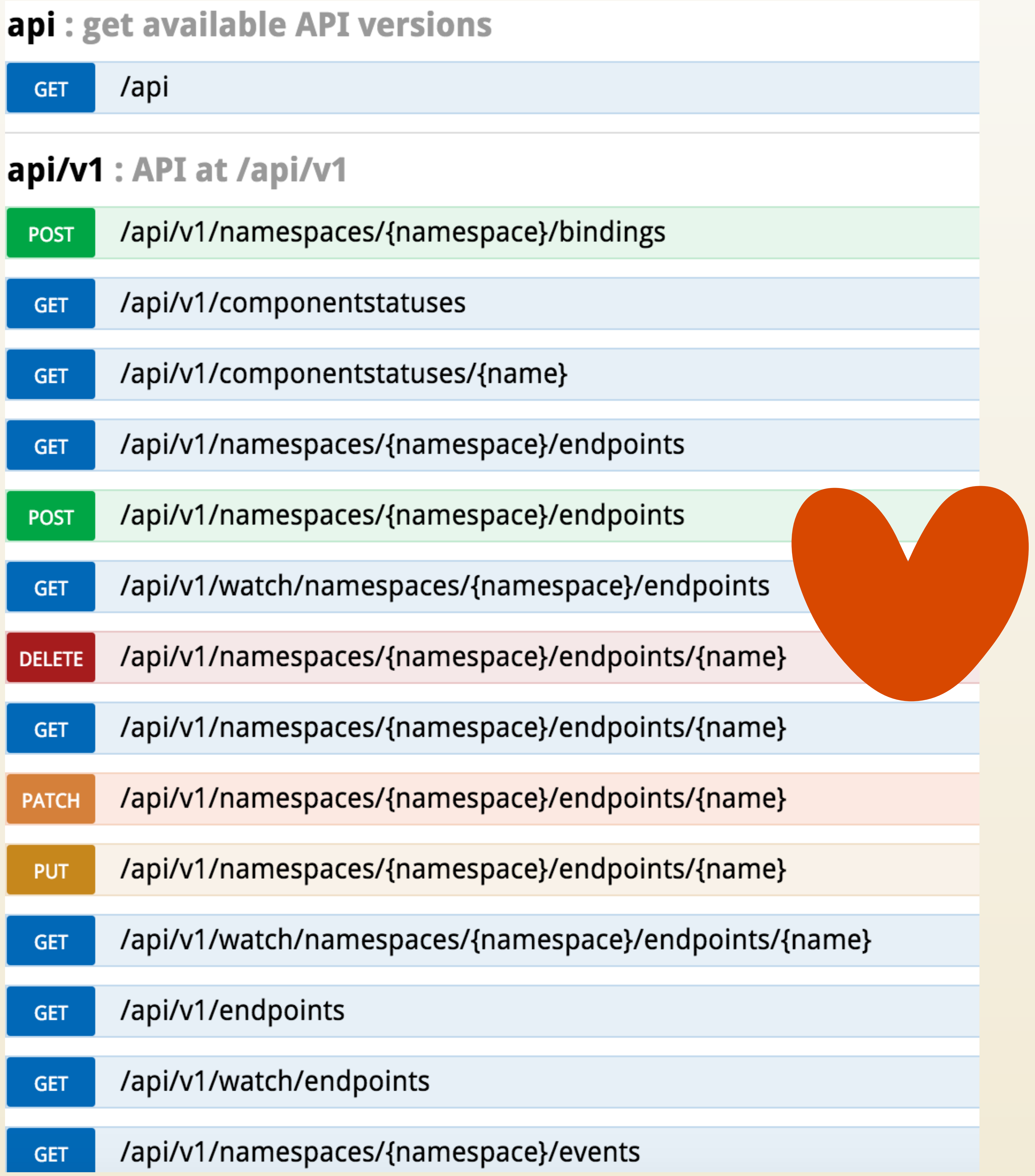
api : get available API versions

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

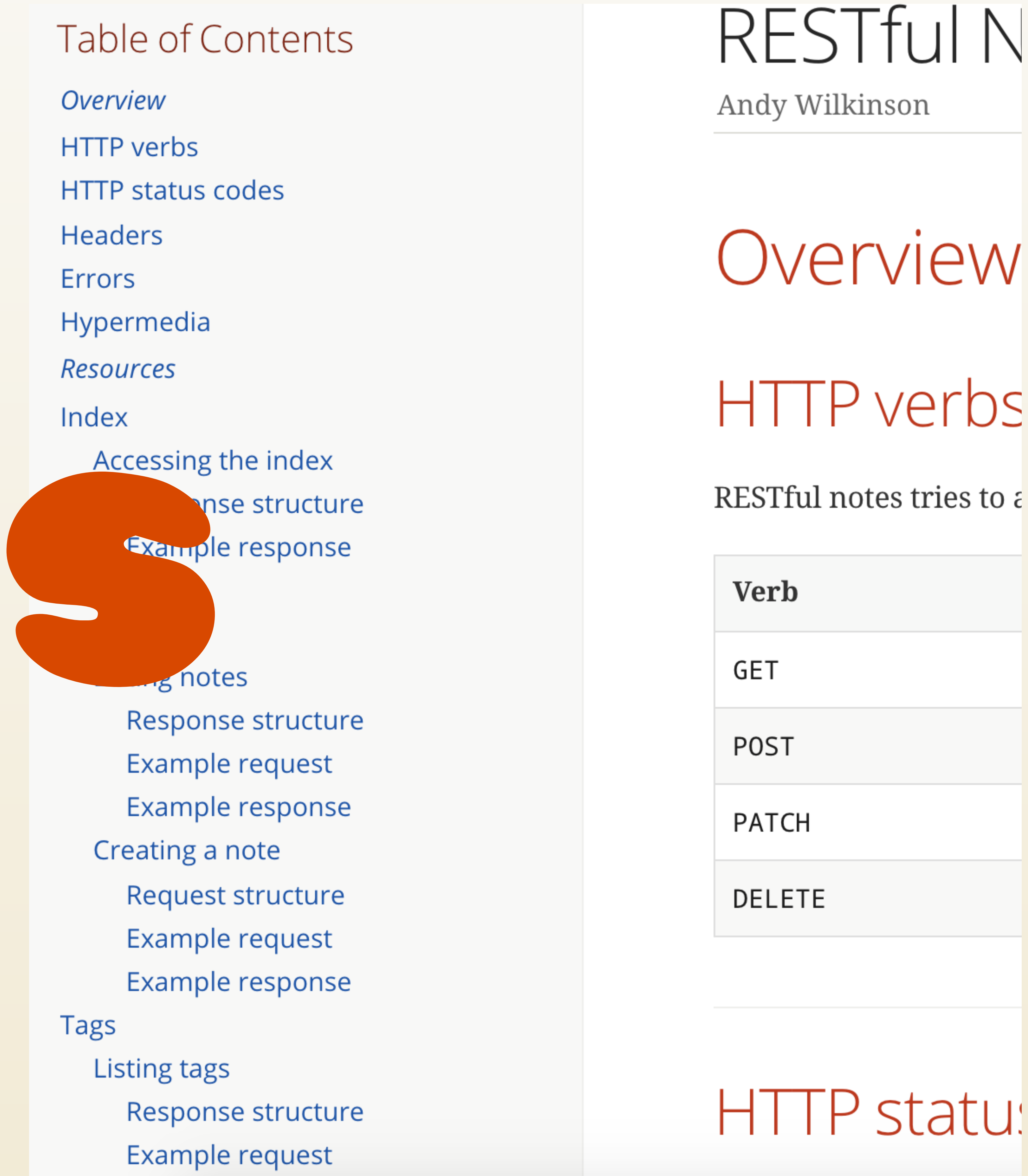


Table of Contents

Overview

HTTP verbs

HTTP status codes

Headers

Errors

Hypermedia

Resources

Index

Accessing the index

nse structure

Example response

ng 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

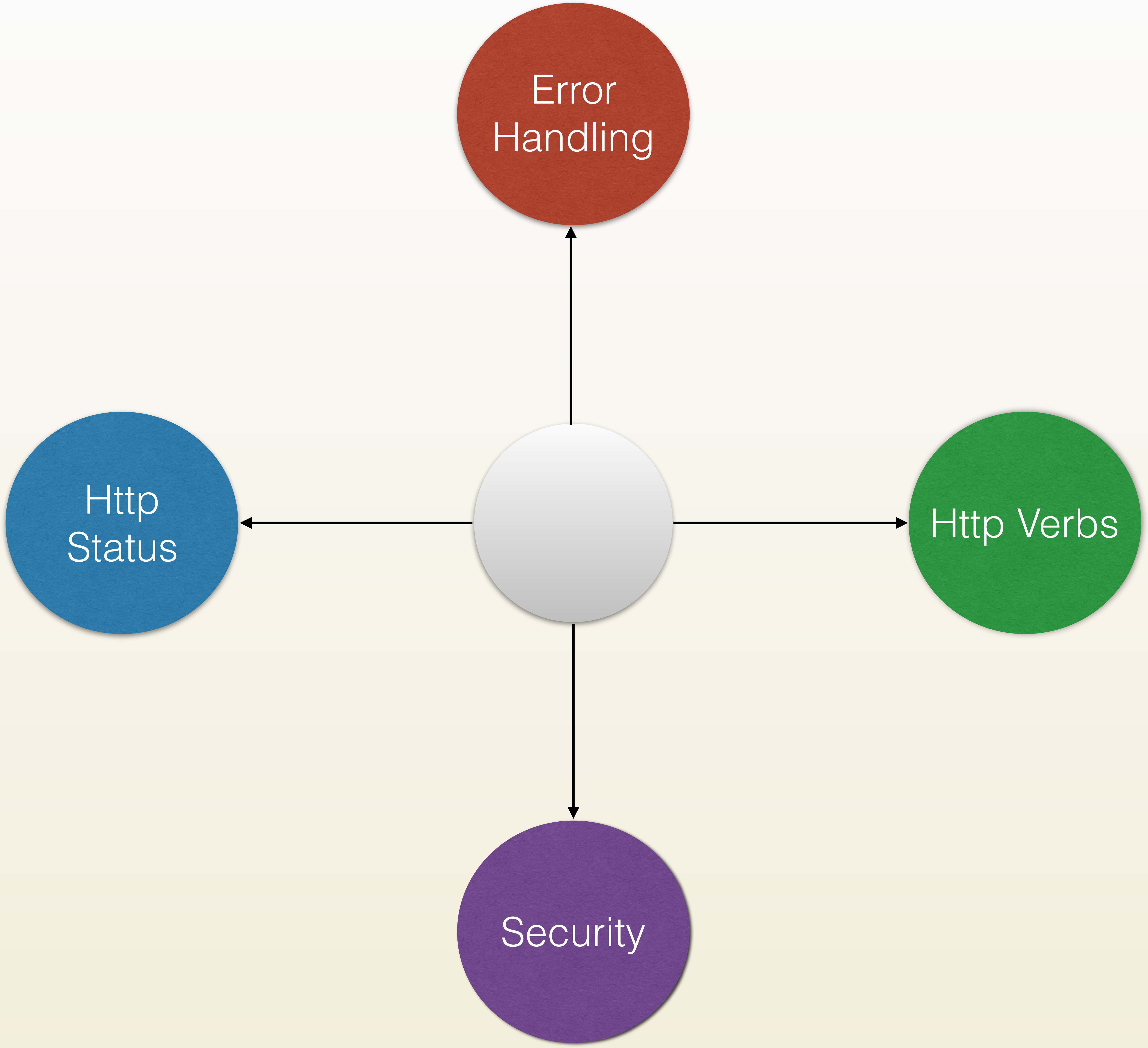
HTTP verbs

RESTful notes tries to a

Verb
GET
POST
PATCH
DELETE

HTTP statu

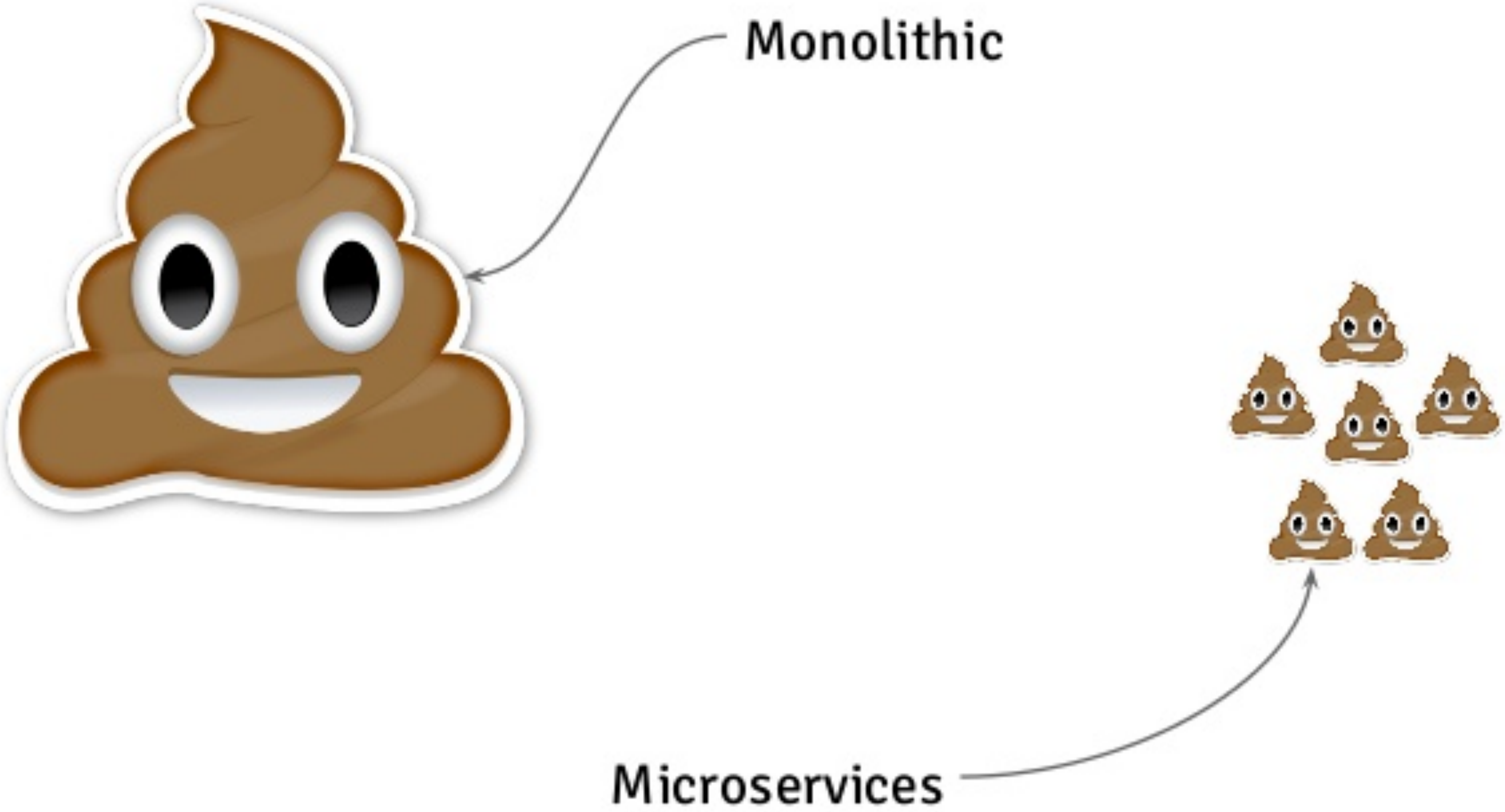
@codeJENNerator



Central Information

@codeJENNerator

Monolithic vs Microservices



Monolithic

Microservices

Swagger

@codeJENNerator

Swagger Approaches

SpringFox

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

@codeJENNerator

Super Easy Install

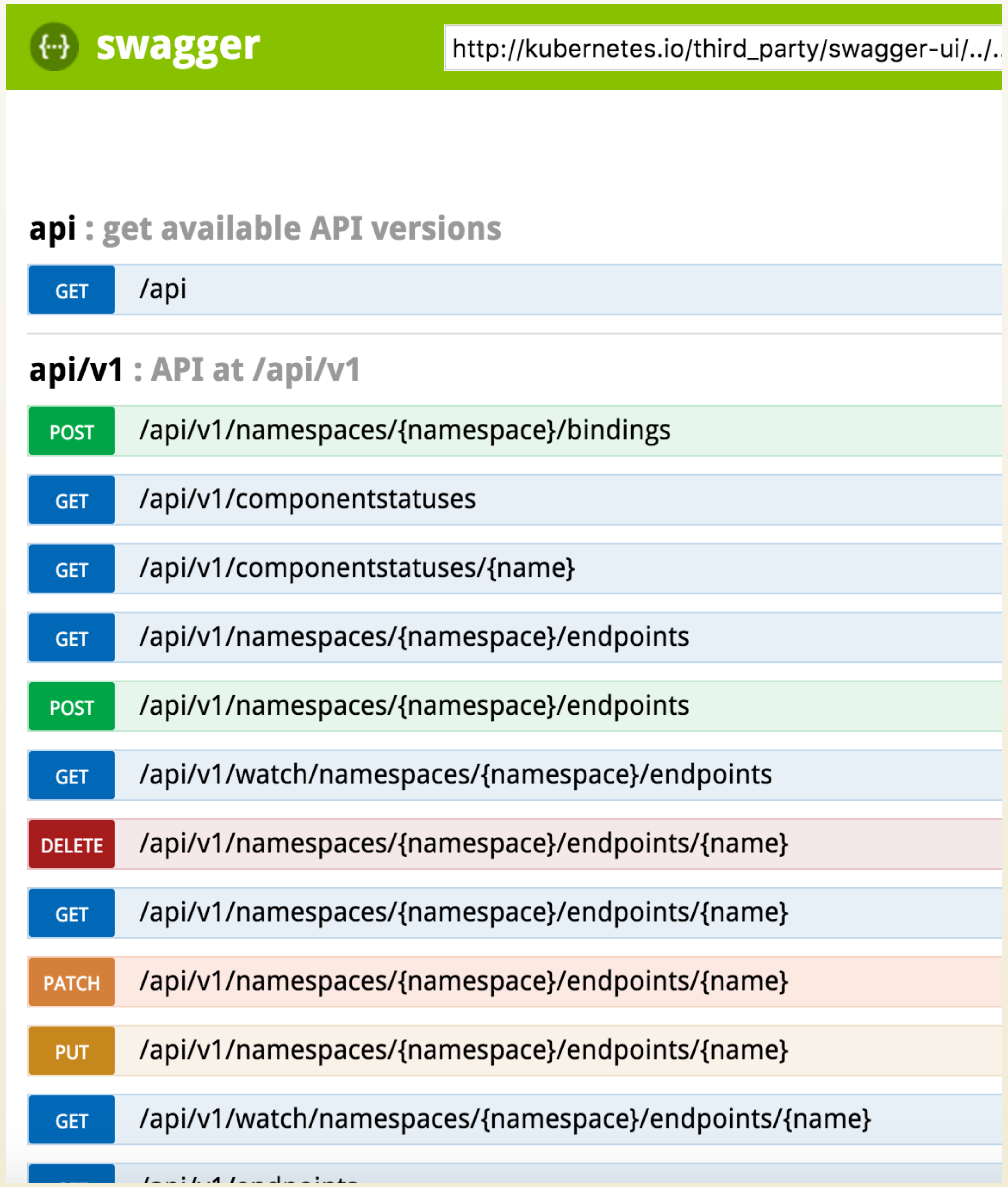
- `io.springfox:springfox-swagger2`
- `io.springfox:springfox-swagger-ui`

Custom Swagger Specification

```
{
  "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":
    .
    .
    .
}
```

Swagger UI

- Springfox generated UI
- Copy static files and customize



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}

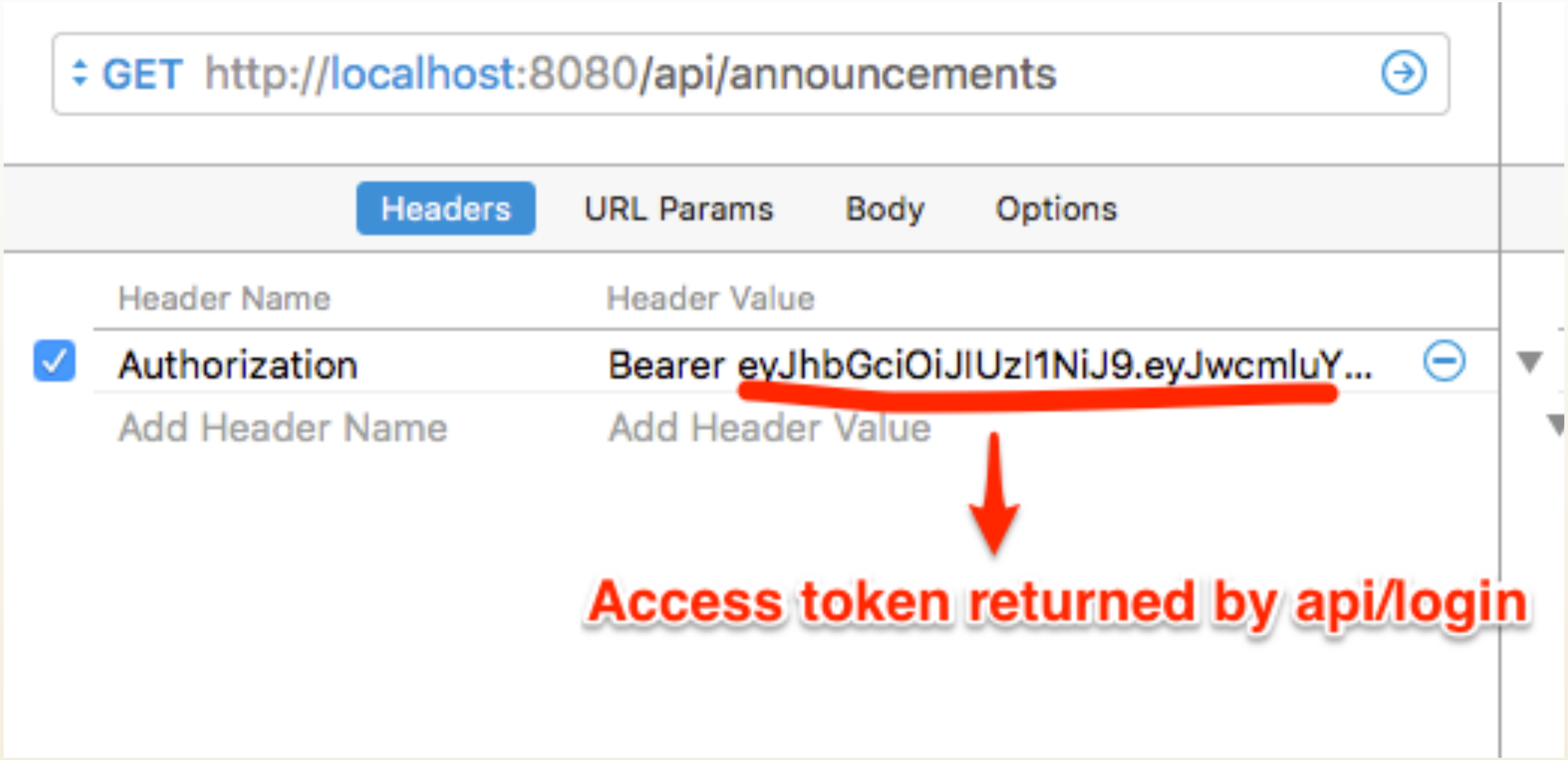
PUT /api/v1/namespaces/{namespace}/endpoints/{name}

GET /api/v1/watch/namespaces/{namespace}/endpoints/{name}

/api/v1/endpoints

@codeJENNerator

Considerations



GET http://localhost:8080/api/announcements

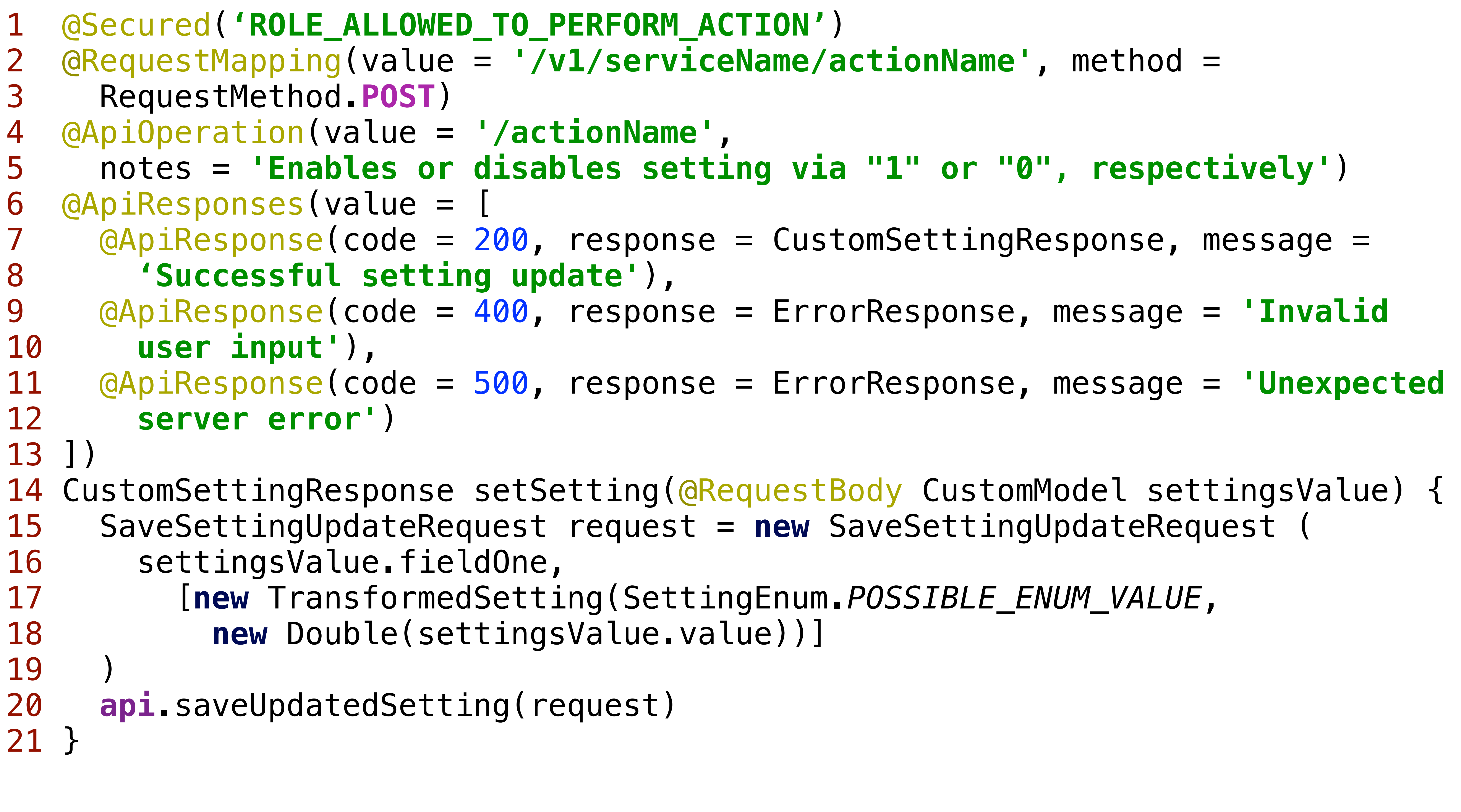
Header Name	Header Value
☑ Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJwcmluY...
Add Header Name	Add Header Value

Access token returned by api/login

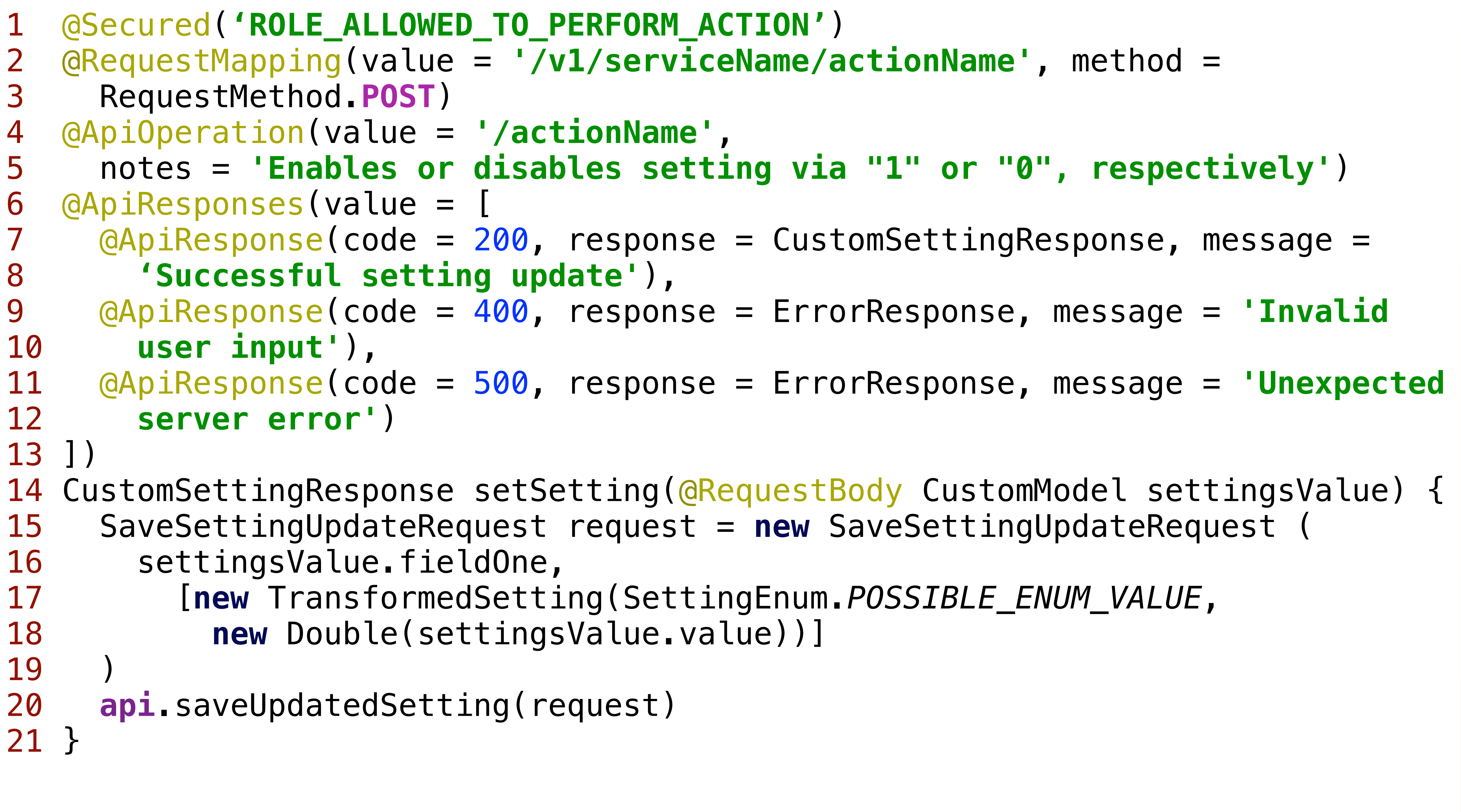
img src: http://sergiodelamo.es/how-to-secure-your-grails-3-api-with-spring-security-rest-for-grails/

Customization

@codeJENNerator

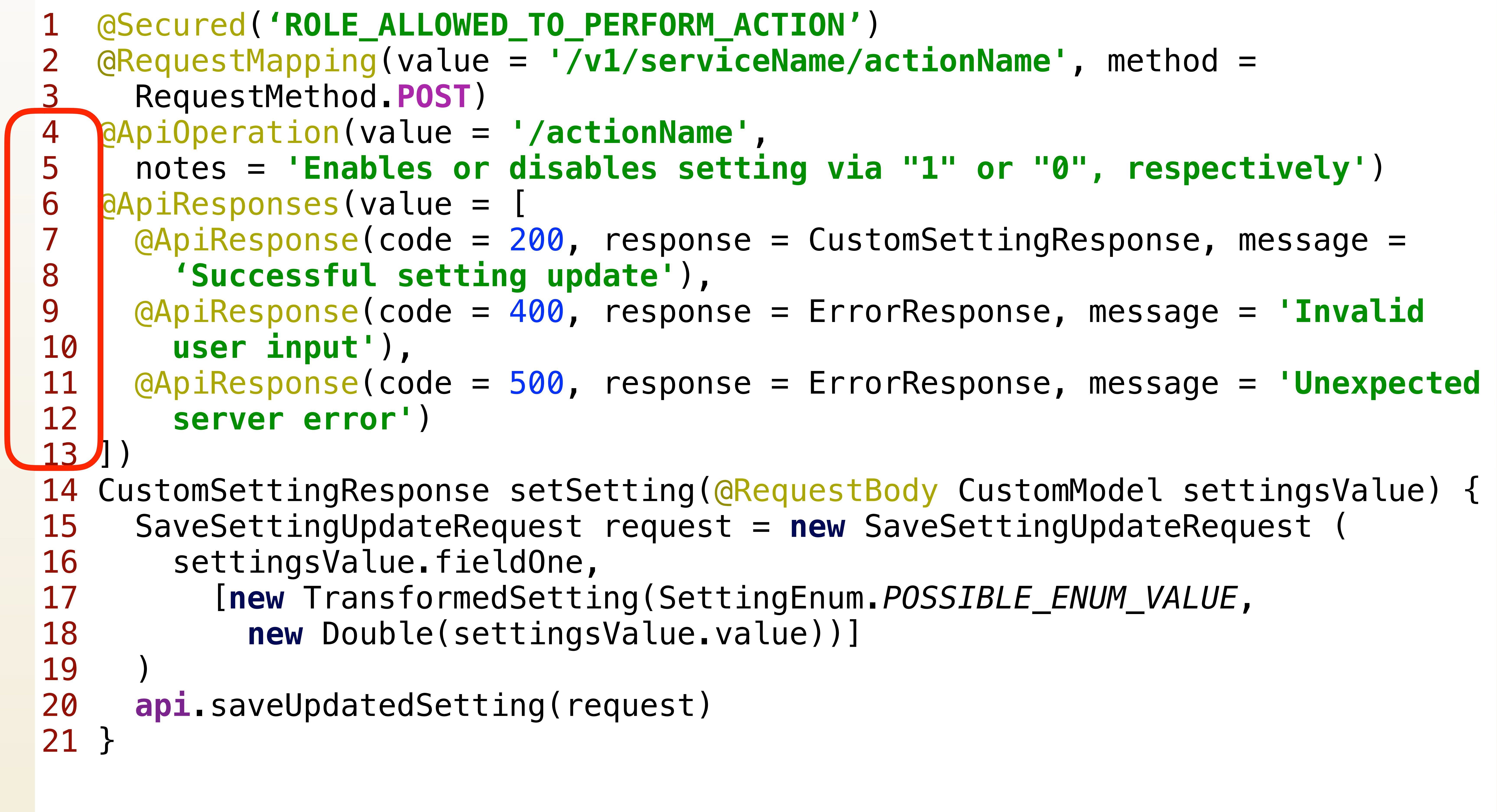


```
@Secured(‘ROLE_ALLOWED_TO_PERFORM_ACTION’)
@RequestMapping(value = '/v1/serviceName/actionName', method =
 RequestMethod.POST)
@ApiOperation(value = '/actionName',
 notes = 'Enables or disables setting via "1" or "0", respectively')
@ApiResponses(value = [
 @ApiResponse(code = 200, response = CustomSettingResponse, message =
  ‘Successful setting update'),
 @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
  user input'),
 @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
  server error')
])
CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
 SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
  settingsValue.fieldOne,
   [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
    new Double(settingsValue.value))]
 )
 api.saveUpdatedSetting(request)
}
```



```
@Secured(‘ROLE_ALLOWED_TO_PERFORM_ACTION’)
@RequestMapping(value = '/v1/serviceName/actionName', method =
  RequestMethod.POST)
@ApiOperation(value = '/actionName',
  notes = 'Enables or disables setting via "1" or "0", respectively')
@ApiResponses(value = [
  @ApiResponse(code = 200, response = CustomSettingResponse, message =
    ‘Successful setting update'),
  @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
    user input'),
  @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
    server error')
])
CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
  SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
    settingsValue.fieldOne,
      [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
        new Double(settingsValue.value))]
  )
  api.saveUpdatedSetting(request)
}
```

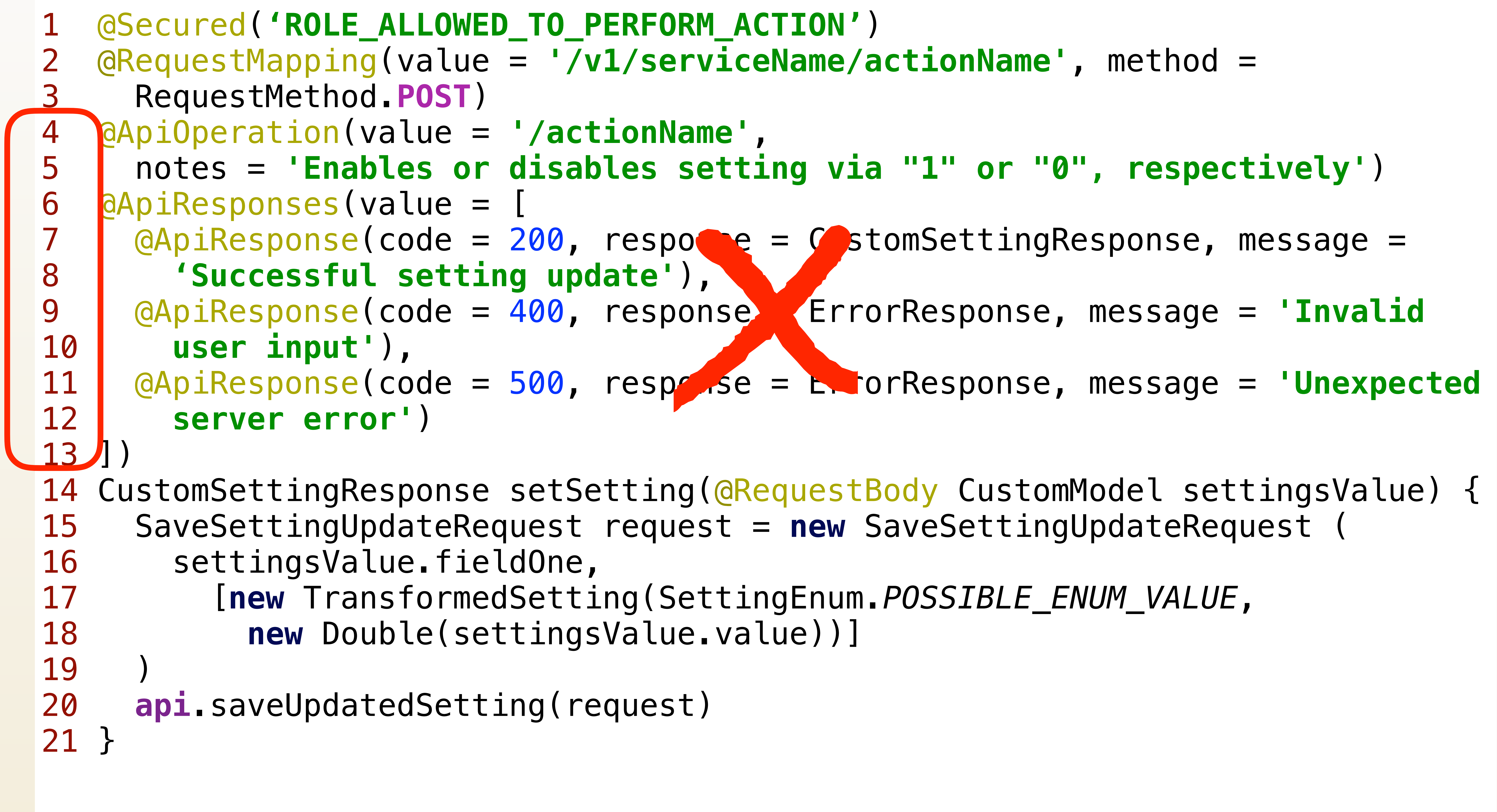
Annotation Hell



```
@Secured(‘ROLE_ALLOWED_TO_PERFORM_ACTION’)
@RequestMapping(value = '/v1/serviceName/actionName', method =
  RequestMethod.POST)
@ApiOperation(value = '/actionName',
  notes = 'Enables or disables setting via "1" or "0", respectively')
@ApiResponses(value = [
  @ApiResponse(code = 200, response = CustomSettingResponse, message =
    ‘Successful setting update'),
  @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
    user input'),
  @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
    server error')
])
CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
  SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
    settingsValue.fieldOne,
      [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
        new Double(settingsValue.value))]
  )
  api.saveUpdatedSetting(request)
}
```

Annotation Hell

@codeJENNerator

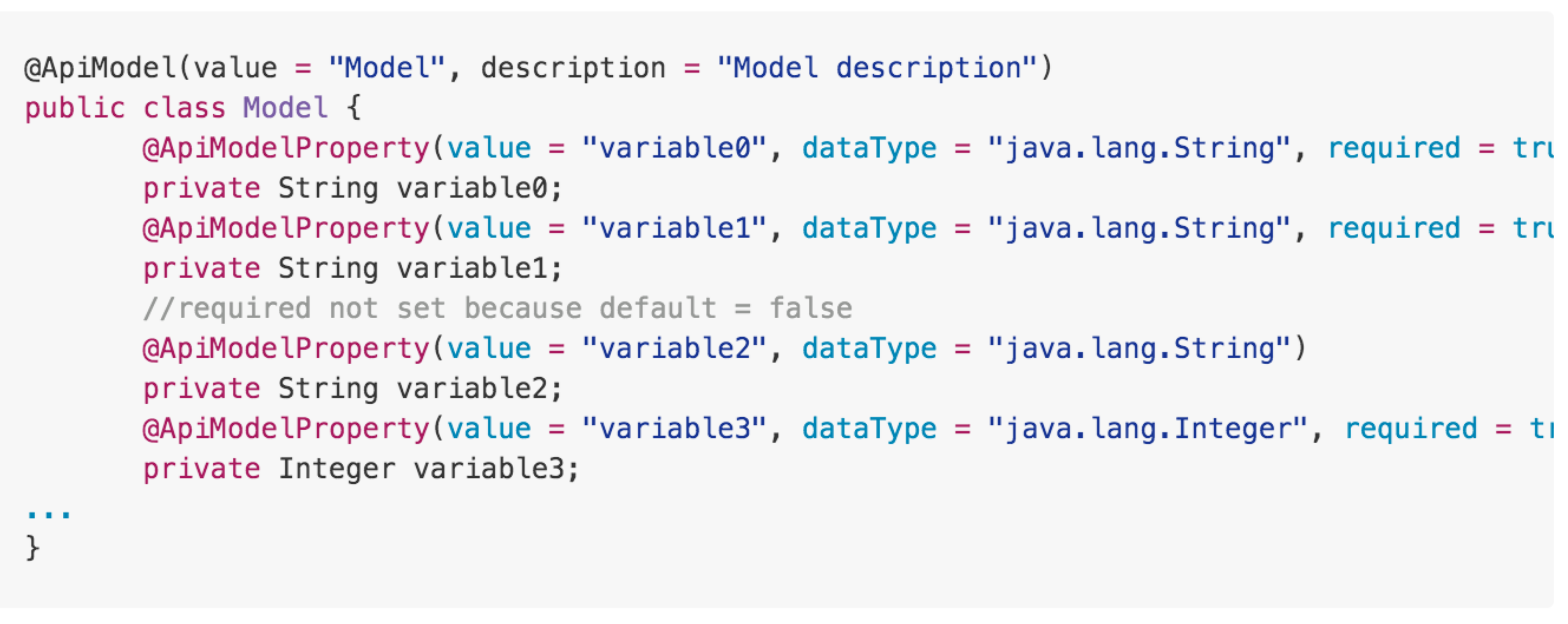


```
@Secured(‘ROLE_ALLOWED_TO_PERFORM_ACTION’)
@RequestMapping(value = '/v1/serviceName/actionName', method =
  RequestMethod.POST)
@ApiOperation(value = '/actionName',
  notes = 'Enables or disables setting via "1" or "0", respectively')
@ApiResponses(value = [
  @ApiResponse(code = 200, response = CustomSettingResponse, message =
    ‘Successful setting update'),
  @ApiResponse(code = 400, response = ErrorResponse, message = 'Invalid
    user input'),
  @ApiResponse(code = 500, response = ErrorResponse, message = 'Unexpected
    server error')
])
CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
  SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
    settingsValue.fieldOne,
      [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
        new Double(settingsValue.value))]
  )
  api.saveUpdatedSetting(request)
}
```

Annotation Hell

@codeJENNerator

```
@Secured(‘ROLE_ALLOWED_TO_PERFORM_ACTION’)
@RequestMapping(value = '/v1/serviceName/actionName', method =
  RequestMethod.POST)
CustomSettingResponse setSetting(@RequestBody CustomModel settingsValue) {
 SaveSettingUpdateRequest request = new SaveSettingUpdateRequest (
   settingsValue.fieldOne,
     [new TransformedSetting(SettingEnum.POSSIBLE_ENUM_VALUE,
       new Double(settingsValue.value))]
 )
  api.saveUpdatedSetting(request)
}
```



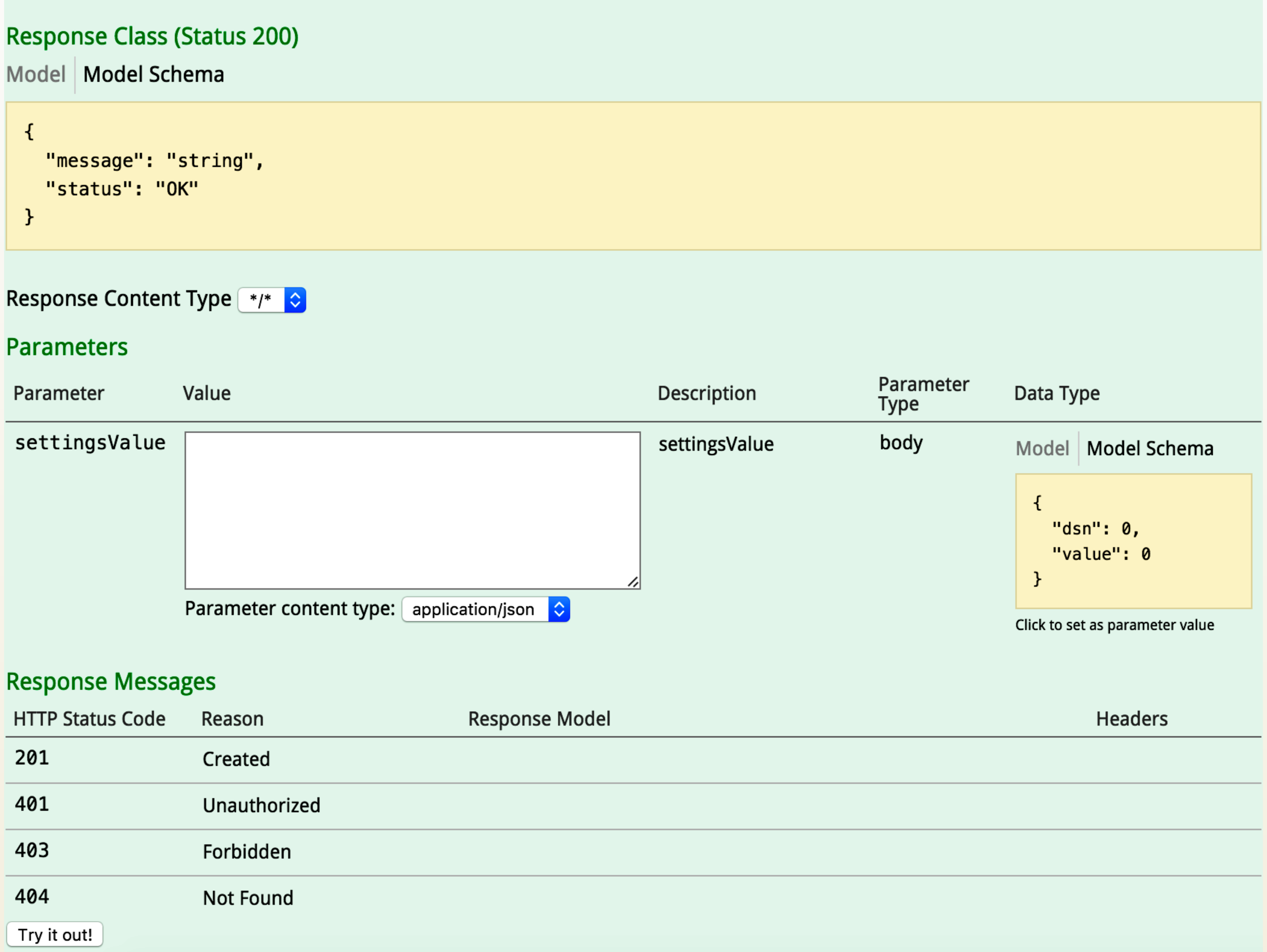
```java
@ApiModel(value = "Model", description = "Model description")
public class Model {
        @ApiModelProperty(value = "variable0", dataType = "java.lang.String", required = tru
        private String variable0;
        @ApiModelProperty(value = "variable1", dataType = "java.lang.String", required = 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

@codeJENNerator

Advantages



Response Class (Status 200)

```
{
  "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

```
{
  "dsn": 0,
  "value": 0
}
```

Response Messages

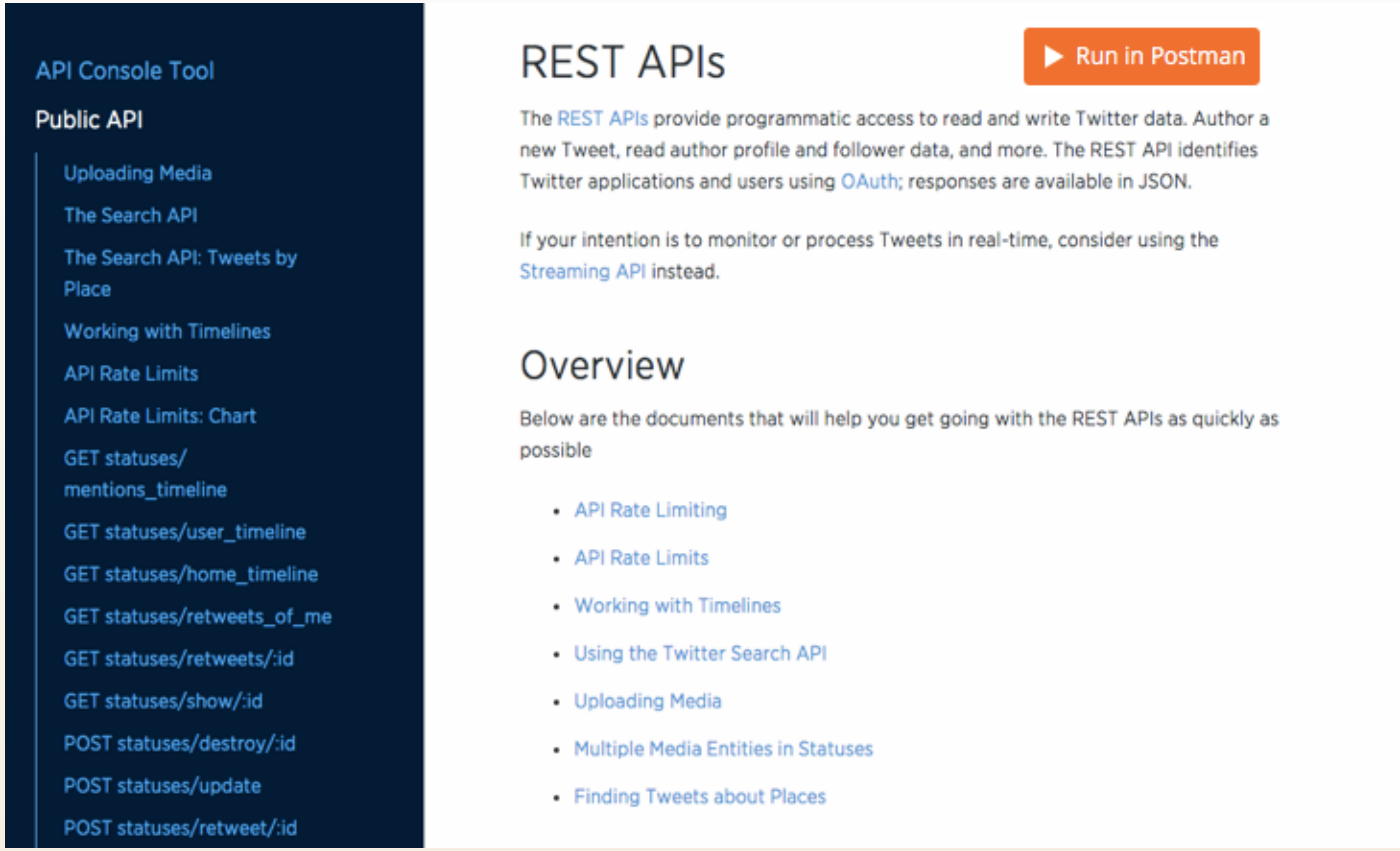
HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Swagger UI

“Try-It” Button

@codeJENNerator

The Alternatives



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
▶ Run in Postman

REST APIs

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

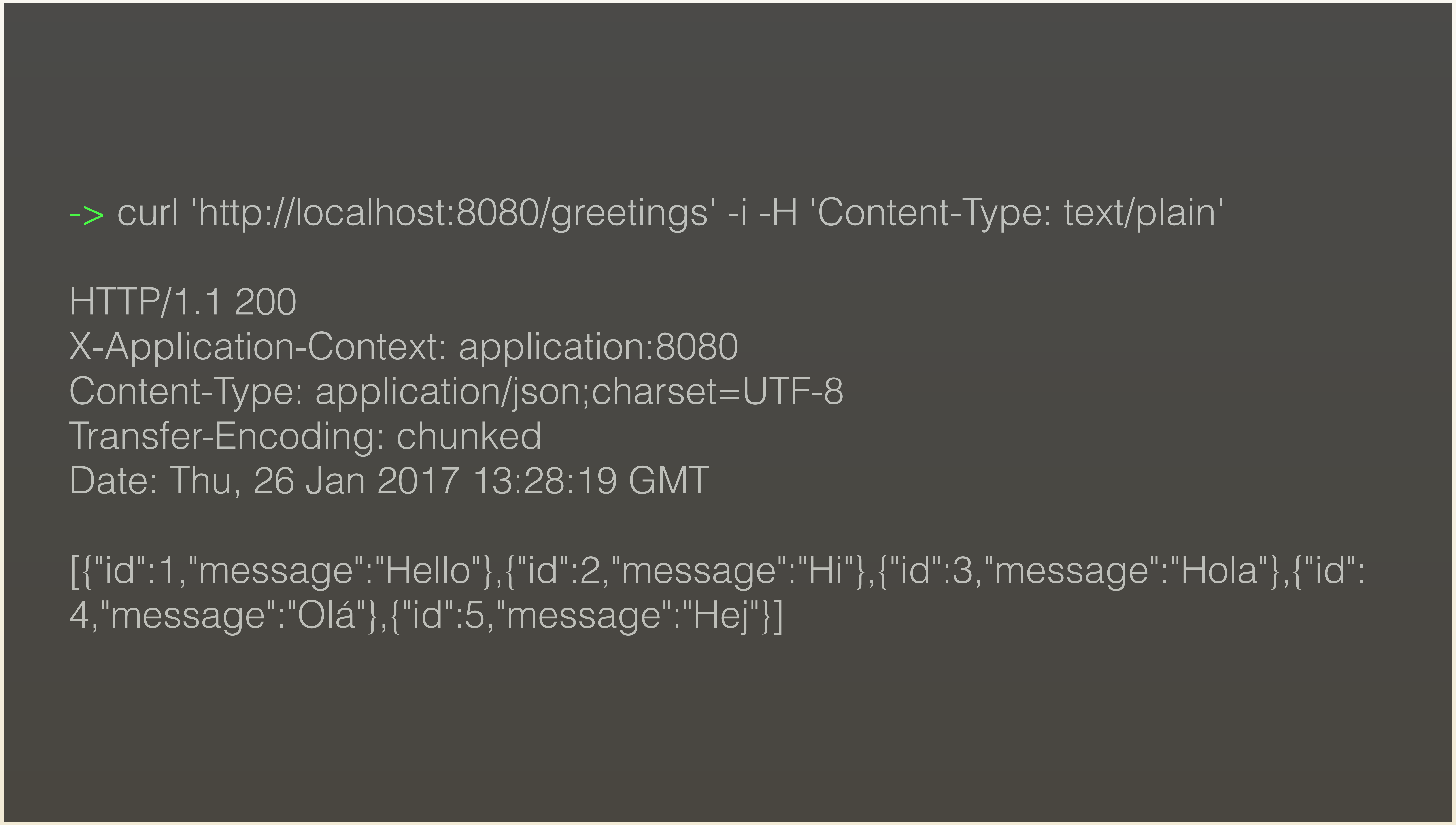
Below are the documents that will help you get going with the REST APIs as quickly as possible

- 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 Collections

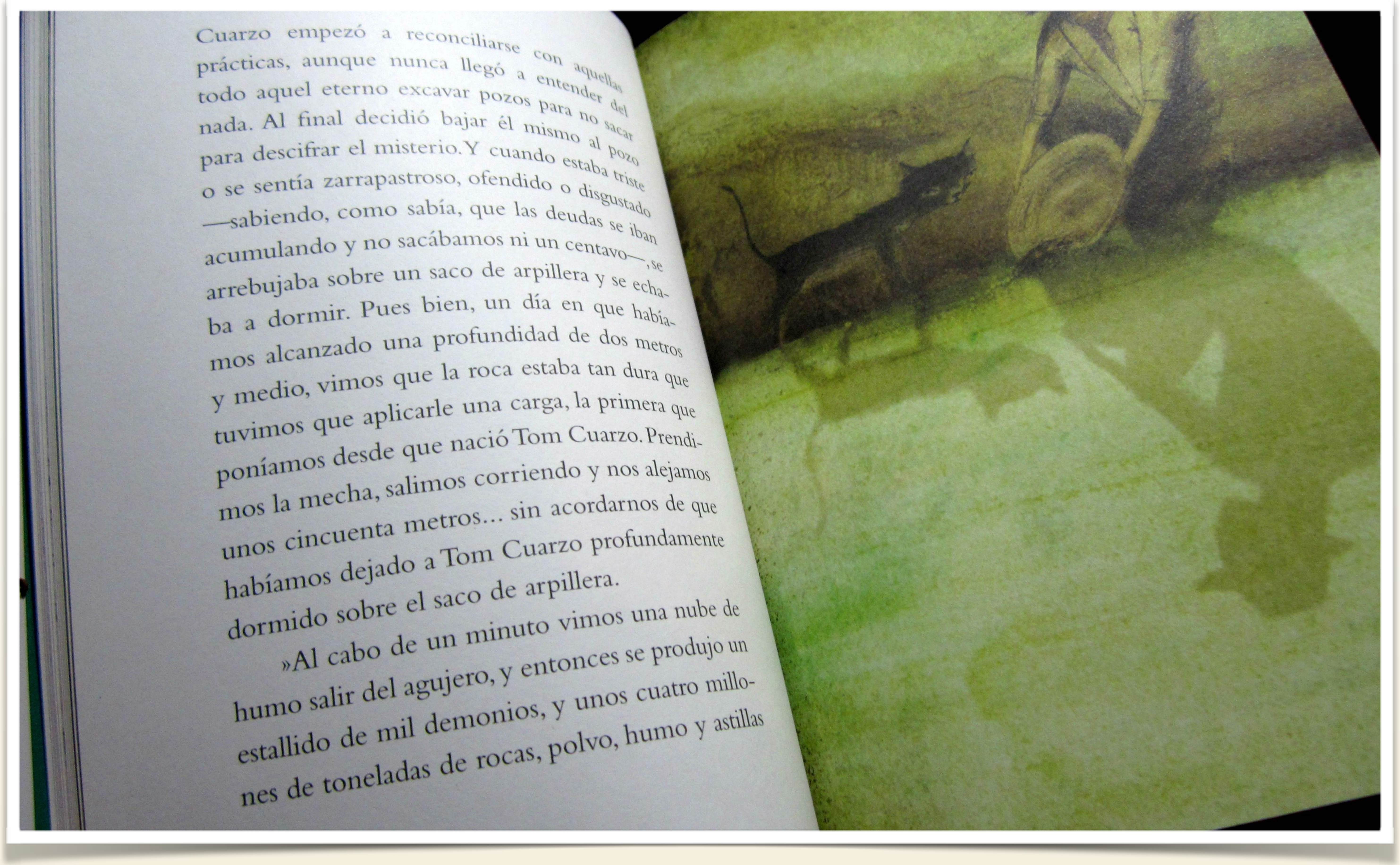
src: https://www.getpostman.com/img/v1/docs/run_btn_ux/run_btn_ux_1.png

Curl



```
-> curl 'http://localhost:8080/greetings' -i -H 'Content-Type: text/plain'

HTTP/1.1 200
X-Application-Context: application:8080
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 26 Jan 2017 13:28:19 GMT

[{"id":1,"message":"Hello"},{"id":2,"message":"Hi"},{"id":3,"message":"Hola"},{"id":
4,"message":"Olá"},{"id":5,"message":"Hej"}]
```

Mixing Spring REST Docs and Swagger



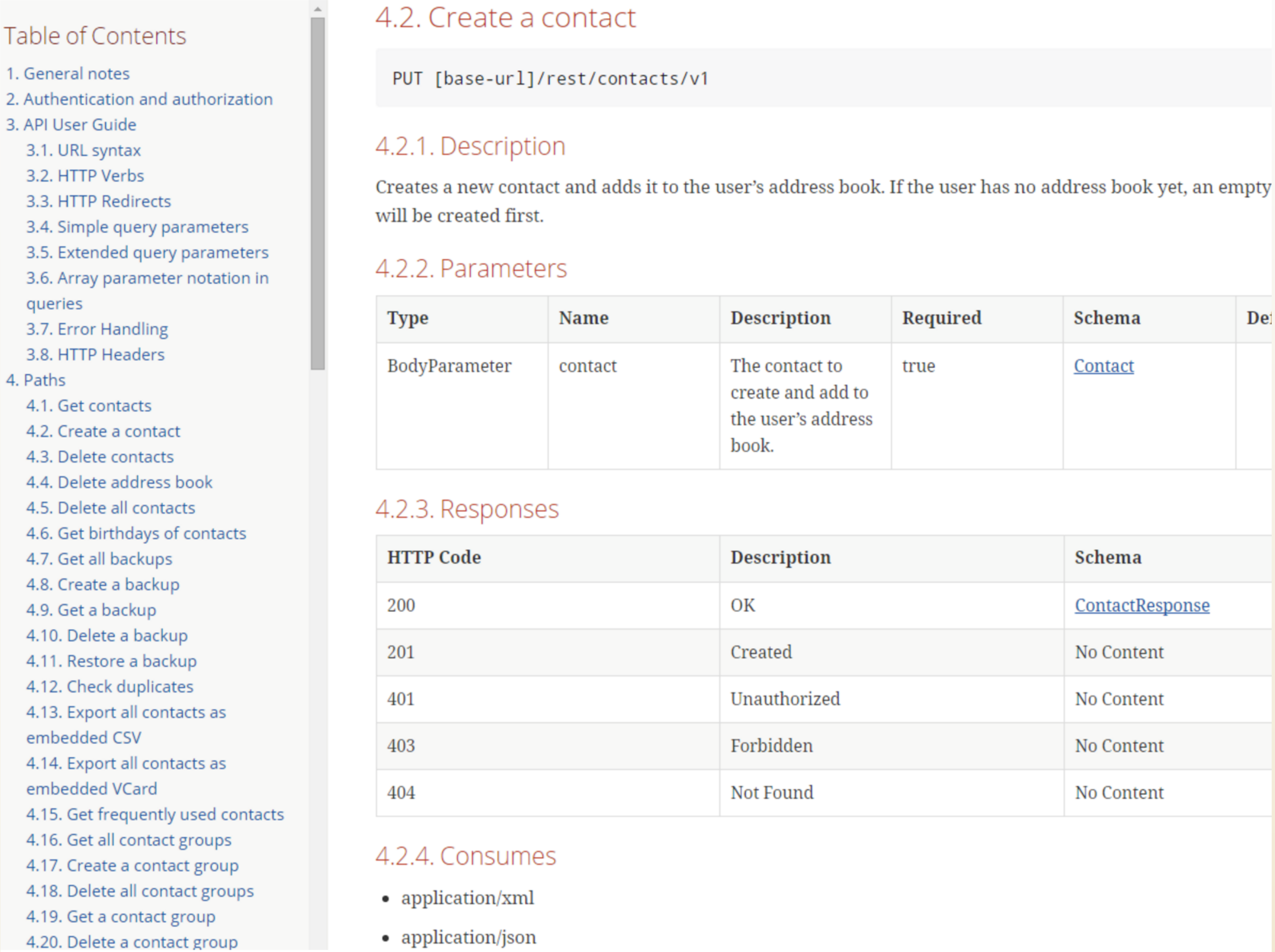
Cuarzo empezó a reconciliarse con aquellas prácticas, aunque nunca llegó a entender del todo aquel eterno excavar pozos para no sacar nada. Al final decidió bajar él mismo al pozo para descifrar el misterio.Y cuando estaba triste o se sentía zarrapastroso, ofendido o disgustado —sabiendo, como sabía, 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/

@codeJENNerator



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

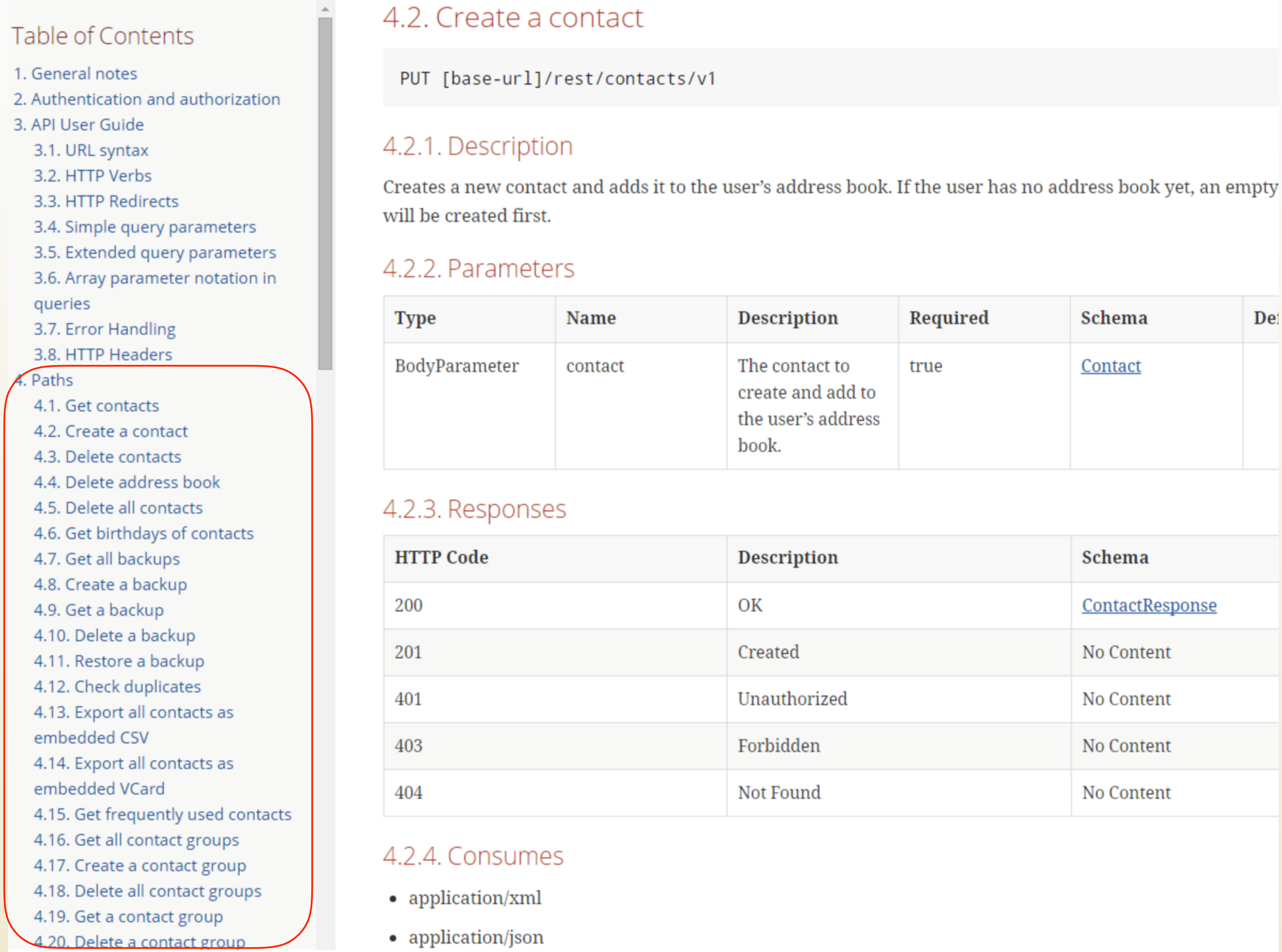


Table of Contents

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

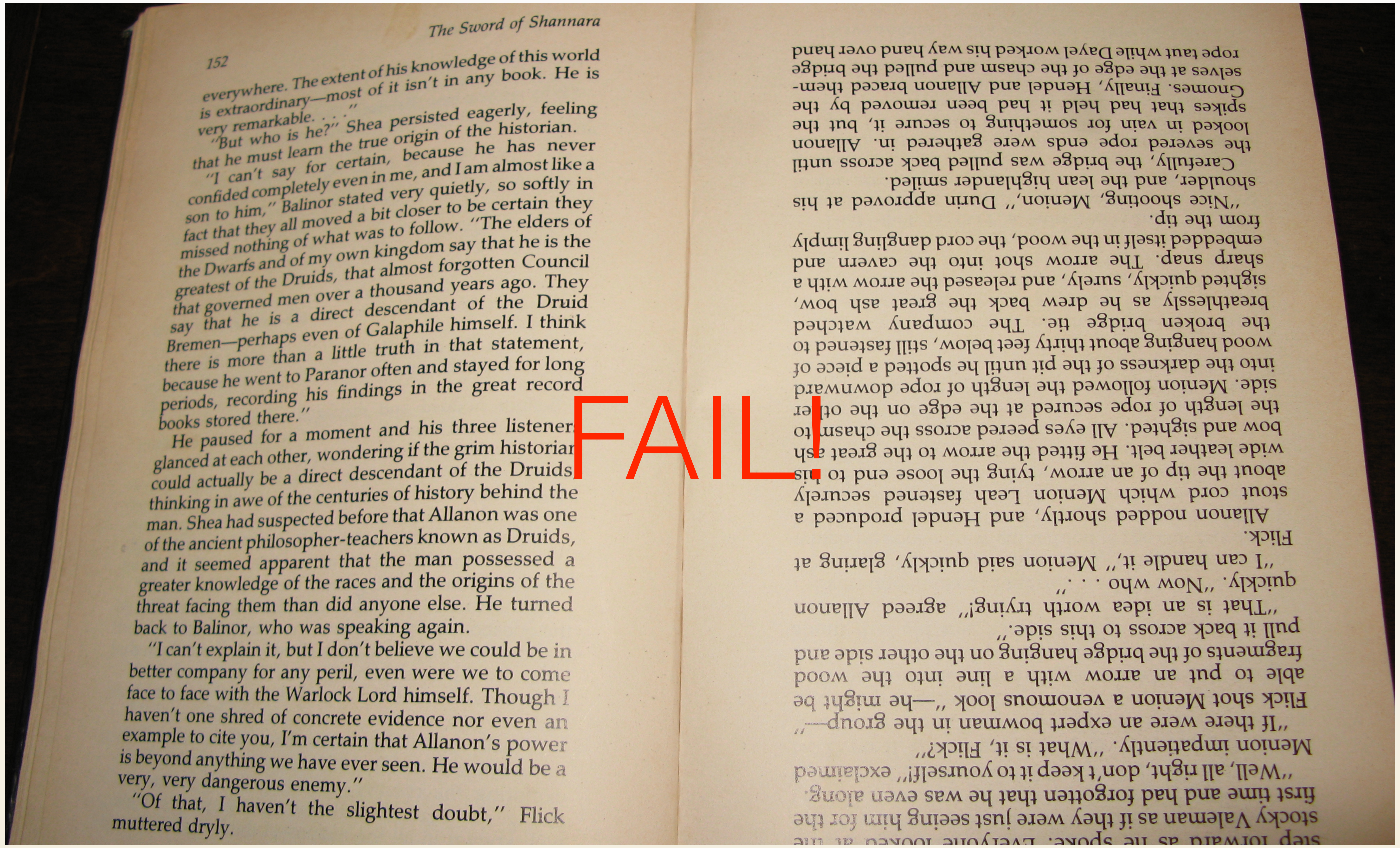
Type	Name	Description	Required	Schema	De
BodyParameter	contact	The contact to create and add to the user's address book.	true	Contact	

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



152 The Sword of Shannara

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."

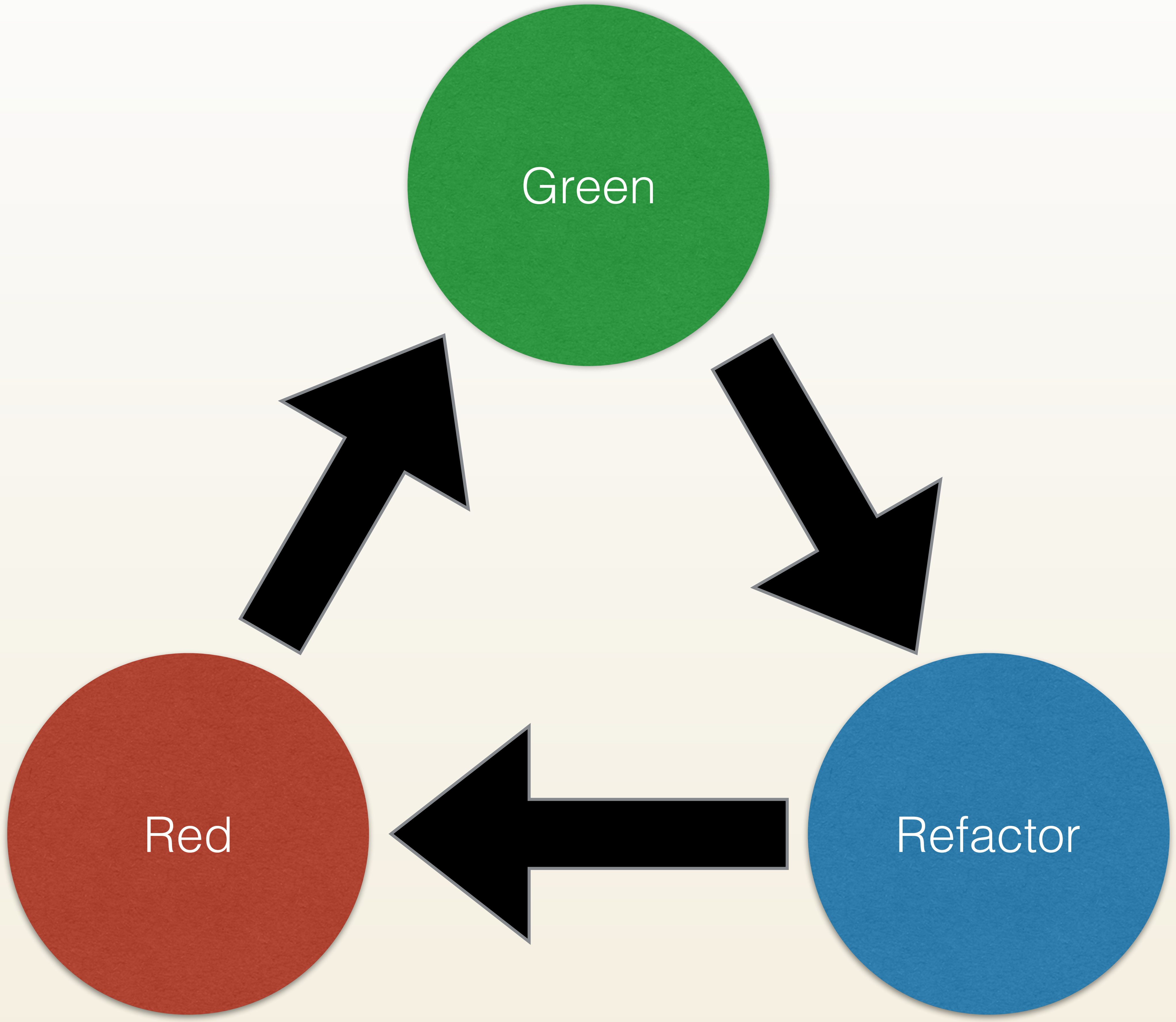
"Of that, I haven't the slightest doubt," Flick muttered dryly.

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

AssertJ-Swagger

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

@codeJENNerator



Test-Driven Documentation

@codeJENNerator

Test-Driven Documentation

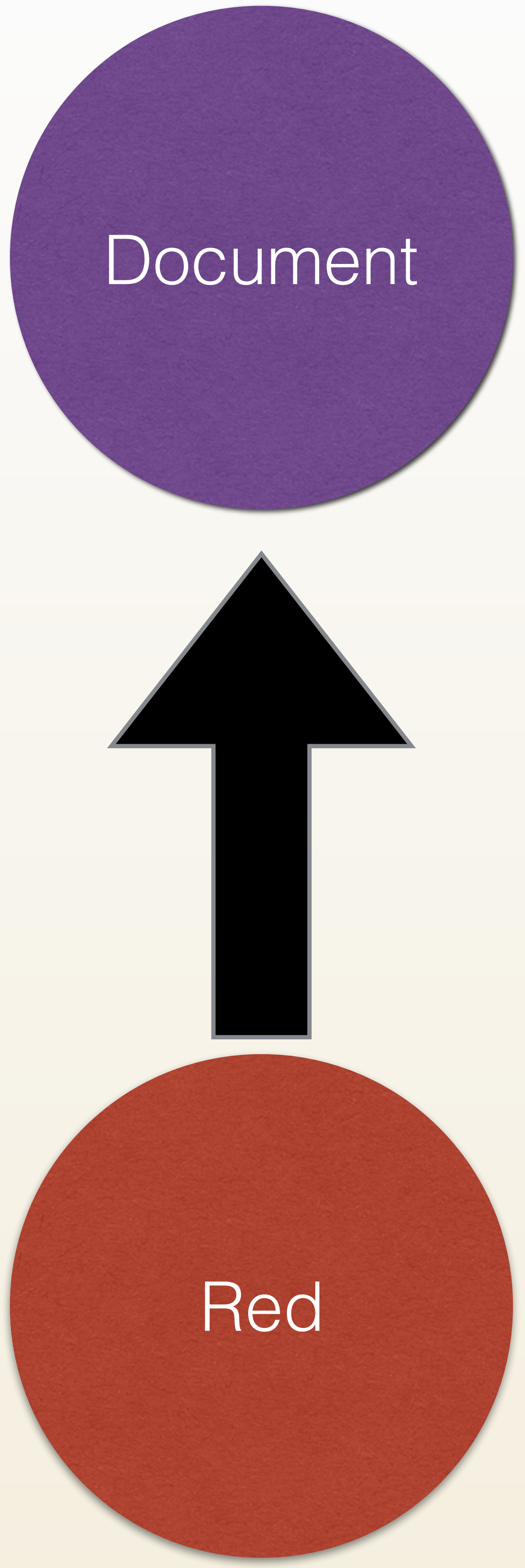
@codeJENNerator

Red

Test-Driven Documentation

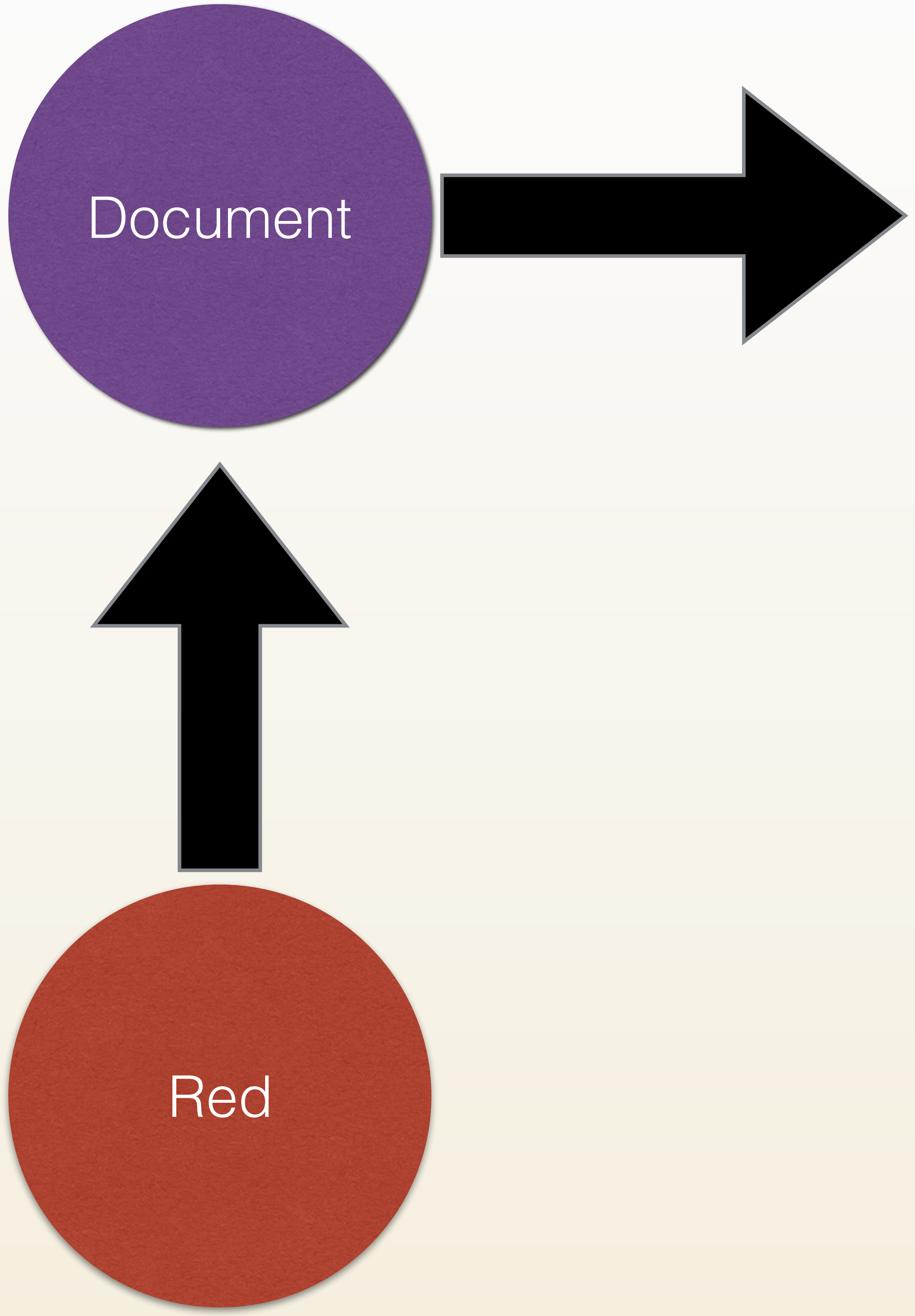
@codeJENNerator

Test-Driven Documentation



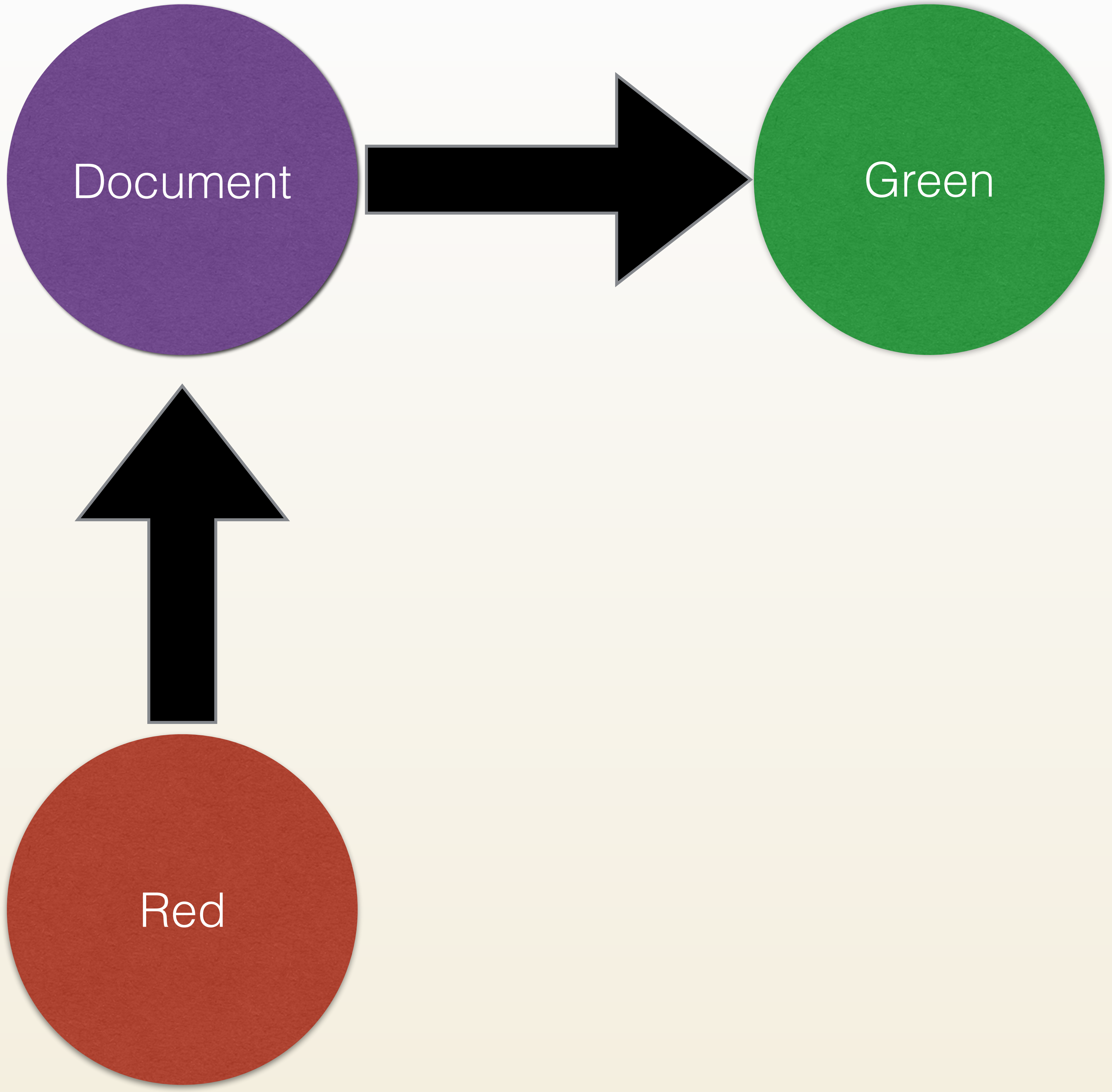
Test-Driven Documentation

@codeJENNerator



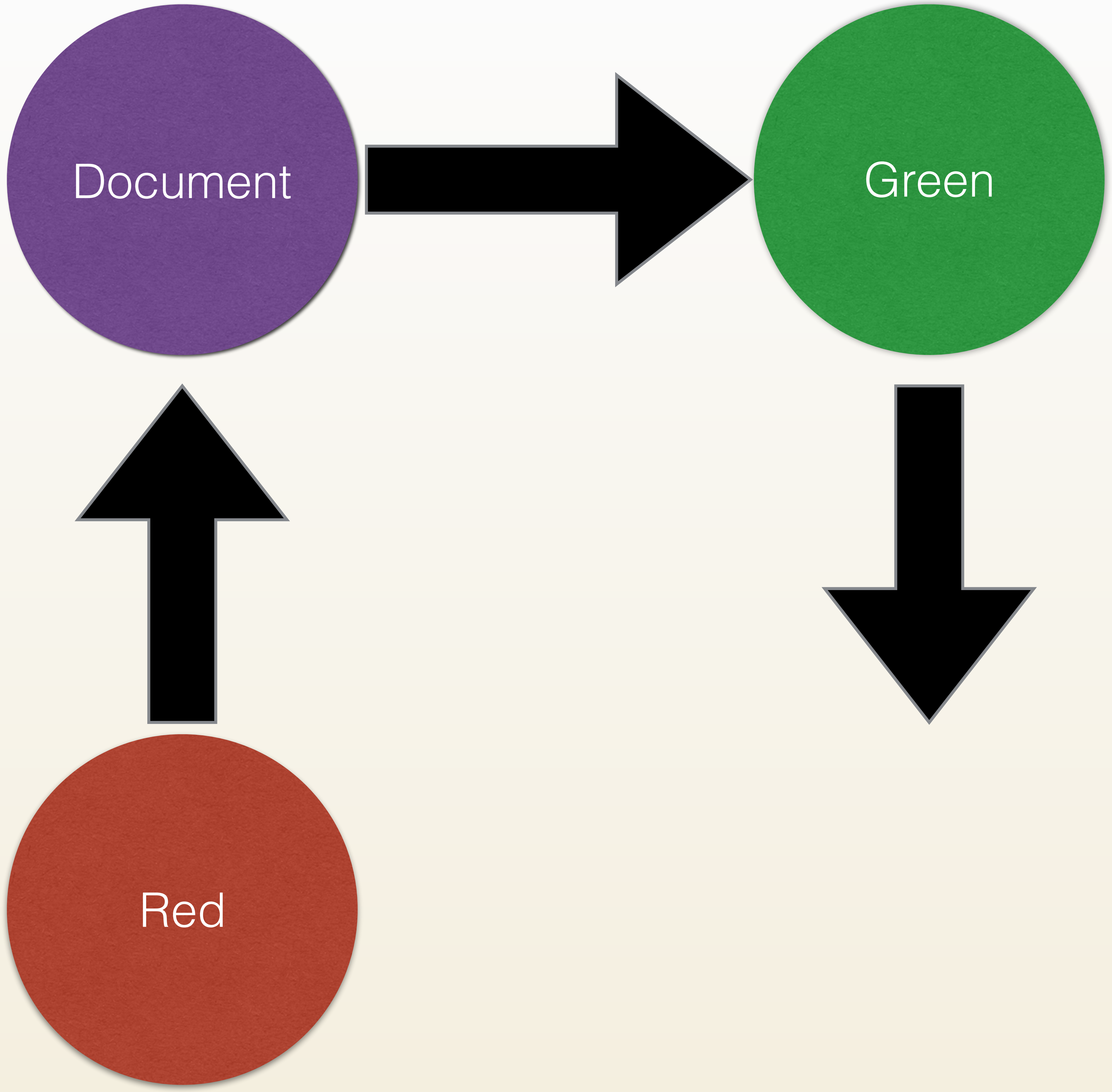
Test-Driven Documentation

@codeJENNerator



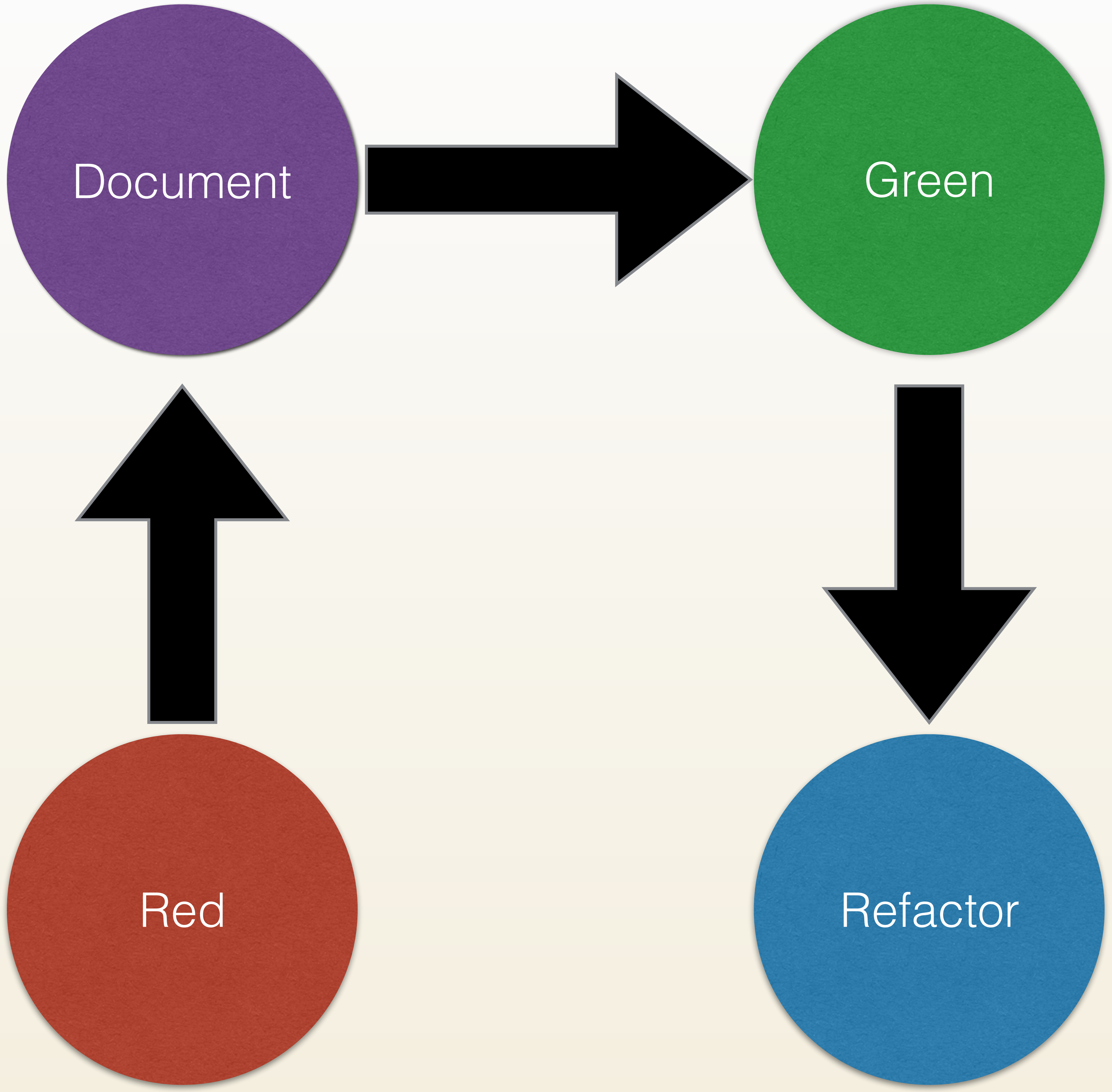
Test-Driven Documentation

@codeJENNerator



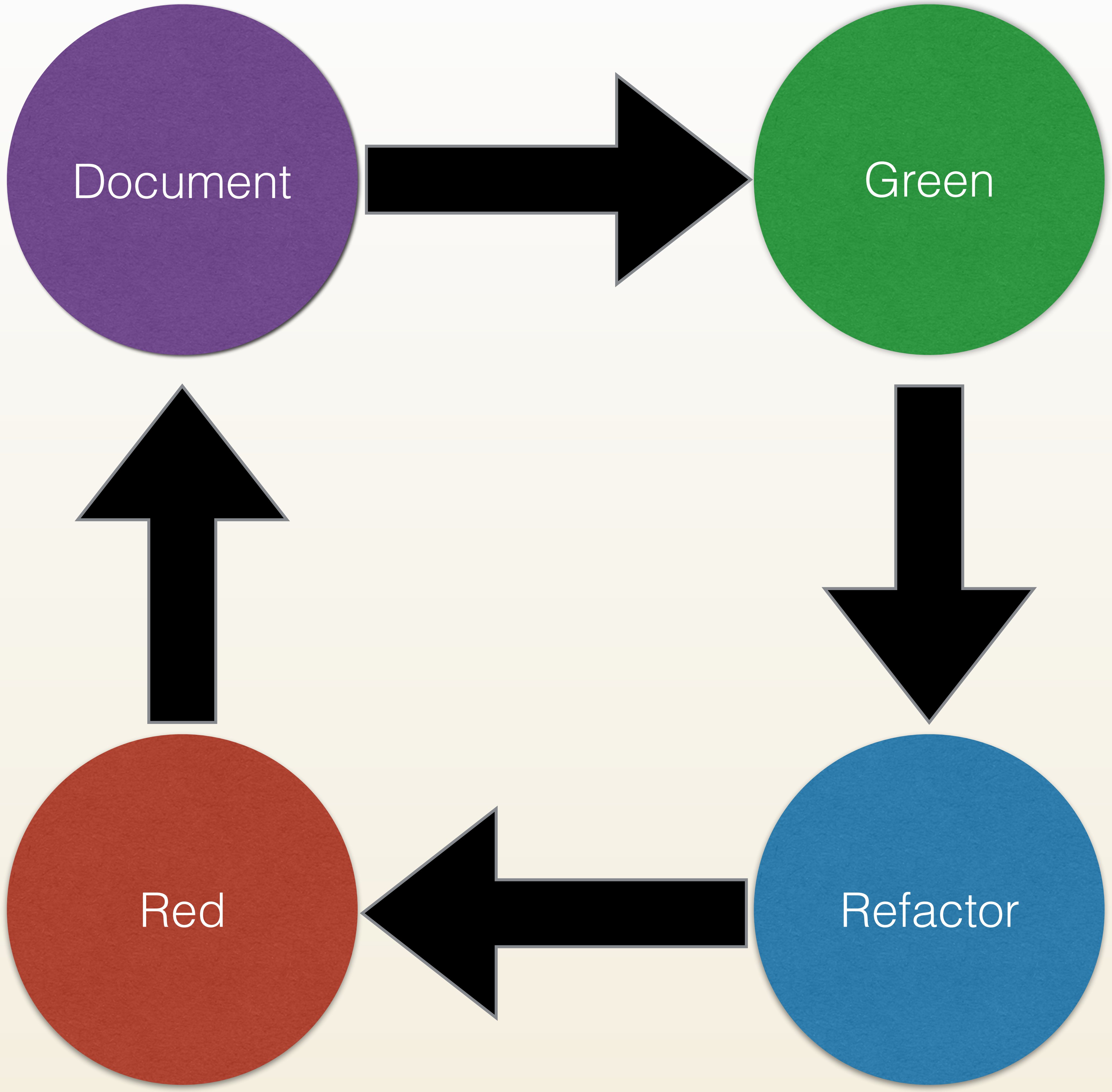
Test-Driven Documentation

@codeJENNerator



Test-Driven Documentation

@codeJENNerator



Test-Driven Documentation

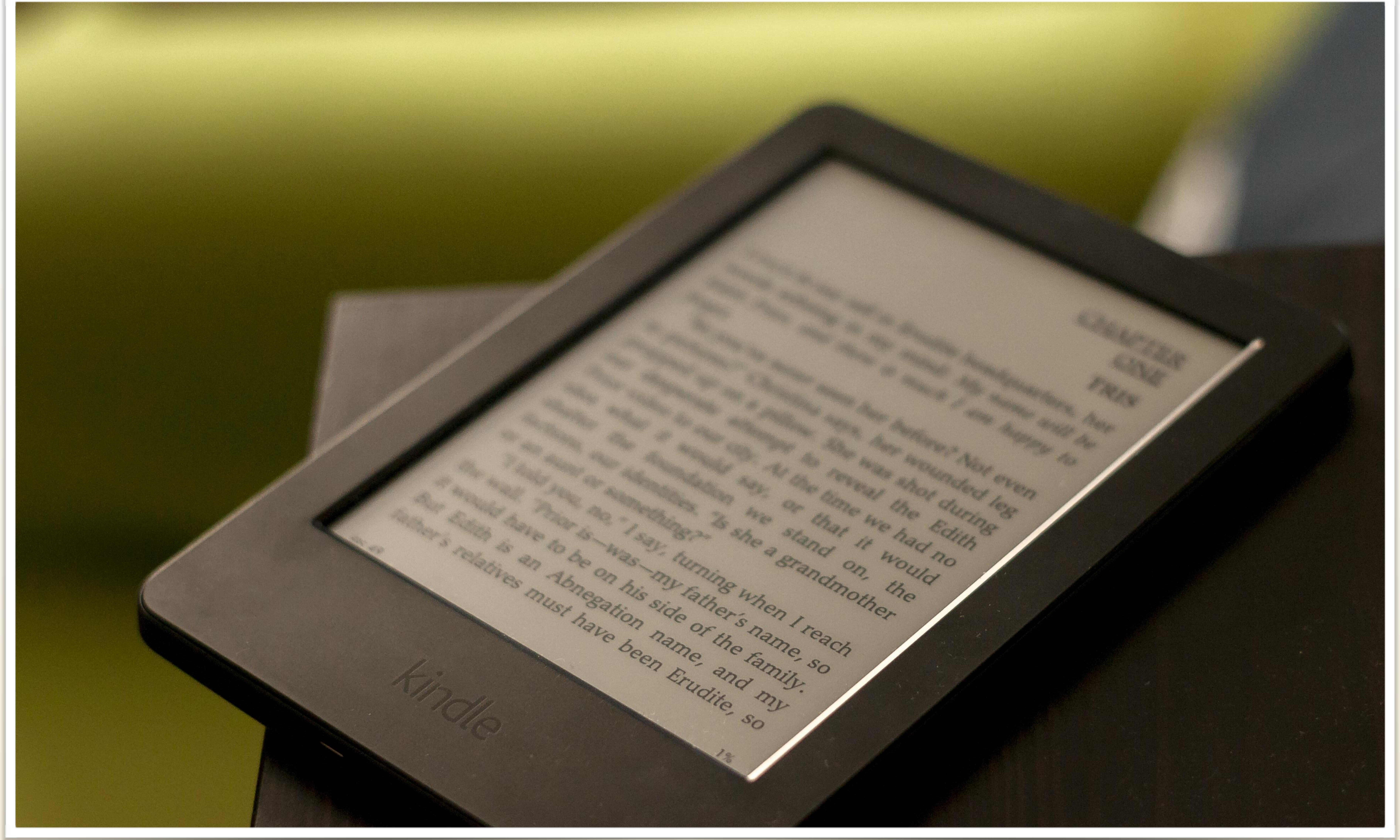
@codeJENNerator

Winning Solution

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

Winning Solution

- Ensures documentation matches implementation
- Encourages writing more tests
- Reduces duplication in docs and tests
- Removes annotations from source



Spring REST Docs

@codeJENNerator

Overview

projects.spring.io/spring-restdocs

Overview

- Sponsored by Pivotal
- Project Lead - Andy Wilkinson
- Current Version - 1.1.2 released in August 2016

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

Game Changers

@codeJENNerator

Game Changers

- Generated code snippets
- Tests fail when documentation is missing or out-of-date
- Rest APIs with Hypermedia
- Ratpack
  - Dynamic routing doesn’t work with Swagger

Getting Started

- Documentation - Spring Projects
- Documenting RESTful Apis - SpringOne2GX 2015 Andy Wilkinson
- Writing comprehensive and guaranteed up-to-date REST API documentation - SpringOne Platform 2016 Anders Evers

Release Highlights

1.0.0

MockMVC

1.0.0

AutoGenerated Snippets

MockMVC

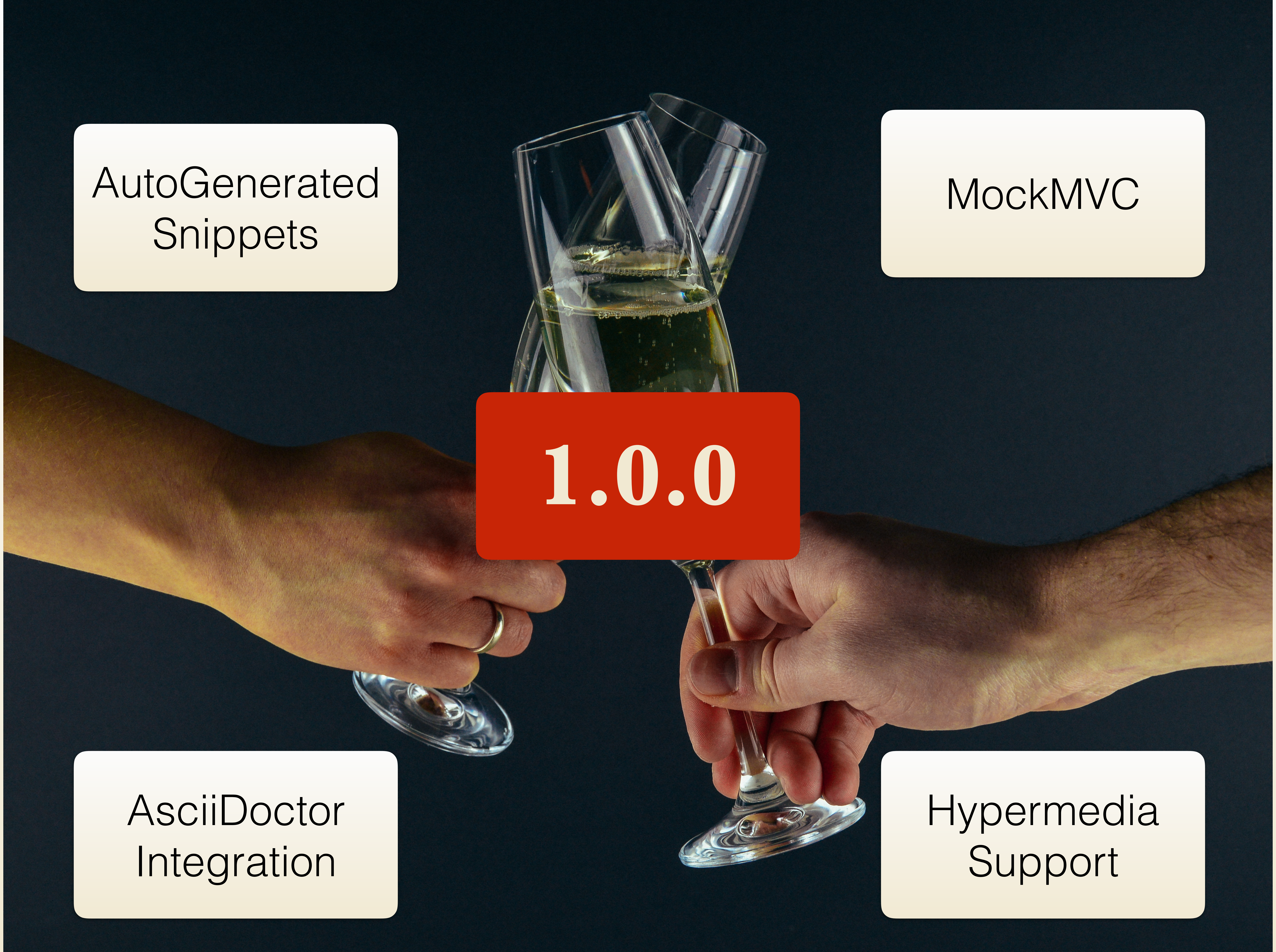
1.0.0

AutoGenerated Snippets

MockMVC

1.0.0

AsciiDoctor Integration



AutoGenerated Snippets

MockMVC

1.0.0

AsciiDoctor Integration

Hypermedia Support

1.1.0

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

1.1.0

RestAssured

Reusable Snippets

TestNG

Markdown

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

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

Support for Deeply Nested JSON

AsciiDoctor Macro

Rest Assured 3.0

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

Groovier Spring REST docs

- Spring Boot
- Ratpack
- Grails

@codeJENNerator

Groovier Spring REST docs Example - Spring Boot

@codeJENNerator

Groovier Spring REST docs Example - Spring Boot

- Groovy Spring Boot Project

Groovier Spring REST docs Example - Spring Boot

- Groovy Spring Boot Project

+ Asciidoctor Gradle plugin

Groovier Spring REST docs Example - Spring Boot

- Groovy Spring Boot Project

+ Asciidoctor Gradle plugin

+ **MockMVC** and documentation to Spock tests

Groovier Spring REST docs Example - Spring Boot

- Groovy Spring Boot Project

+ Asciidoctor Gradle plugin

+ **MockMVC** and documentation to Spock tests

+ Add to static assets during build

Groovy Spring Boot App

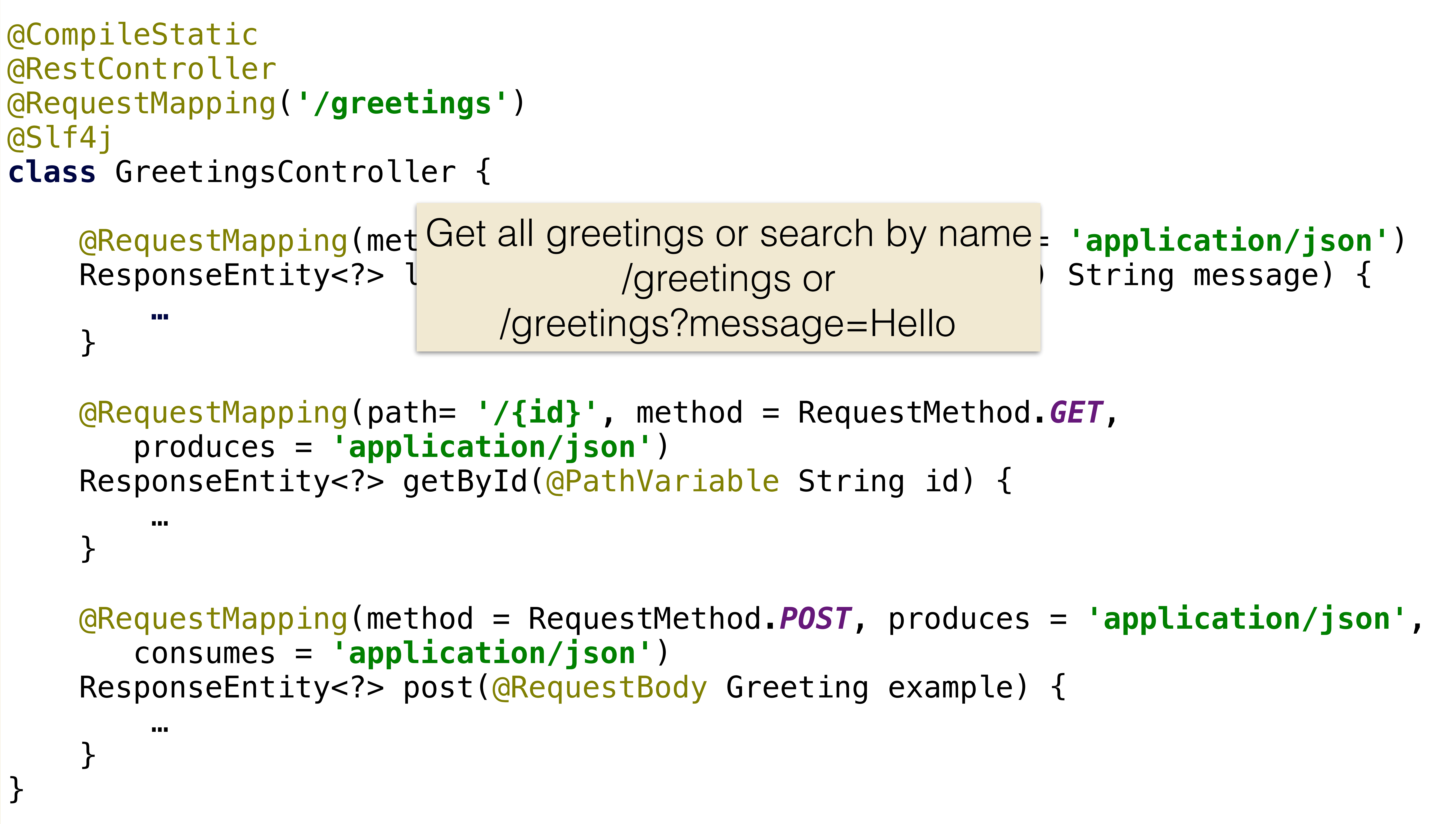
- Start with lazybones spring boot app
- Add mock endpoints for example

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

Endpoints

```
@CompileStatic
@RestController
@RequestMapping('/greetings')
@Slf4j
class GreetingsController {

    @RequestMapping(method = RequestMethod.GET, produces = 'application/json')
    ResponseEntity<?> list(@RequestParam(required = false) String message) {
        …
    }

    @RequestMapping(path= '/{id}', method = RequestMethod.GET,
       produces = 'application/json')
    ResponseEntity<?> getById(@PathVariable String id) {
        …
    }

    @RequestMapping(method = RequestMethod.POST, produces = 'application/json',
       consumes = 'application/json')
    ResponseEntity<?> post(@RequestBody Greeting example) {
        …
    }
}
```

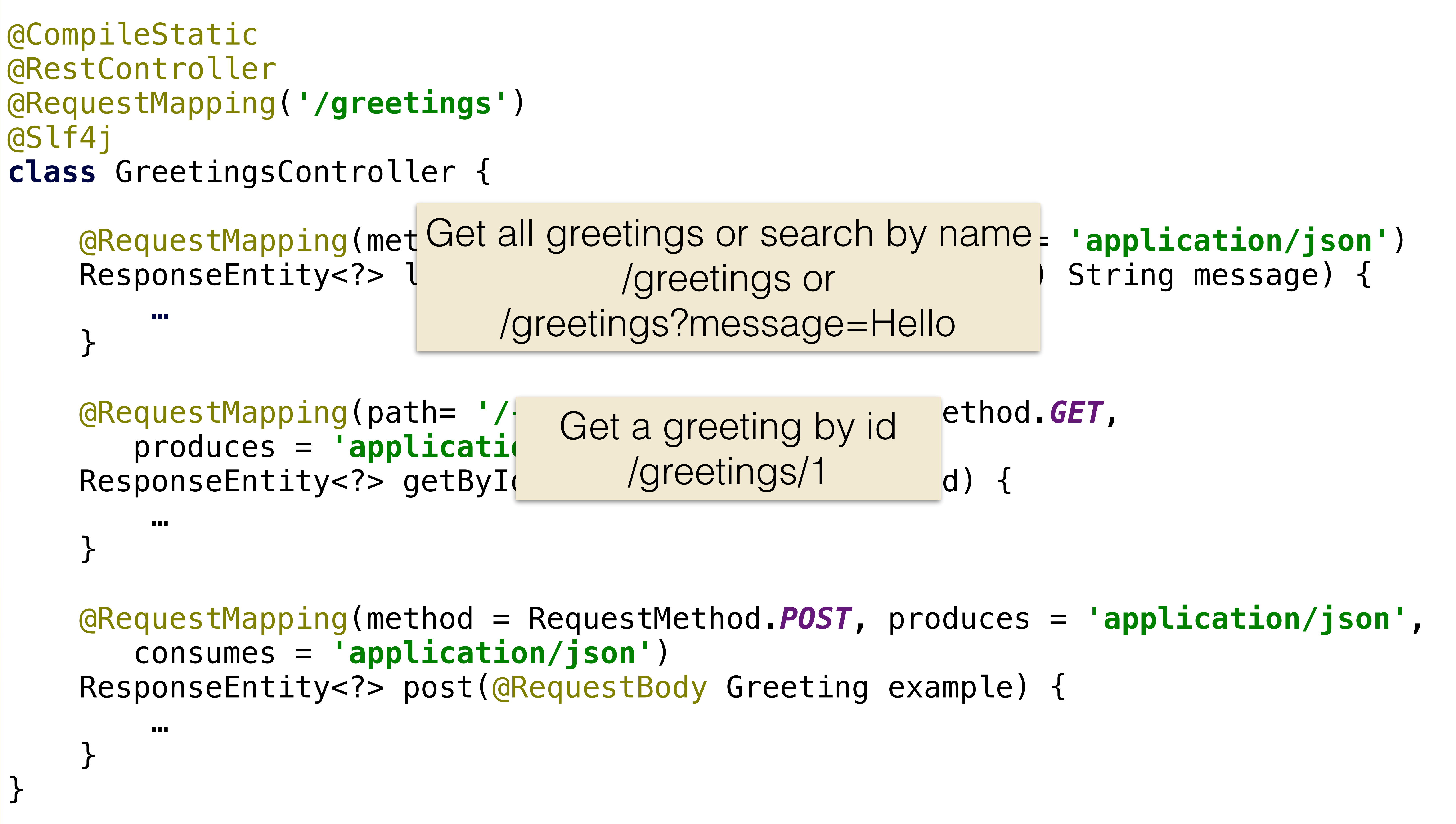
Endpoints



```
@CompileStatic
@RestController
@RequestMapping('/greetings')
@Slf4j
class GreetingsController {

    @RequestMapping(met                                   = 'application/json')
    ResponseEntity<?> l                                   ) String message) {
        …
    }

    @RequestMapping(path= '/{id}', method = RequestMethod.GET,
       produces = 'application/json')
    ResponseEntity<?> getById(@PathVariable String id) {
        …
    }

    @RequestMapping(method = RequestMethod.POST, produces = 'application/json',
       consumes = 'application/json')
    ResponseEntity<?> post(@RequestBody Greeting example) {
        …
    }
}
```

Get all greetings or search by name
/greetings or
/greetings?message=Hello

Endpoints

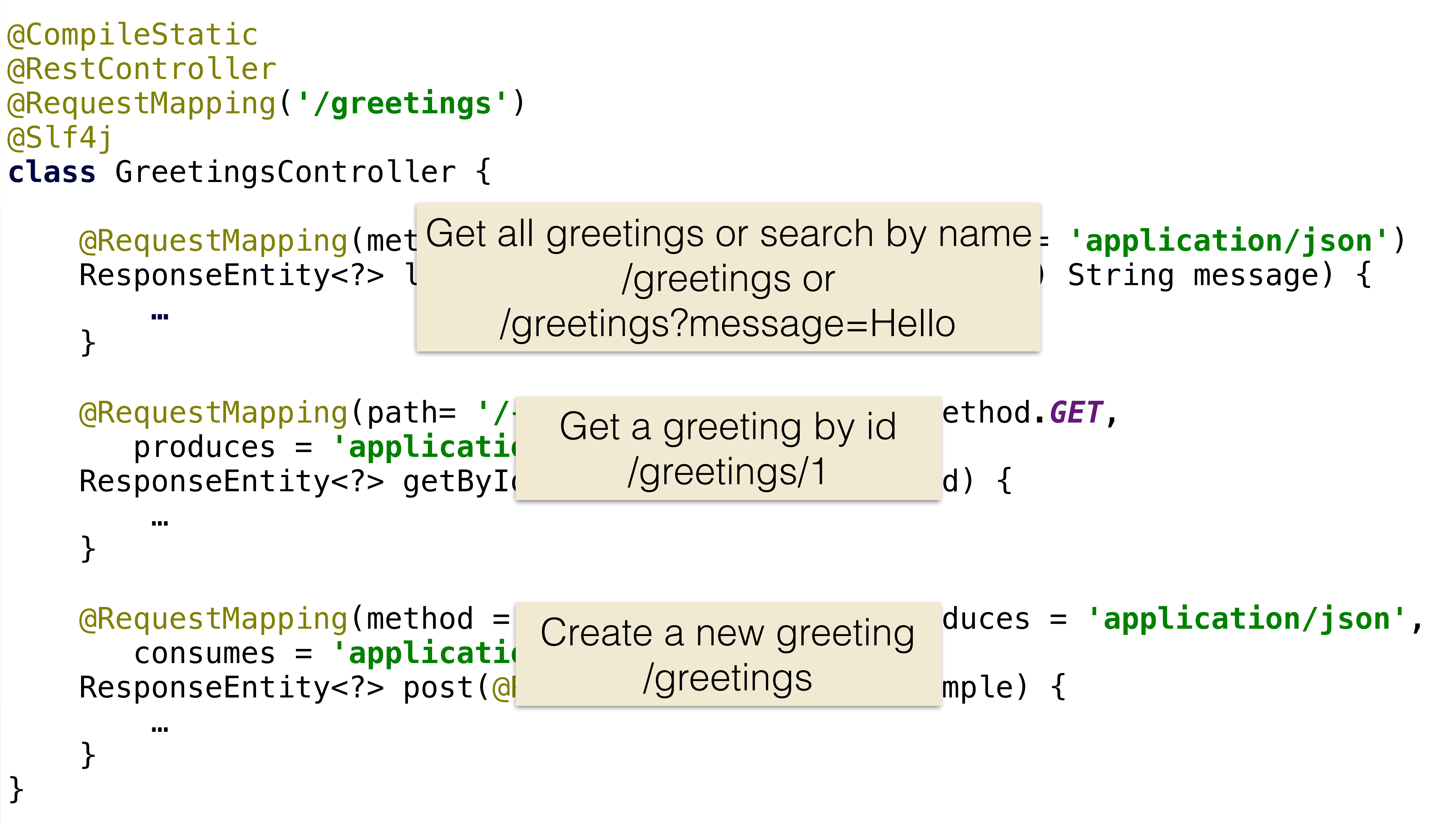


```
@CompileStatic
@RestController
@RequestMapping('/greetings')
@Slf4j
class GreetingsController {

    @RequestMapping(met                                  = 'application/json')
    ResponseEntity<?> l                                  ) String message) {
        …
    }

    @RequestMapping(path= '/                       ethod.GET,
        produces = 'applicati
    ResponseEntity<?> getByI                       d) {
        …
    }

    @RequestMapping(method = RequestMethod.POST, produces = 'application/json',
        consumes = 'application/json')
    ResponseEntity<?> post(@RequestBody Greeting example) {
        …
    }
}
```

Get all greetings or search by name
/greetings or
/greetings?message=Hello

Get a greeting by id
/greetings/1

Endpoints



```
@CompileStatic
@RestController
@RequestMapping('/greetings')
@Slf4j
class GreetingsController {

    @RequestMapping(met                                   = 'application/json')
    ResponseEntity<?> l                                   ) String message) {
        …
    }

    @RequestMapping(path= '/                         ethod.GET,
        produces = 'applicati
    ResponseEntity<?> getByI                         d) {
        …
    }

    @RequestMapping(method =                         duces = 'application/json',
        consumes = 'applicati
    ResponseEntity<?> post(@                         mple) {
        …
    }
}
```

Get all greetings or search by name
/greetings or
/greetings?message=Hello

Get a greeting by id
/greetings/1

Create a new greeting
/greetings

@codeJENNerator

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.
```

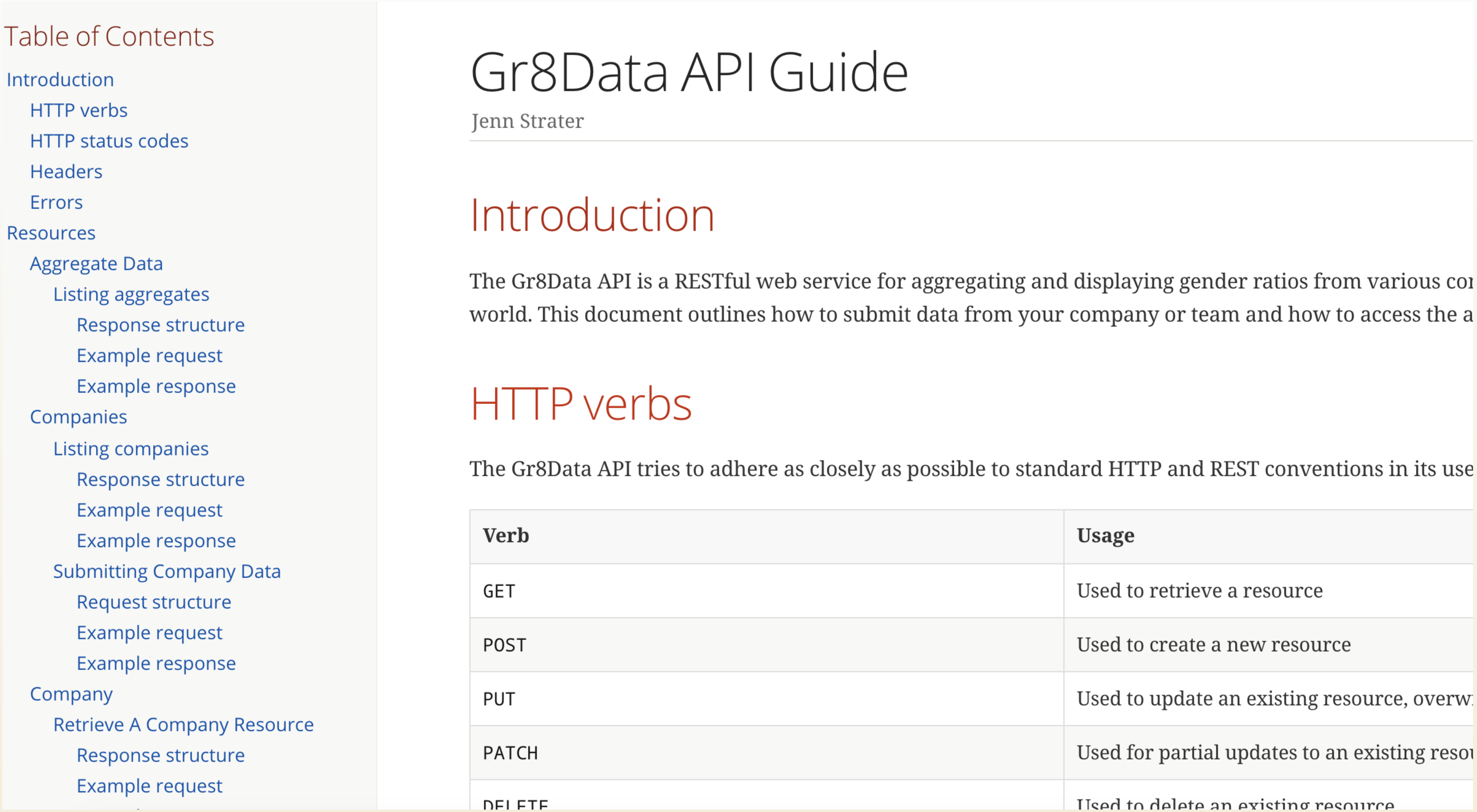


Table of Contents

Gr8Data API Guide

Jenn Strater

Introduction

The Gr8Data API is a RESTful web service for aggregating and displaying gender ratios from various cor 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 resou
`DELETE`	Used to delete an existing resource

Converted to HTML

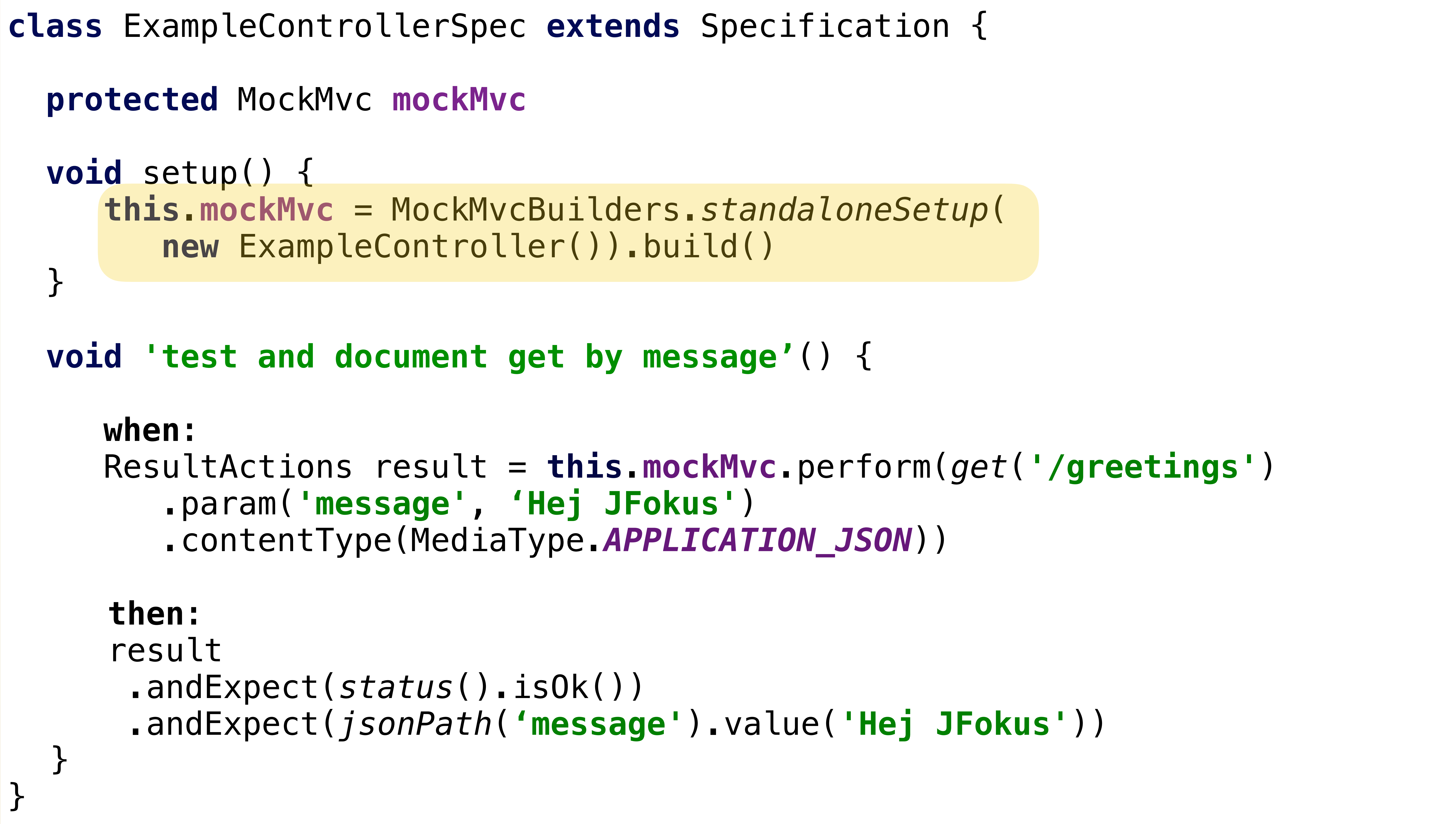
@codeJENNerator

SPRING
MVC
TEST

MockMvc Test

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

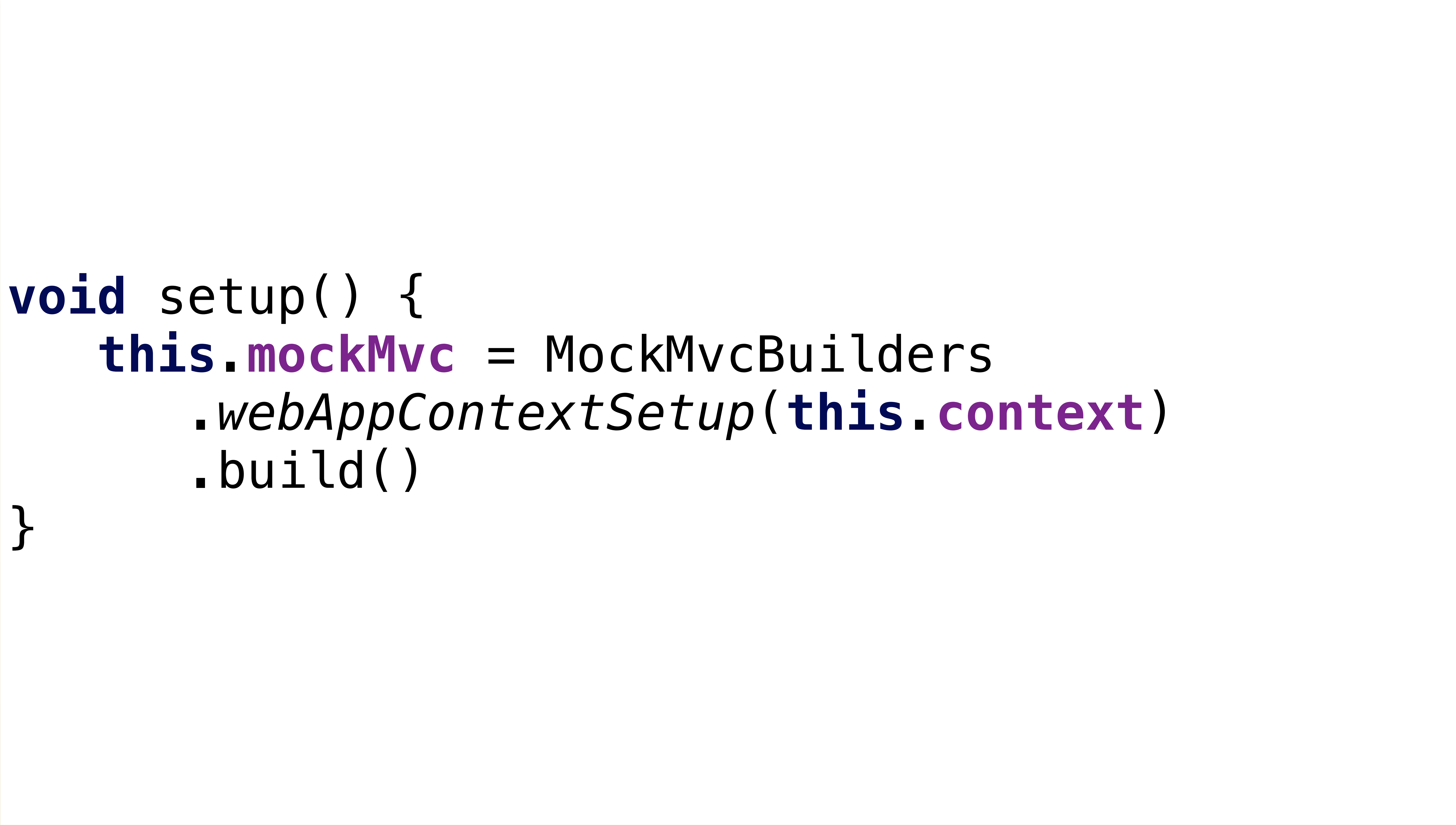
Stand Alone Setup



```
class ExampleControllerSpec extends Specification {

  protected MockMvc mockMvc

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

  void 'test and document get by message’() {

     when:
     ResultActions result = this.mockMvc.perform(get('/greetings')
        .param('message', ‘Hej JFokus')
        .contentType(MediaType.APPLICATION_JSON))

     then:
     result
      .andExpect(status().isOk())
      .andExpect(jsonPath(‘message').value('Hej JFokus'))
  }
}
```

@codeJENNerator

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!!

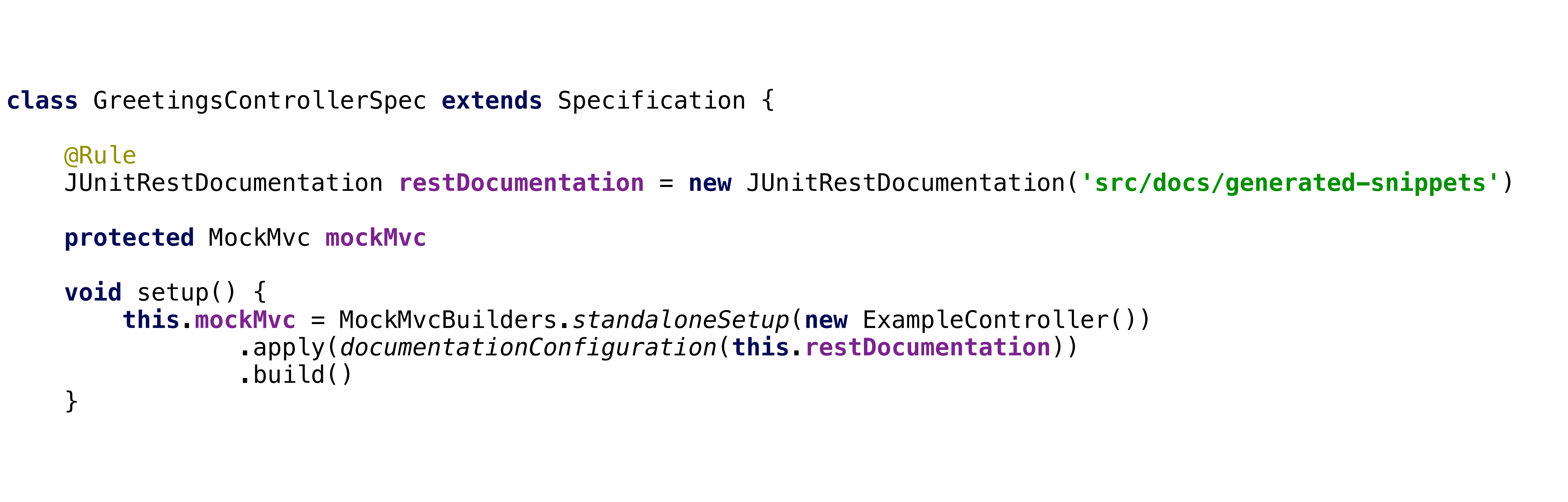
Spring REST docs

Gradle

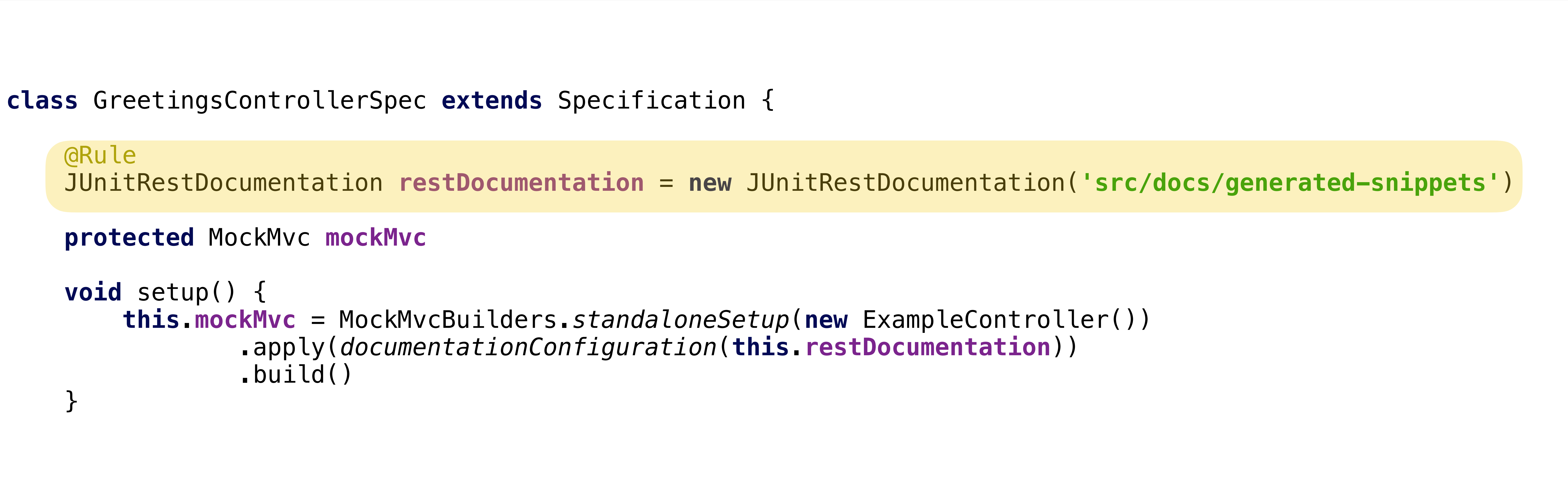
```
ext {
    snippetsDir = file('src/docs/generated-snippets')
}

testCompile "org.springframework.restdocs:spring-
restdocs-mockmvc:${springRestDocsVersion}"
```

Gradle

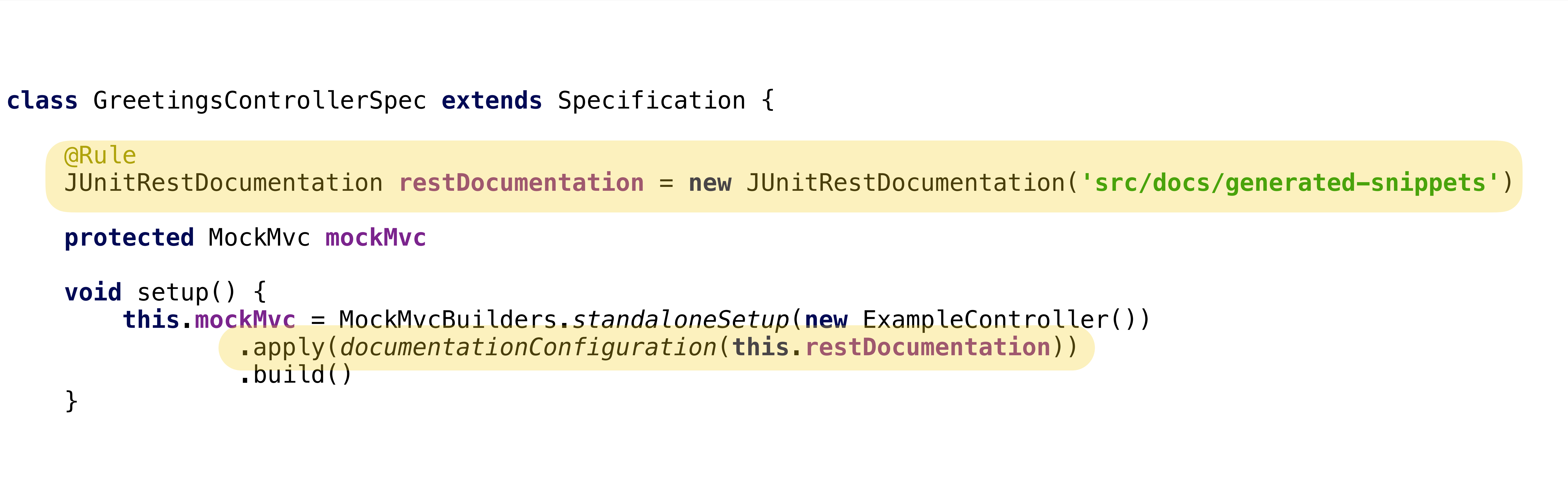
```
asciidoctor {
    dependsOn test
    sourceDir = file('src/docs')
    outputDir "$projectDir/src/main/resources/public"
+   inputs.dir snippetsDir
    backends 'html5'
    attributes 'source-highlighter' : 'prettify',
            'imagesdir':'images',
            'toc':'left',
            'icons':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



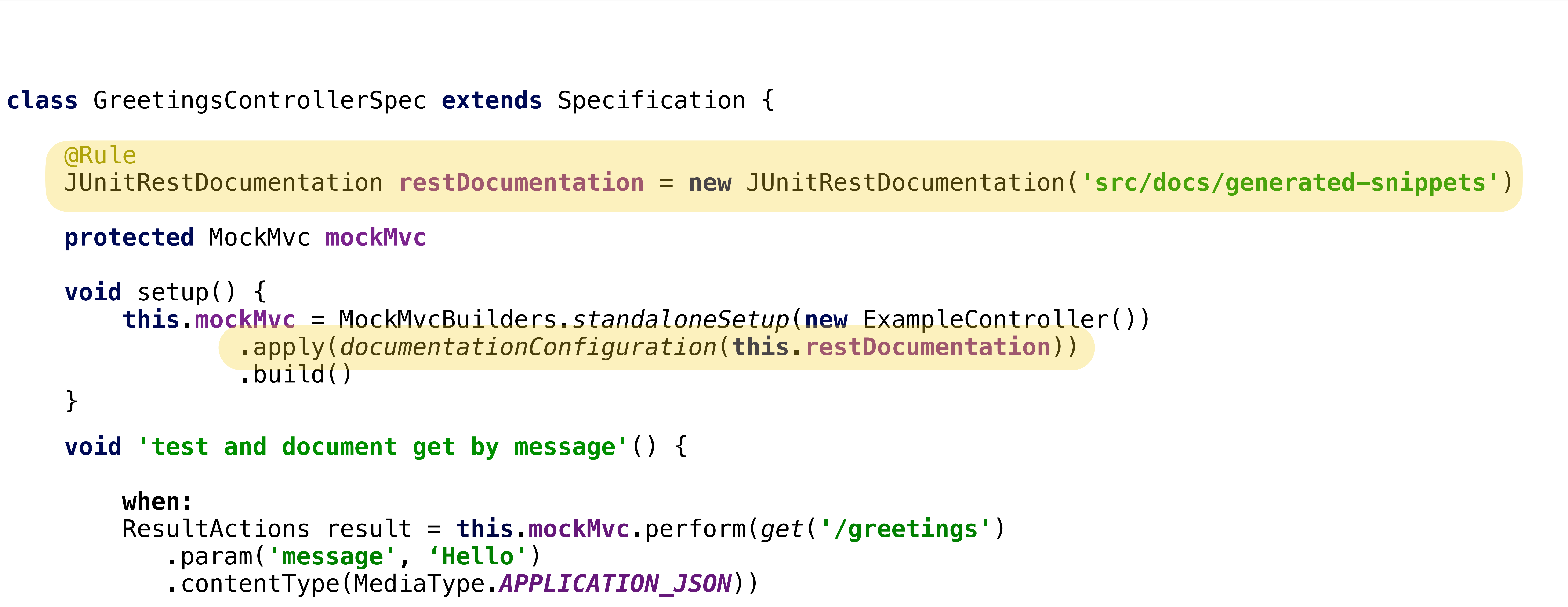
```
class GreetingsControllerSpec extends Specification {

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

    protected MockMvc mockMvc

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



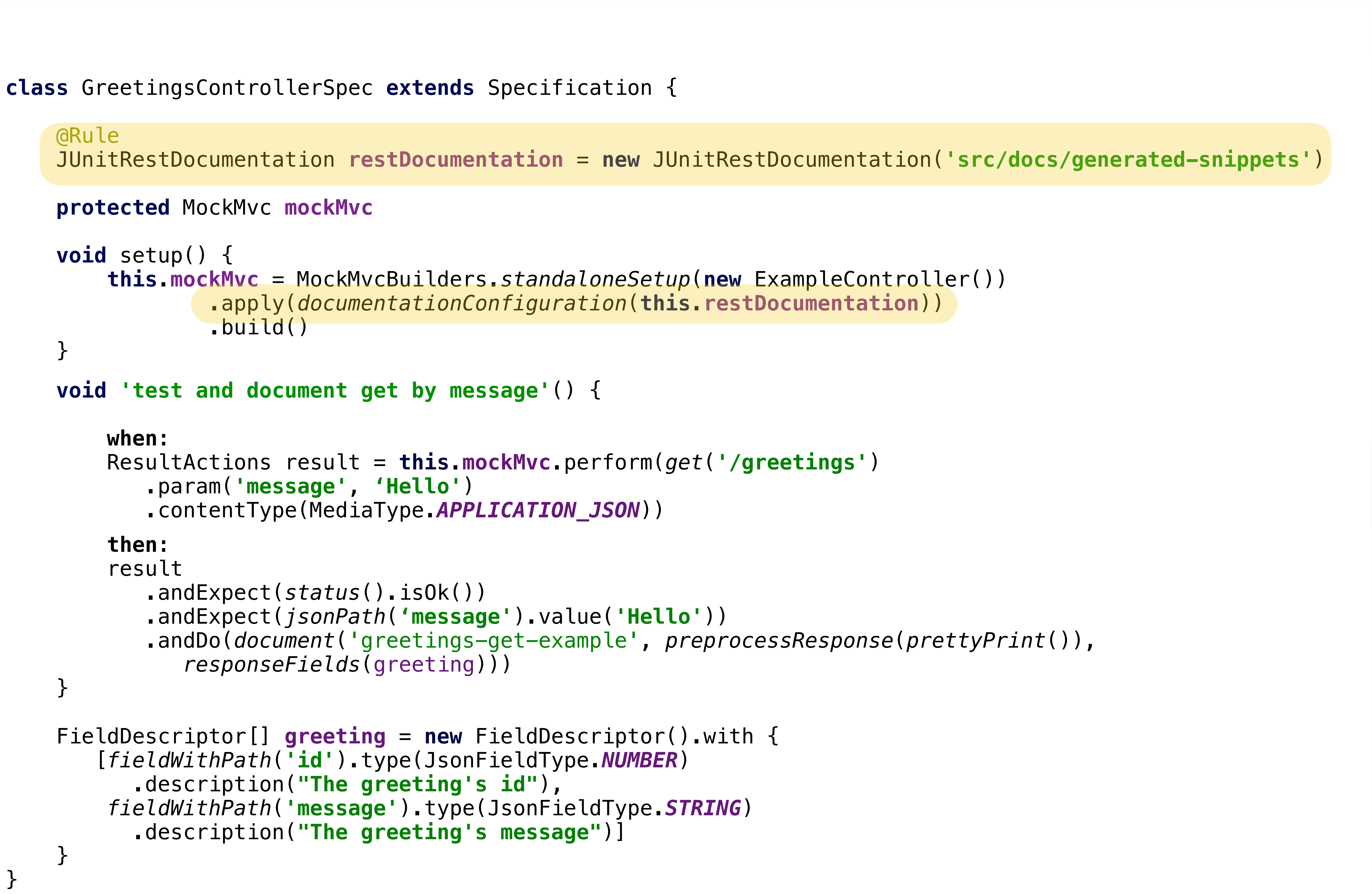
```
class GreetingsControllerSpec extends Specification {

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

    protected MockMvc mockMvc

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



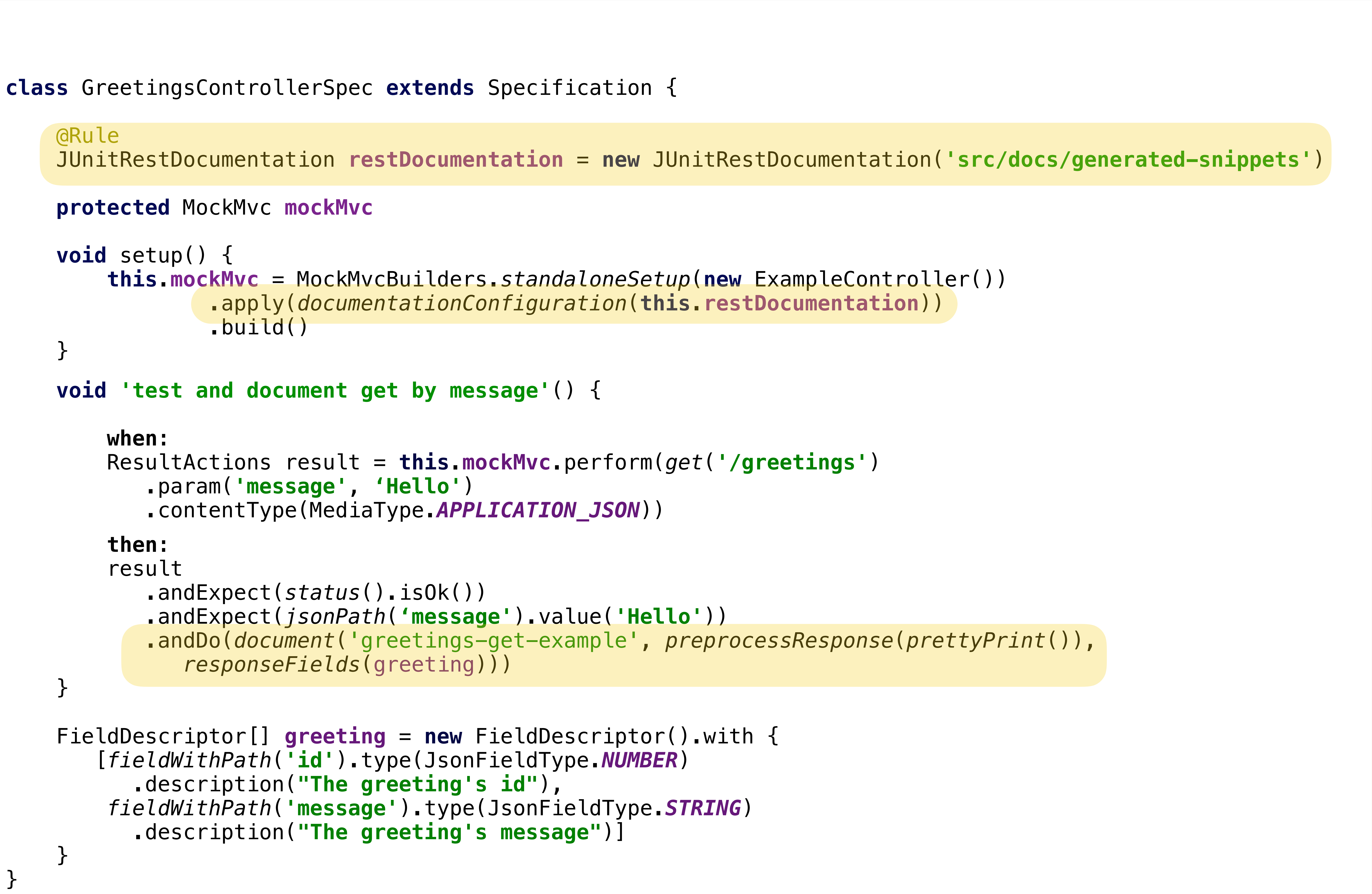
```
class GreetingsControllerSpec extends Specification {

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

    protected MockMvc mockMvc

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



```
class GreetingsControllerSpec extends Specification {

    @Rule
    JUnitRestDocumentation restDocumentation = new JUnitRestDocumentation('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 by message'() {

        when:
        ResultActions result = this.mockMvc.perform(get('/greetings')
           .param('message', ‘Hello')
           .contentType(MediaType.APPLICATION_JSON))
```

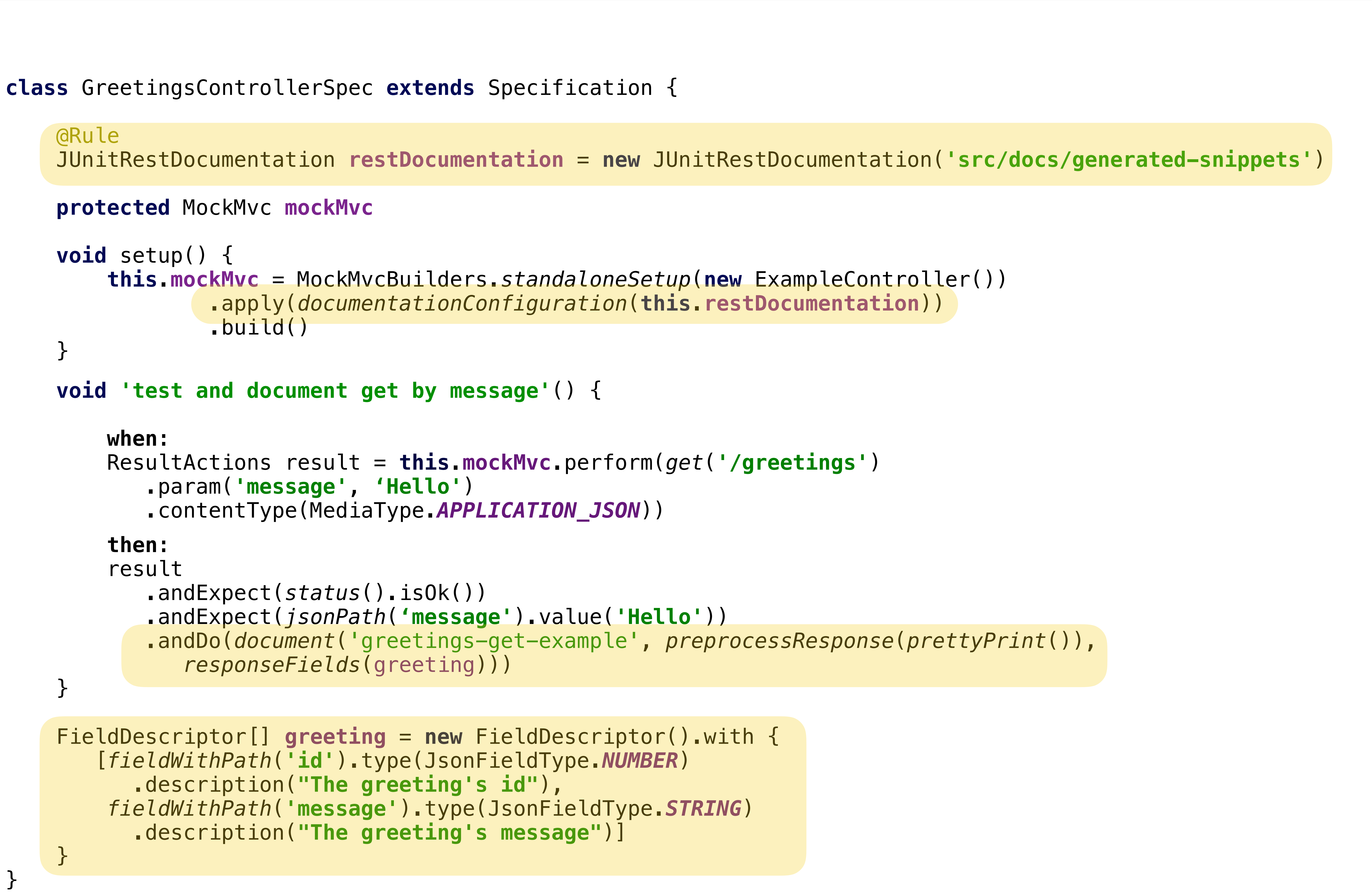


```
class GreetingsControllerSpec extends Specification {

    @Rule
    JUnitRestDocumentation restDocumentation = new JUnitRestDocumentation('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 by message'() {

        when:
        ResultActions result = this.mockMvc.perform(get('/greetings')
            .param('message', ‘Hello')
            .contentType(MediaType.APPLICATION_JSON))

        then:
        result
            .andExpect(status().isOk())
            .andExpect(jsonPath(‘message').value('Hello'))
            .andDo(document('greetings-get-example', preprocessResponse(prettyPrint()),
               responseFields(greeting)))
    }

    FieldDescriptor[] greeting = new FieldDescriptor().with {
       [fieldWithPath('id').type(JsonFieldType.NUMBER)
          .description("The greeting's id"),
        fieldWithPath('message').type(JsonFieldType.STRING)
          .description("The greeting's message")]
    }
}
```



```
class GreetingsControllerSpec extends Specification {

    @Rule
    JUnitRestDocumentation restDocumentation = new JUnitRestDocumentation('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 by message'() {

        when:
        ResultActions result = this.mockMvc.perform(get('/greetings')
            .param('message', ‘Hello')
            .contentType(MediaType.APPLICATION_JSON))

        then:
        result
            .andExpect(status().isOk())
            .andExpect(jsonPath(‘message').value('Hello'))
            .andDo(document('greetings-get-example', preprocessResponse(prettyPrint()),
               responseFields(greeting)))
    }

    FieldDescriptor[] greeting = new FieldDescriptor().with {
        [fieldWithPath('id').type(JsonFieldType.NUMBER)
           .description("The greeting's id"),
        fieldWithPath('message').type(JsonFieldType.STRING)
           .description("The greeting's message")]
    }
}
```

@codeJENNerator



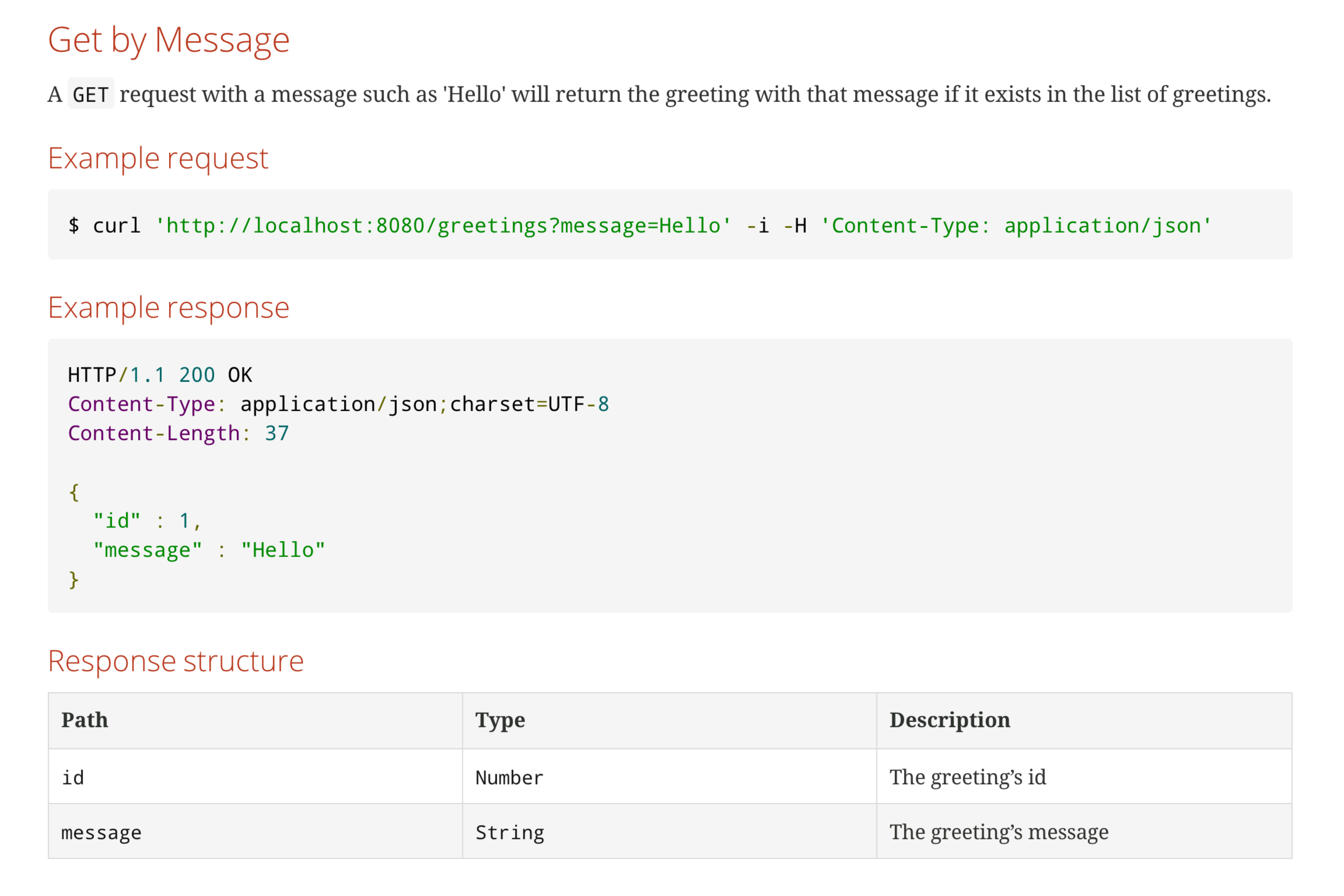
```
class GreetingsControllerSpec extends Specification {

    @Rule
    JUnitRestDocumentation restDocumentation = new JUnitRestDocumentation('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 by message'() {

        when:
        ResultActions result = this.mockMvc.perform(get('/greetings')
           .param('message', ‘Hello')
           .contentType(MediaType.APPLICATION_JSON))

        then:
        result
           .andExpect(status().isOk())
           .andExpect(jsonPath(‘message').value('Hello'))
           .andDo(document('greetings-get-example', preprocessResponse(prettyPrint()),
              responseFields(greeting)))
    }

    FieldDescriptor[] greeting = new FieldDescriptor().with {
       [fieldWithPath('id').type(JsonFieldType.NUMBER)
          .description("The greeting's id"),
        fieldWithPath('message').type(JsonFieldType.STRING)
          .description("The greeting's message")]
    }
}
```

@codeJENNerator

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



Get by Message

A `GET` request with a message such as 'Hello' will return the greeting with that message if it exists in the list of greetings.

Example request

```
$ curl 'http://localhost:8080/greetings?message=Hello' -i -H 'Content-Type: application/json'
```

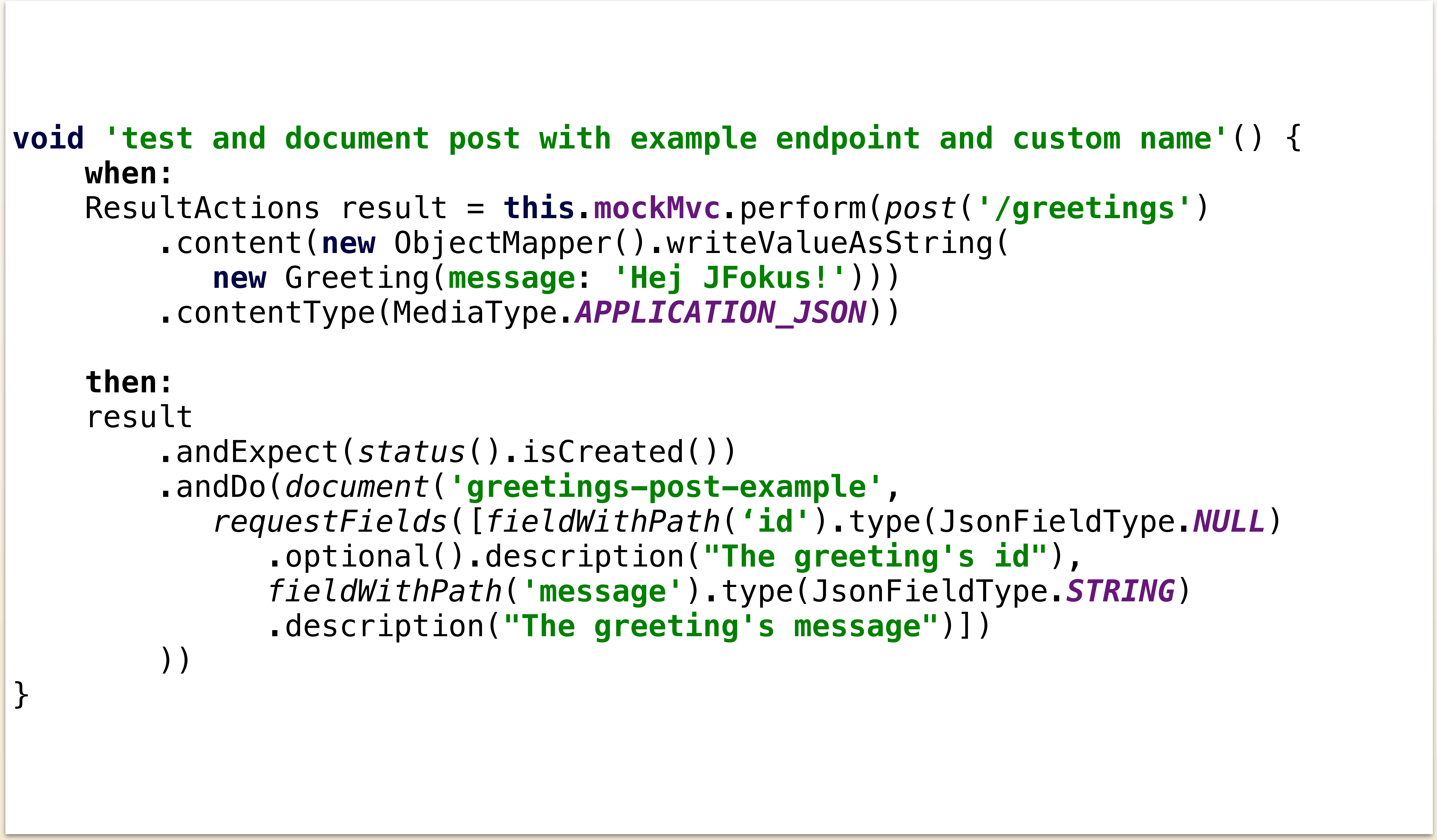
Example response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 37

{
  "id" : 1,
  "message" : "Hello"
}
```

Response structure

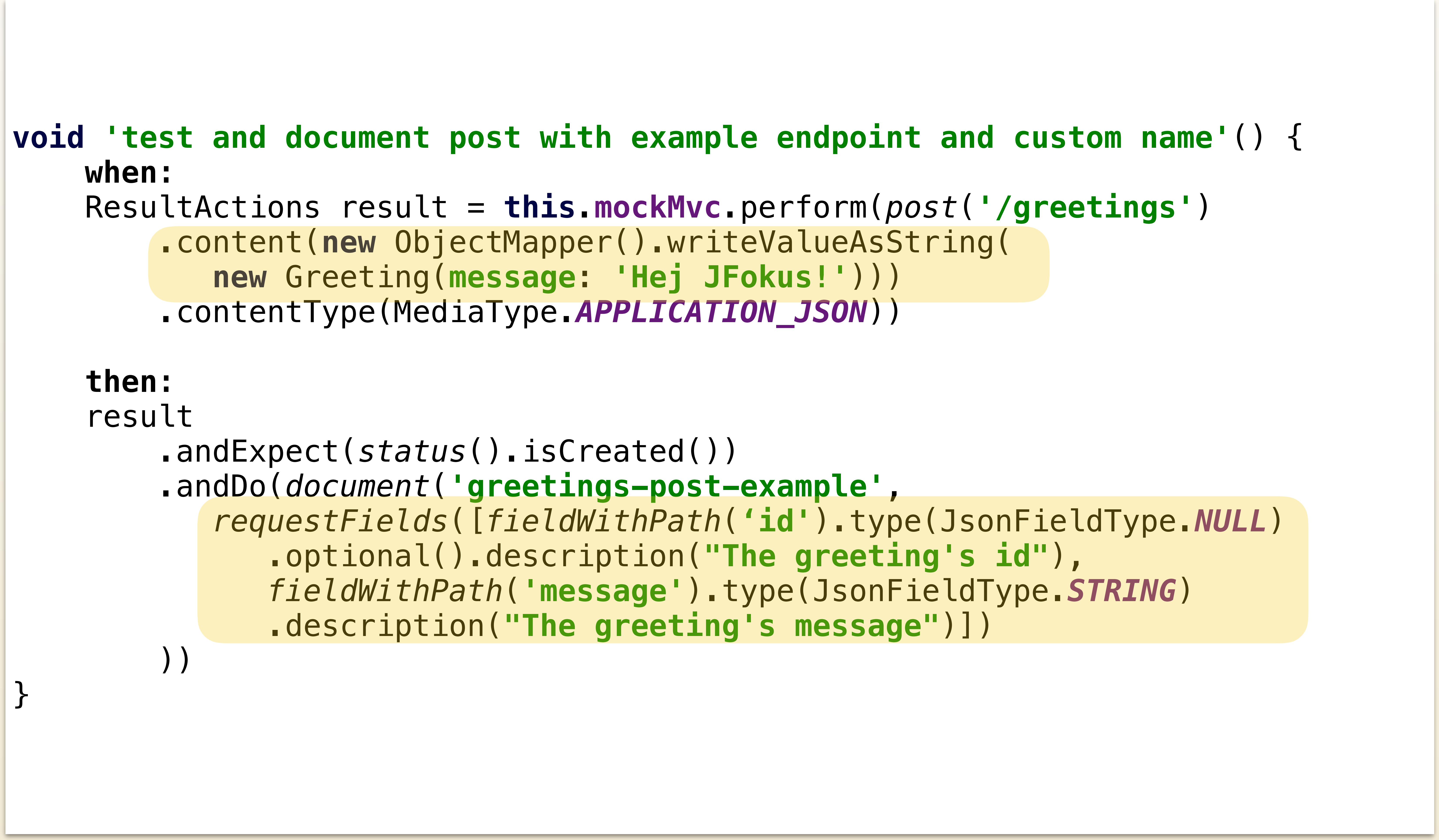
Path	**Type**	**Description**
`id`	`Number`	The greeting’s id
`message`	`String`	The greeting’s message

Create a new greeting

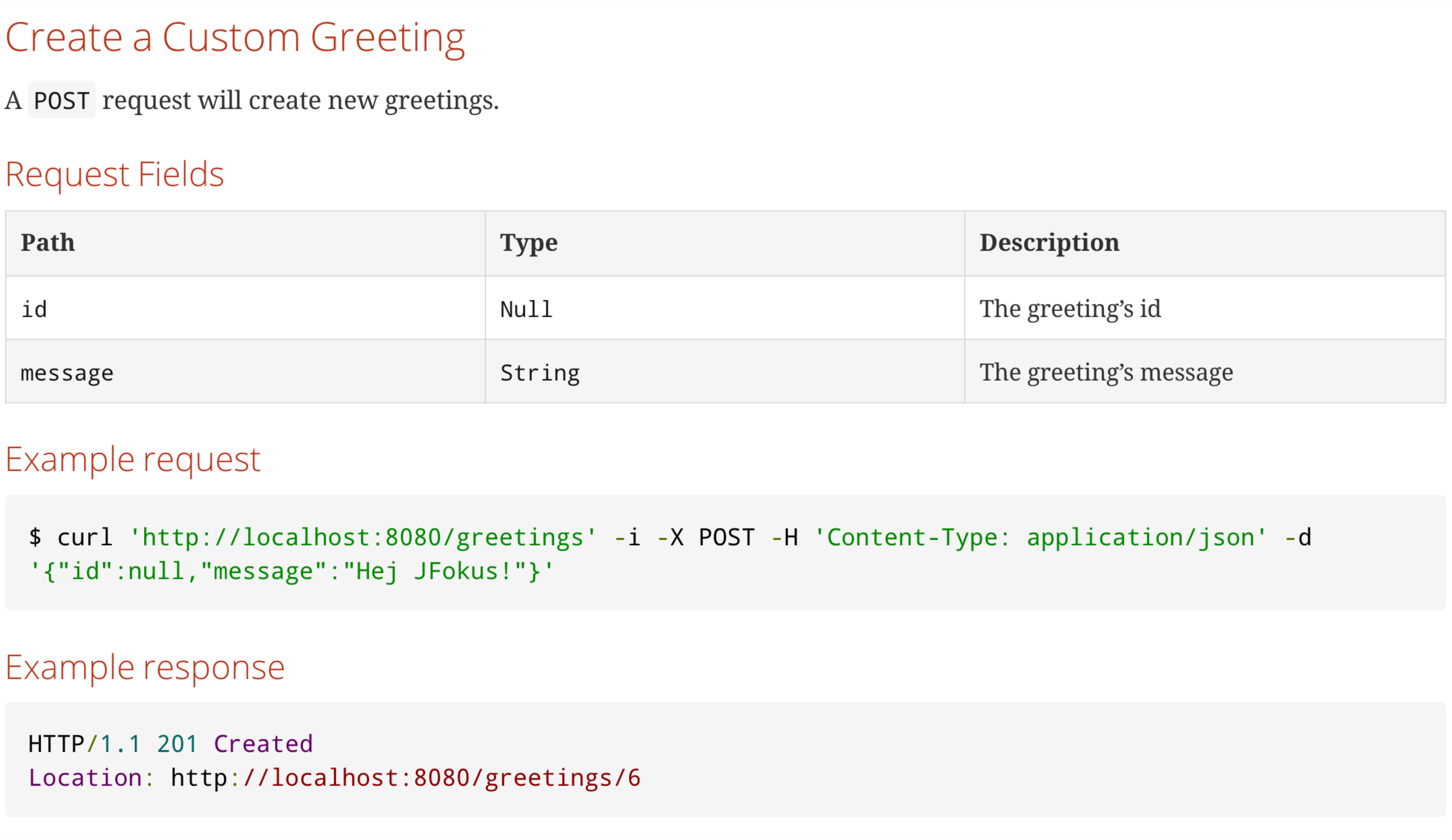


```
void 'test and document post with example endpoint and custom name'() {
    when:
    ResultActions result = this.mockMvc.perform(post('/greetings')
        .content(new ObjectMapper().writeValueAsString(
           new Greeting(message: 'Hej JFokus!')))
        .contentType(MediaType.APPLICATION_JSON))

    then:
    result
        .andExpect(status().isCreated())
        .andDo(document('greetings-post-example',
           requestFields([fieldWithPath(‘id').type(JsonFieldType.NULL)
              .optional().description("The greeting's id"),
              fieldWithPath('message').type(JsonFieldType.STRING)
              .description("The greeting's message")])
        ))
}
```

Create a new greeting



```
void 'test and document post with example endpoint and custom name'() {
    when:
    ResultActions result = this.mockMvc.perform(post('/greetings')
        .content(new ObjectMapper().writeValueAsString(
           new Greeting(message: 'Hej JFokus!')))
        .contentType(MediaType.APPLICATION_JSON))

    then:
    result
        .andExpect(status().isCreated())
        .andDo(document('greetings-post-example',
           requestFields([fieldWithPath(‘id').type(JsonFieldType.NULL)
              .optional().description("The greeting's id"),
              fieldWithPath('message').type(JsonFieldType.STRING)
              .description("The greeting's message")])
        ))
}
```



Create a Custom Greeting

A `POST` request will create new greetings.

Request Fields

Path	Type	Description
`id`	`Null`	The greeting’s id
`message`	`String`	The greeting’s message

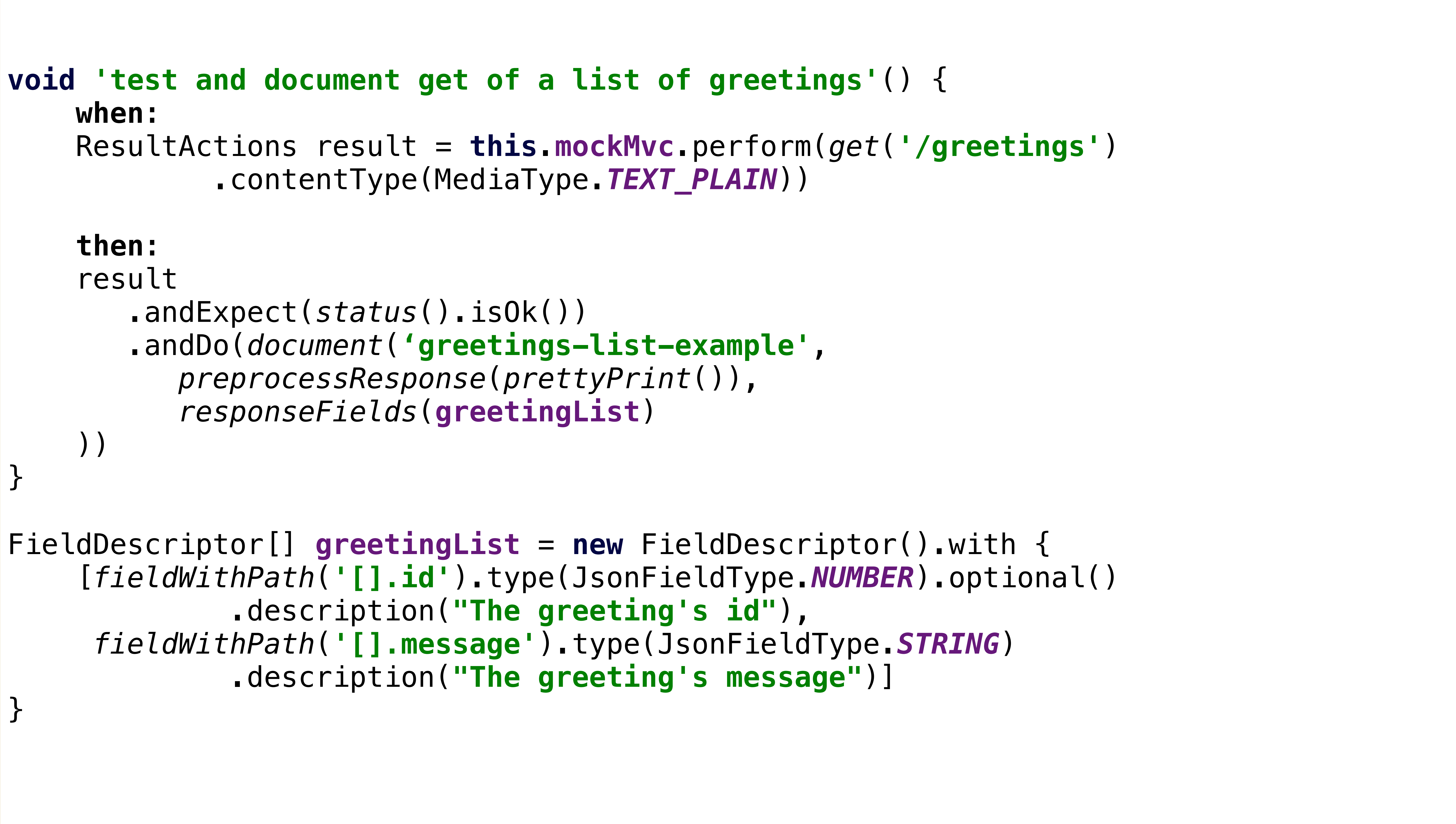
Example request

```
$ curl 'http://localhost:8080/greetings' -i -X POST -H 'Content-Type: application/json' -d
'{"id":null,"message":"Hej JFokus!"}'
```

Example response

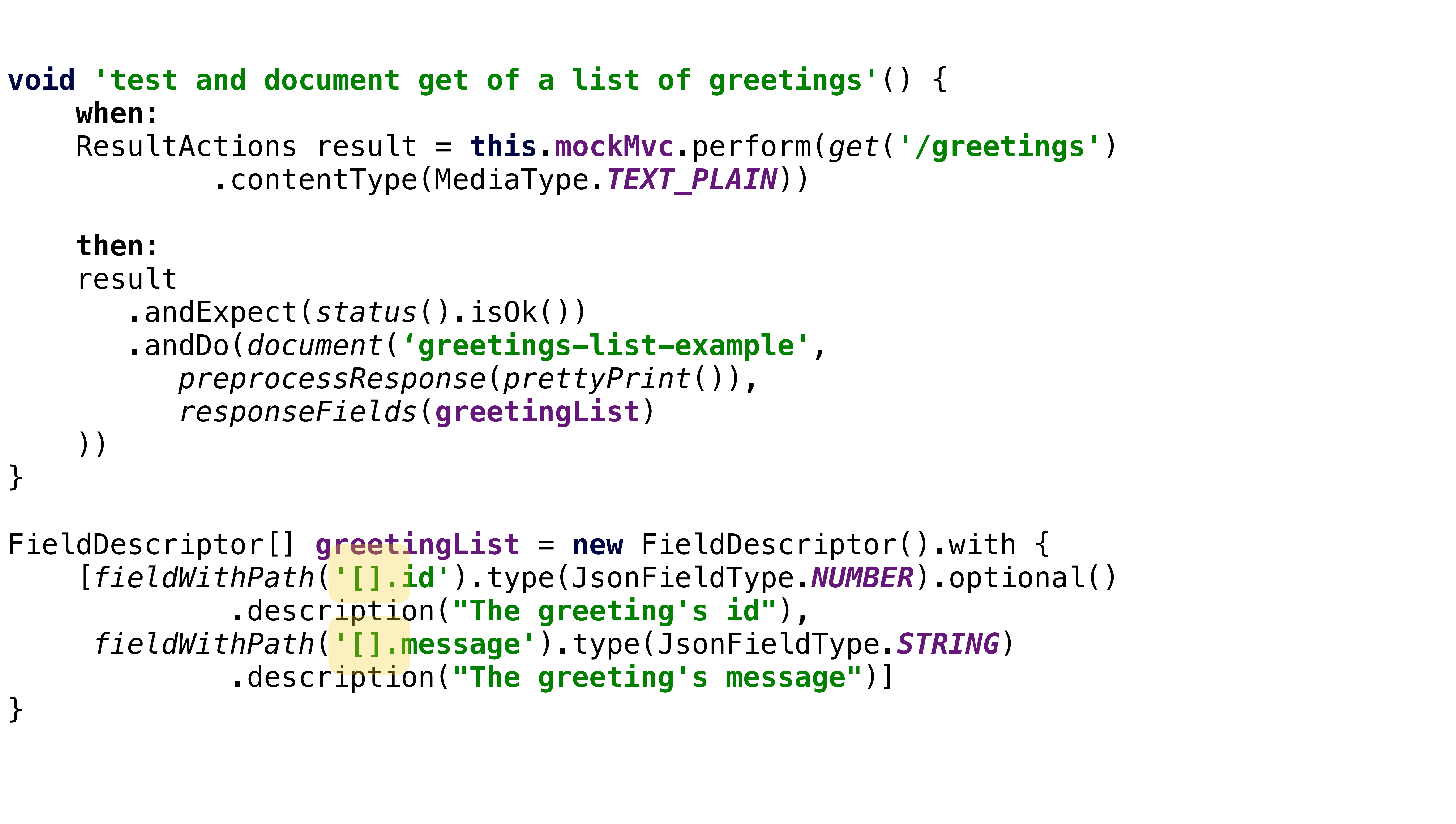
```
HTTP/1.1 201 Created
Location: http://localhost:8080/greetings/6
```

List Greetings



```
void 'test and document get of a list of greetings'() {
    when:
    ResultActions result = this.mockMvc.perform(get('/greetings')
            .contentType(MediaType.TEXT_PLAIN))

    then:
    result
       .andExpect(status().isOk())
       .andDo(document(‘greetings-list-example',
          preprocessResponse(prettyPrint()),
          responseFields(greetingList)
    ))
}

FieldDescriptor[] greetingList = new FieldDescriptor().with {
    [fieldWithPath('[].id').type(JsonFieldType.NUMBER).optional()
             .description("The greeting's id"),
     fieldWithPath('[].message').type(JsonFieldType.STRING)
             .description("The greeting's message")]
}
```

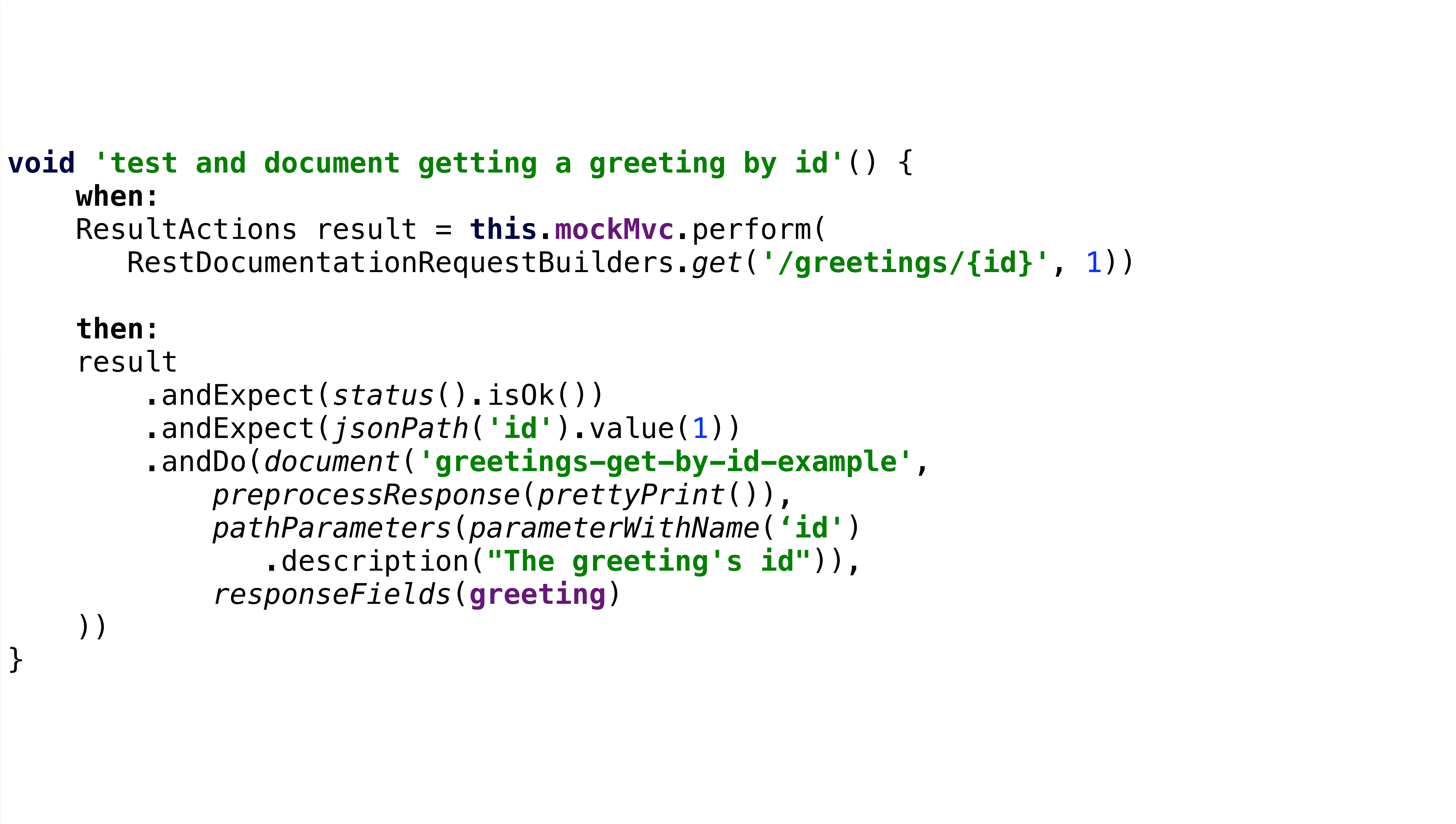
List Greetings



```
void 'test and document get of a list of greetings'() {
    when:
    ResultActions result = this.mockMvc.perform(get('/greetings')
            .contentType(MediaType.TEXT_PLAIN))

    then:
    result
       .andExpect(status().isOk())
       .andDo(document(‘greetings-list-example',
          preprocessResponse(prettyPrint()),
          responseFields(greetingList)
    ))
}

FieldDescriptor[] greetingList = new FieldDescriptor().with {
    [fieldWithPath('[].id').type(JsonFieldType.NUMBER).optional()
             .description("The greeting's id"),
     fieldWithPath('[].message').type(JsonFieldType.STRING)
             .description("The greeting's message")]
}
```

@codeJENNerator

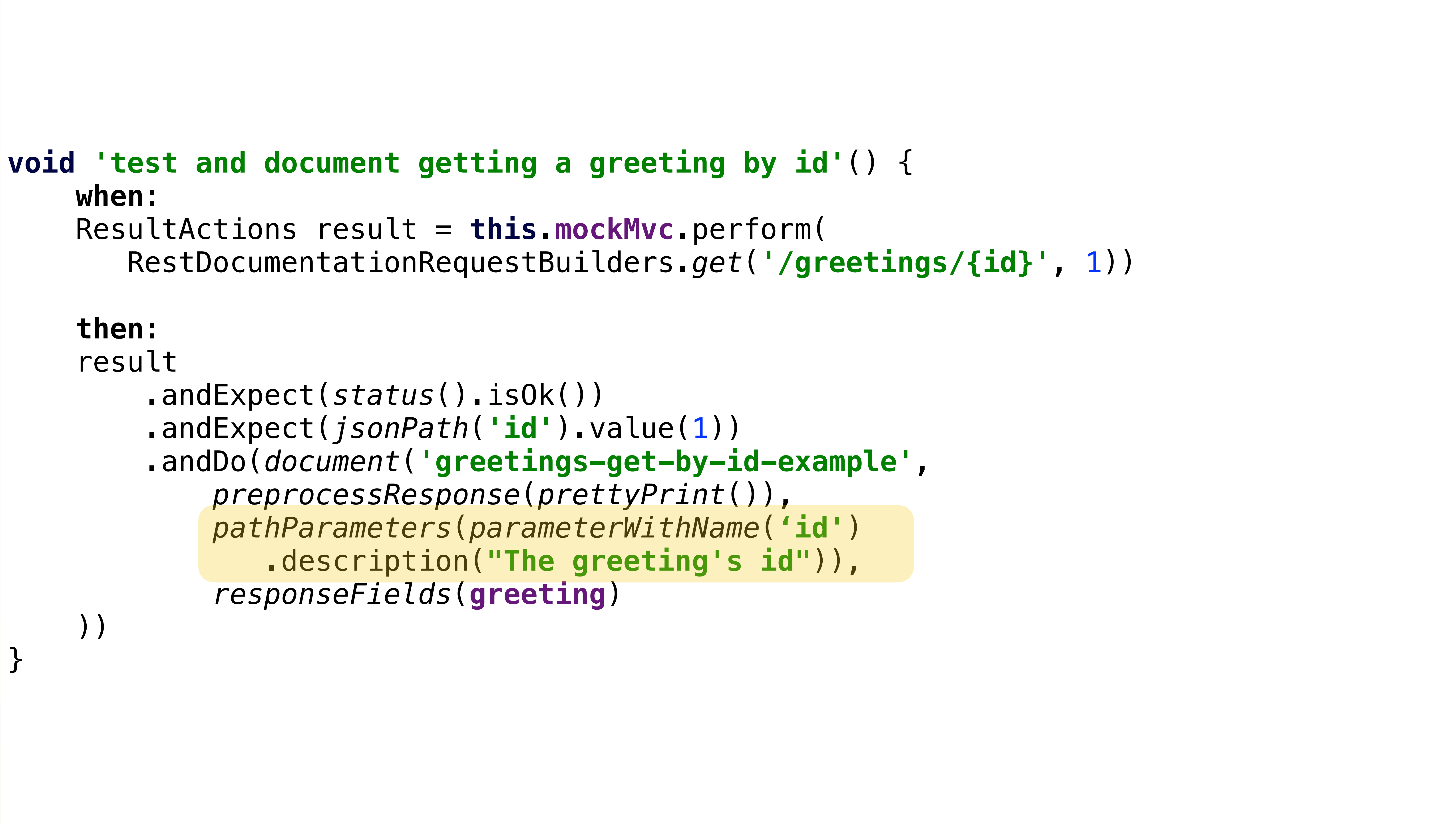
Path Parameters



```
void 'test and document getting a greeting by id'() {
    when:
    ResultActions result = this.mockMvc.perform(
       RestDocumentationRequestBuilders.get('/greetings/{id}', 1))

    then:
    result
        .andExpect(status().isOk())
        .andExpect(jsonPath('id').value(1))
        .andDo(document('greetings-get-by-id-example',
            preprocessResponse(prettyPrint()),
            pathParameters(parameterWithName(‘id')
               .description("The greeting's id")),
            responseFields(greeting)
    ))
}
```

@codeJENNerator

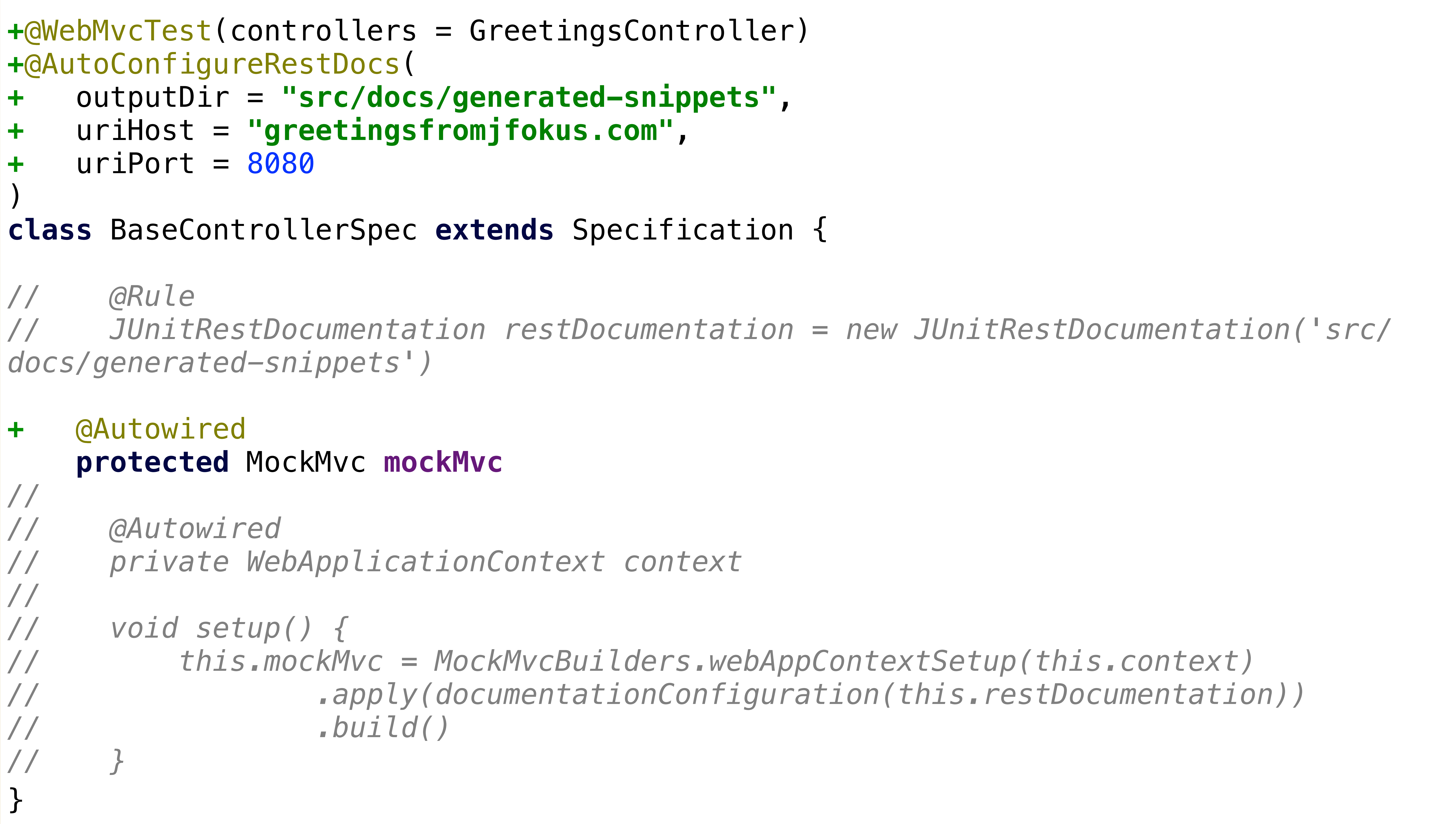
Path Parameters



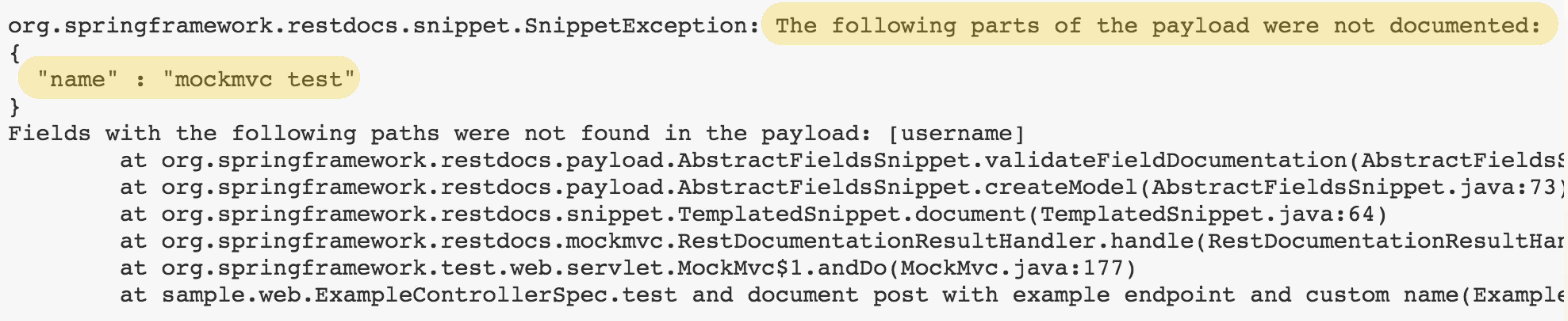
```
void 'test and document getting a greeting by id'() {
    when:
    ResultActions result = this.mockMvc.perform(
       RestDocumentationRequestBuilders.get('/greetings/{id}', 1))

    then:
    result
        .andExpect(status().isOk())
        .andExpect(jsonPath('id').value(1))
        .andDo(document('greetings-get-by-id-example',
            preprocessResponse(prettyPrint()),
            pathParameters(parameterWithName(‘id')
               .description("The greeting's id")),
            responseFields(greeting)
    ))
}
```

@codeJENNerator

Setup Via Annotation

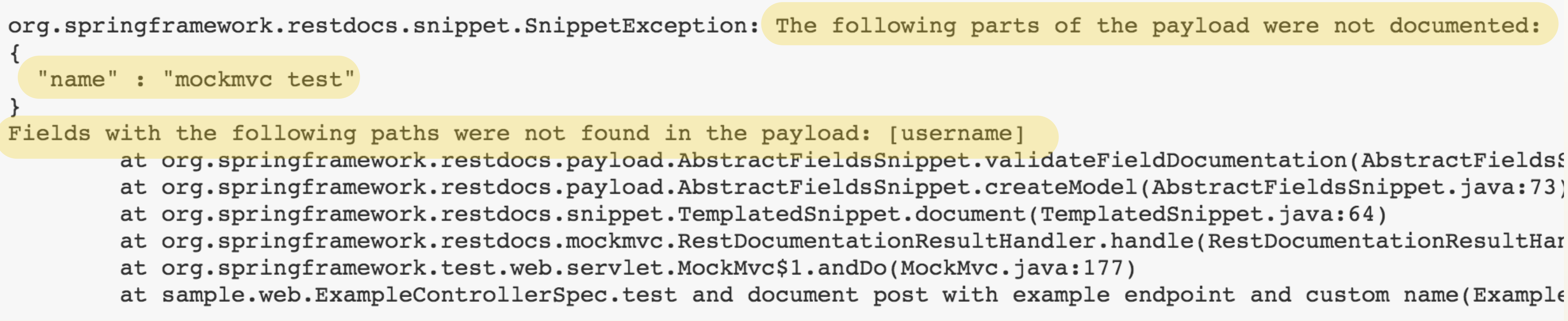


```
+@WebMvcTest(controllers = GreetingsController)
+@AutoConfigureRestDocs(
+   outputDir = "src/docs/generated-snippets",
+   uriHost = "greetingsfromjfokus.com",
+   uriPort = 8080
)
class BaseControllerSpec extends Specification {

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

+   @Autowired
    protected MockMvc mockMvc
//
//    @Autowired
//    private WebApplicationContext context
//
//    void setup() {
//        this.mockMvc = MockMvcBuilders.webAppContextSetup(this.context)
//                .apply(documentationConfiguration(this.restDocumentation))
//                .build()
//    }
}
```



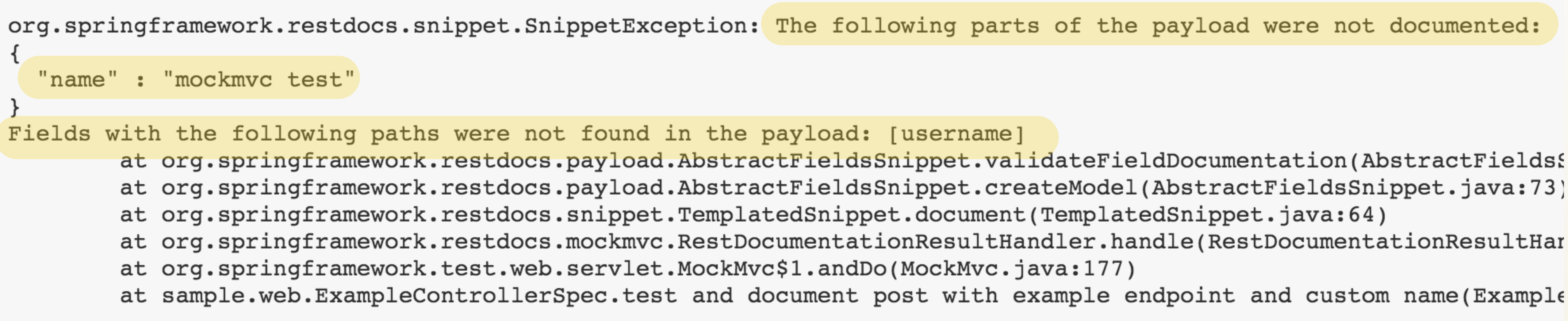
```
org.springframework.restdocs.snippet.SnippetException: The following parts of the payload were not documented:
{
  "name" : "mockmvc test"
}
Fields with the following paths were not found in the payload: [username]
        at org.springframework.restdocs.payload.AbstractFieldsSnippet.validateFieldDocumentation(AbstractFieldsS
        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(RestDocumentationResultHar
        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(Example
```

Failing Tests

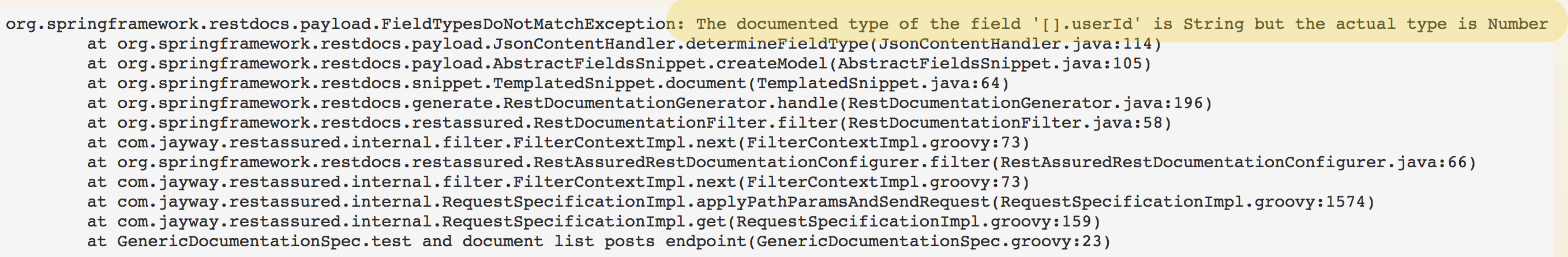


```
org.springframework.restdocs.snippet.SnippetException: The following parts of the payload were not documented:
{
  "name" : "mockmvc test"
}
Fields with the following paths were not found in the payload: [username]
        at org.springframework.restdocs.payload.AbstractFieldsSnippet.validateFieldDocumentation(AbstractFieldsS
        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(RestDocumentationResultHan
        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(Example
```

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(AbstractFieldsS
	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(RestDocumentationResultHar
	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(Example
```

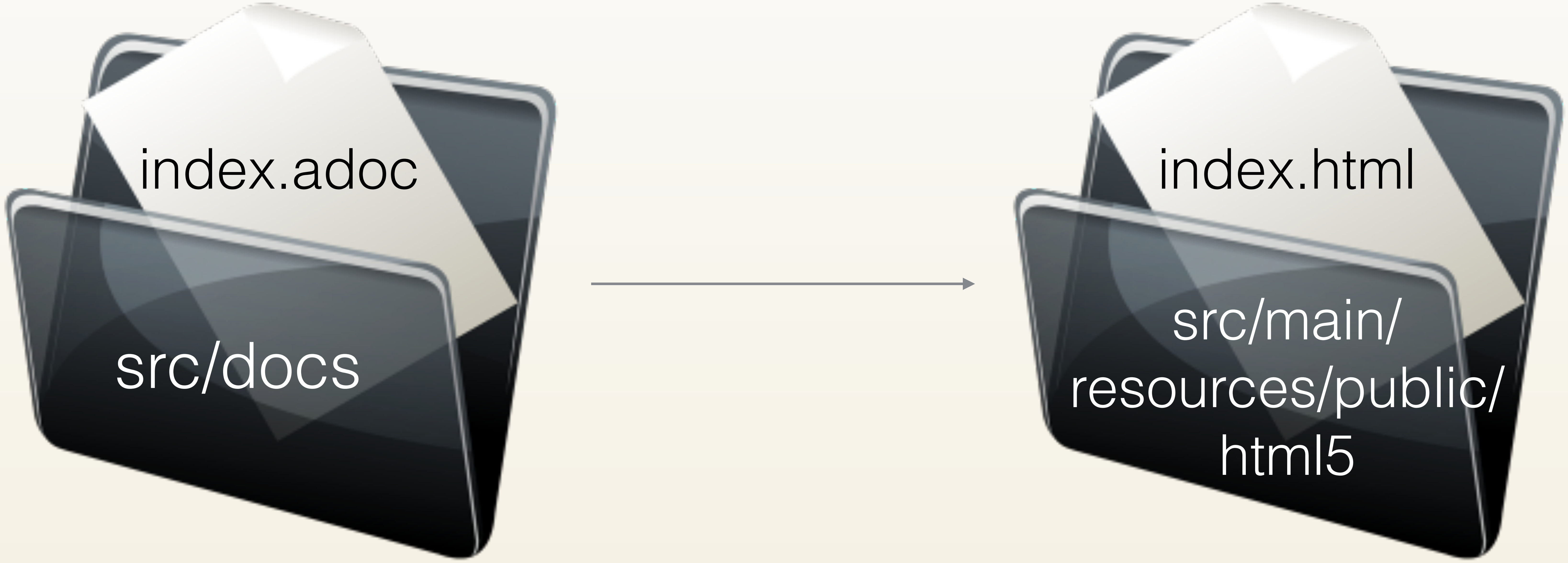


```
org.springframework.restdocs.payload.FieldTypesDoNotMatchException: The documented type of the field '[].userId' is String but the actual type is Number
	at org.springframework.restdocs.payload.JsonContentHandler.determineFieldType(JsonContentHandler.java:114)
	at org.springframework.restdocs.payload.AbstractFieldsSnippet.createModel(AbstractFieldsSnippet.java:105)
	at org.springframework.restdocs.snippet.TemplatedSnippet.document(TemplatedSnippet.java:64)
	at org.springframework.restdocs.generate.RestDocumentationGenerator.handle(RestDocumentationGenerator.java:196)
	at org.springframework.restdocs.restassured.RestDocumentationFilter.filter(RestDocumentationFilter.java:58)
	at com.jayway.restassured.internal.filter.FilterContextImpl.next(FilterContextImpl.groovy:73)
	at org.springframework.restdocs.restassured.RestAssuredRestDocumentationConfigurer.filter(RestAssuredRestDocumentationConfigurer.java:66)
	at com.jayway.restassured.internal.filter.FilterContextImpl.next(FilterContextImpl.groovy:73)
	at com.jayway.restassured.internal.RequestSpecificationImpl.applyPathParamsAndSendRequest(RequestSpecificationImpl.groovy:1574)
	at com.jayway.restassured.internal.RequestSpecificationImpl.get(RequestSpecificationImpl.groovy:159)
	at GenericDocumentationSpec.test and document list posts endpoint(GenericDocumentationSpec.groovy:23)
```

Failing Tests

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[]
```



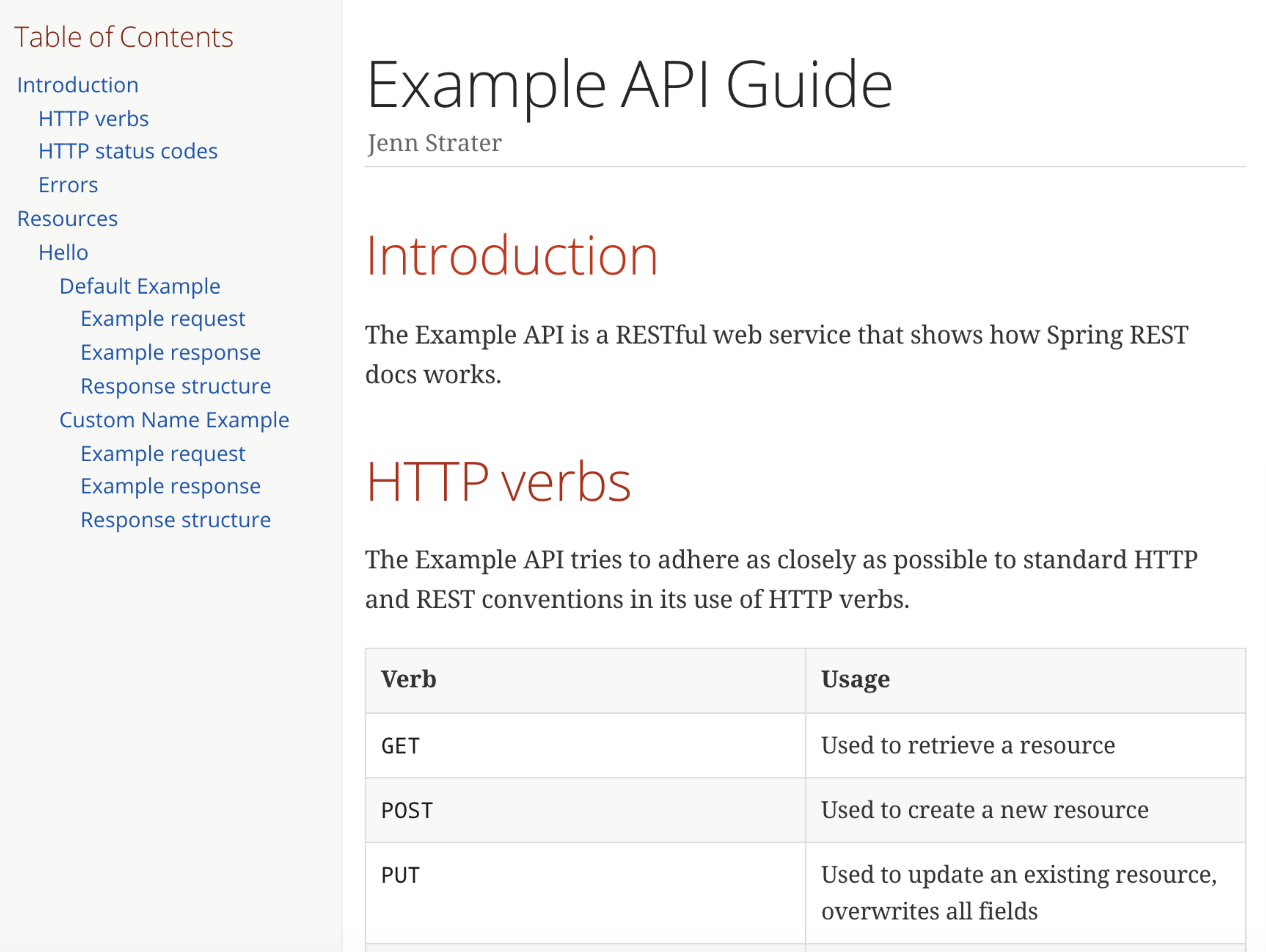
index.adoc

src/docs

index.html

src/main/resources/public/html5

Build Docs



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

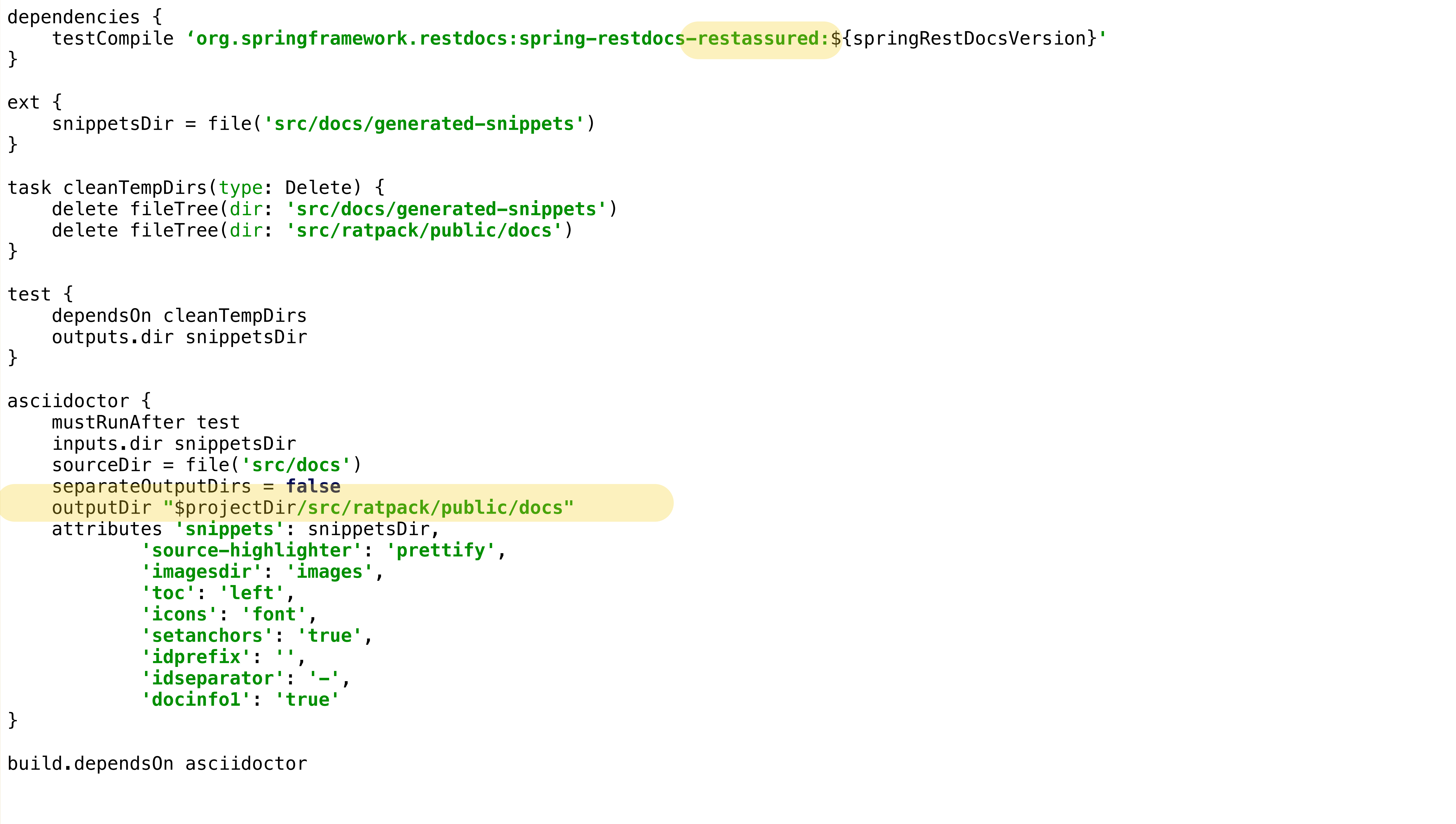
Groovier Spring REST docs

- Ratpack
- Grails

Groovier Spring REST docs Example - Ratpack

- 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:${springRestDocsVersion}'
}

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()
   .
   .
   .
}


//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
         }
    }
}
```

@codeJENNerator

```
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

@codeJENNerator

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

    byMethod {
        get {
            bookService.find(isbn).
                single()
                                       ->
```

Get a book by ISBN
/api/books/1932394842

```
                    } else {
                        render book
                    }
                }
        }
        ...
    }
}
```

```
byMethod {
   get {
      bookSe
         toL
                                          ->
```

Get all books
/api/books

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

Post to create a new book
/api/books

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

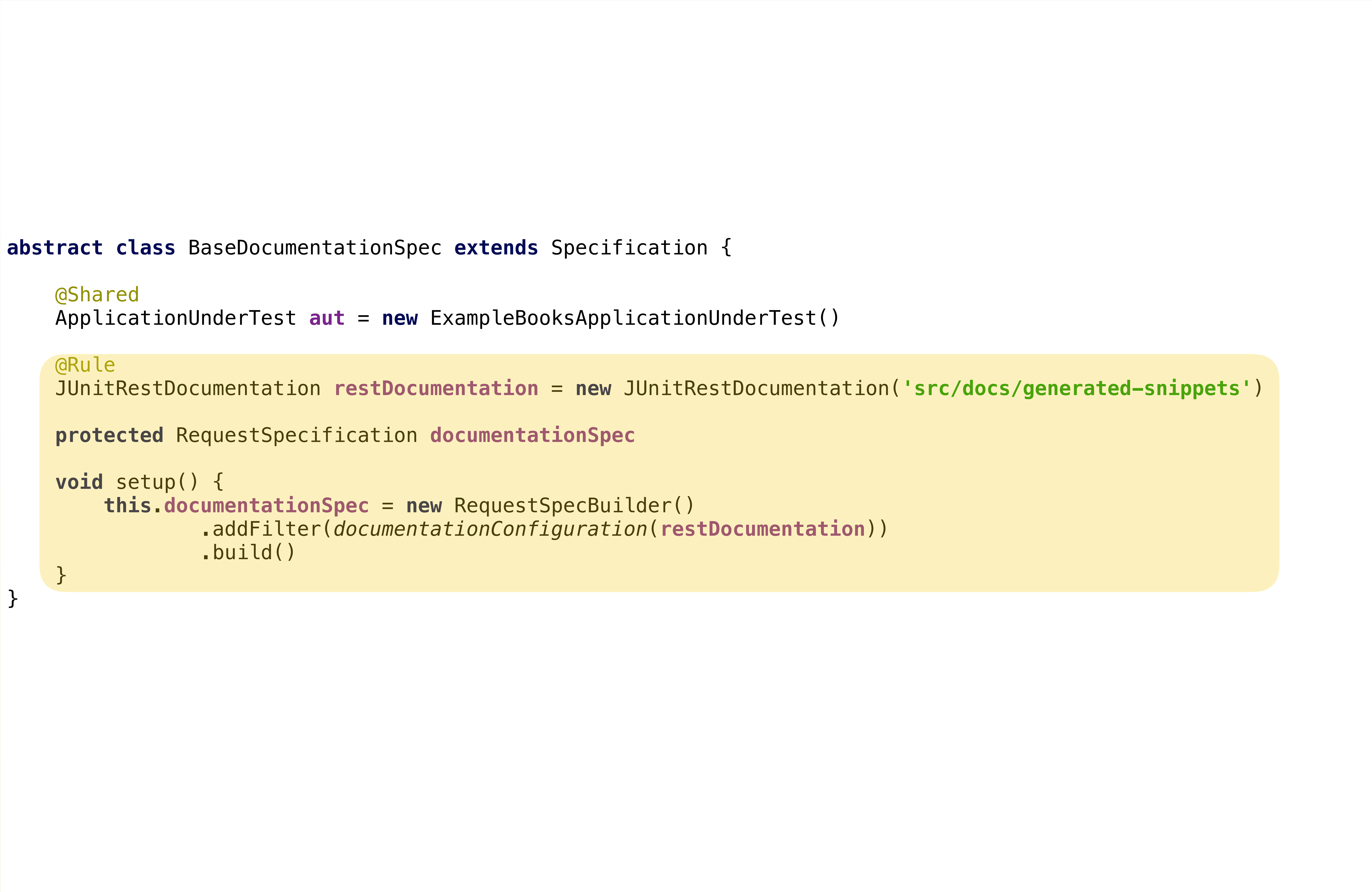
@codeJENNerator

Spring REST docs

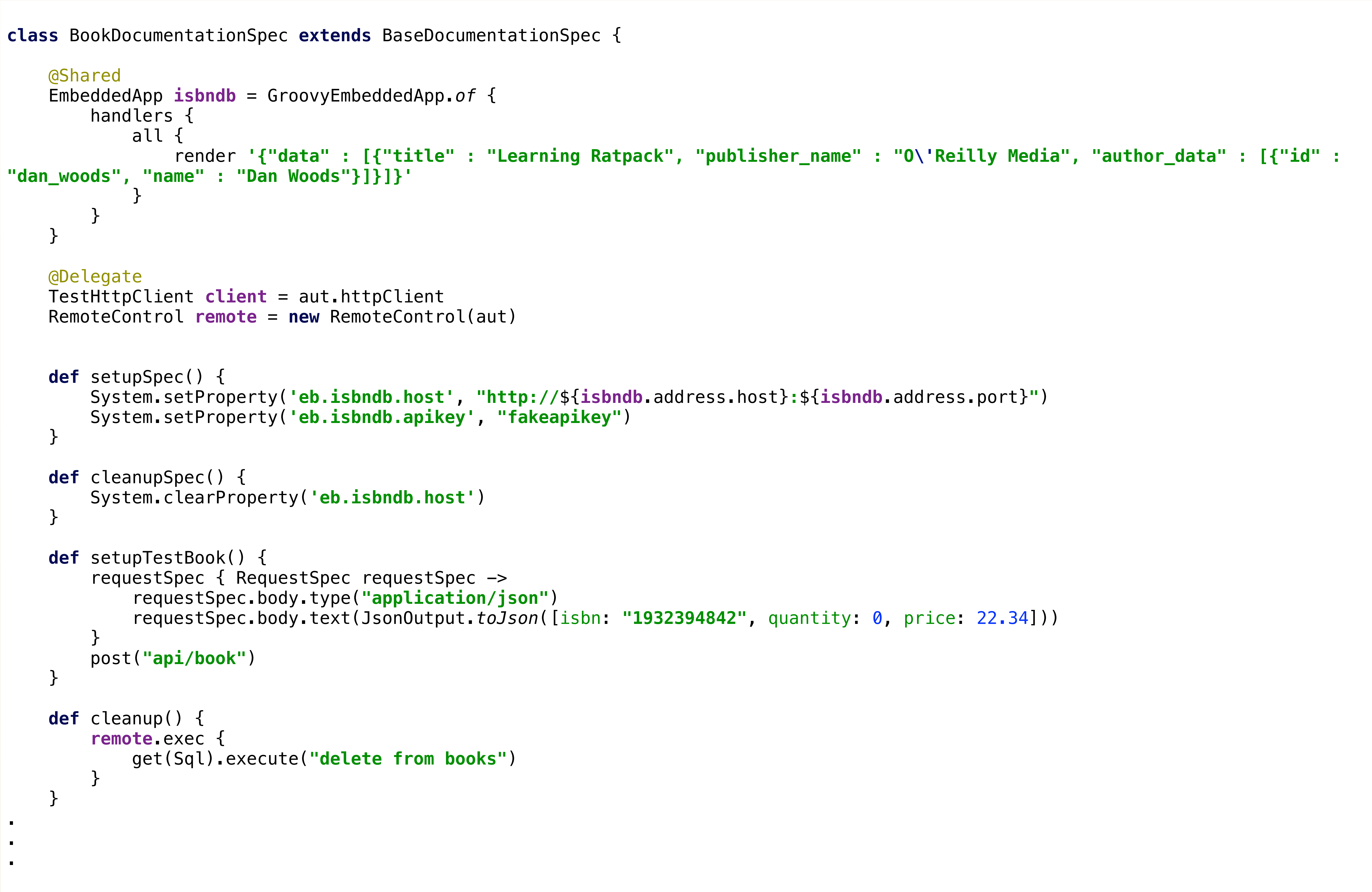
&

REST-assured

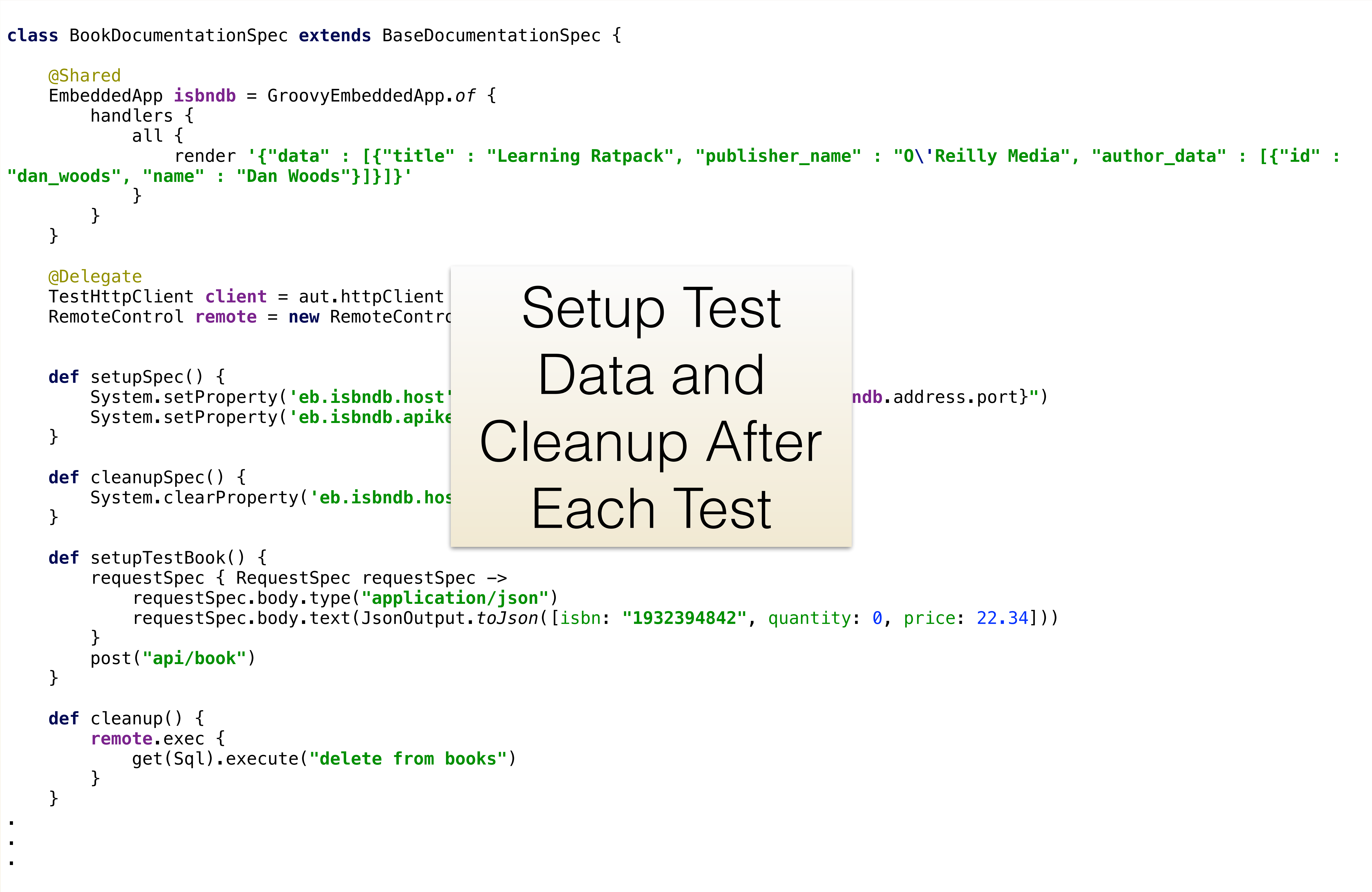
https://github.com/jayway/rest-assured



```
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()
    }
}
```

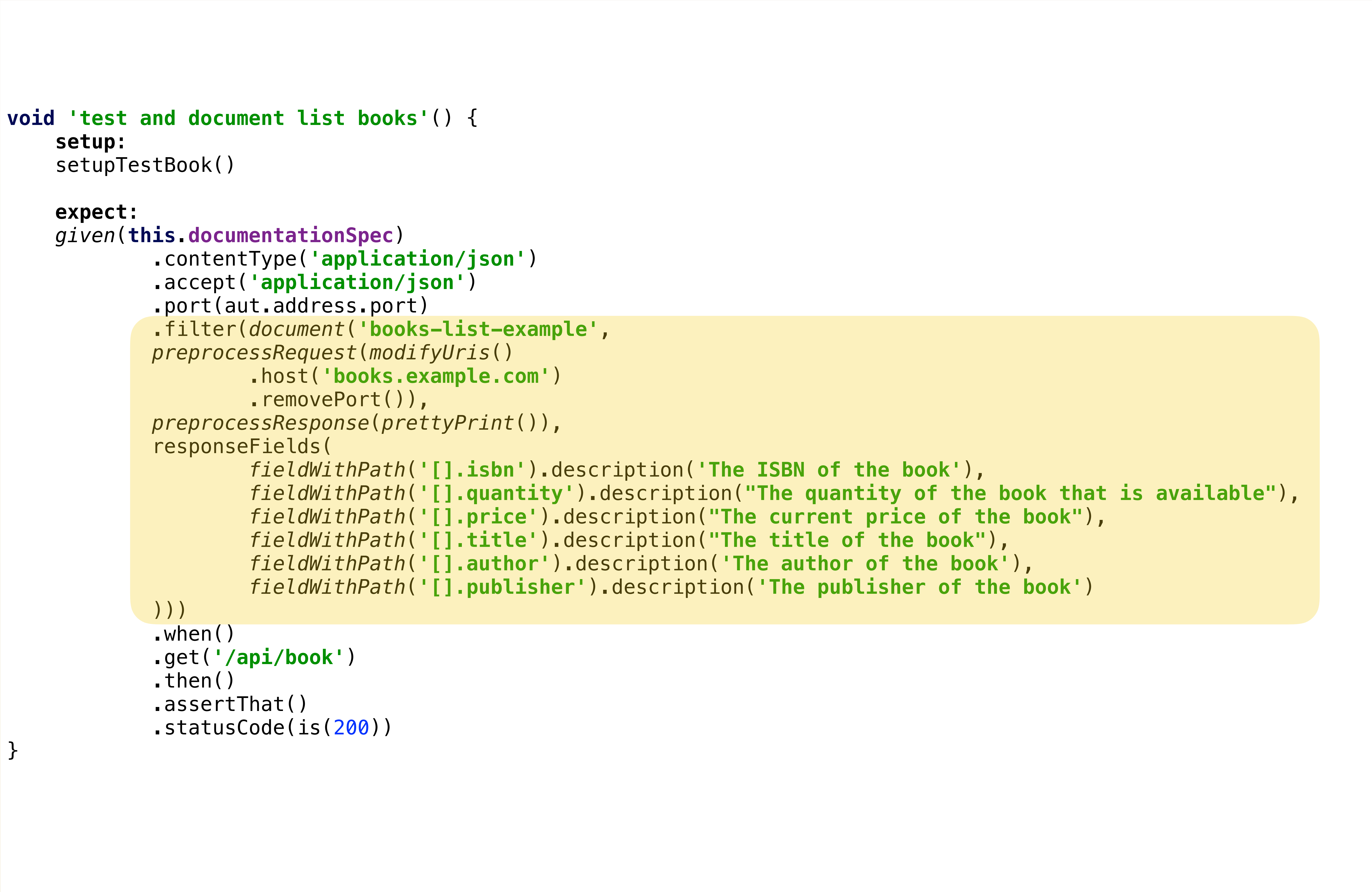


```
class BookDocumentationSpec extends BaseDocumentationSpec {

    @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")
        }
    }
.
.
.
```



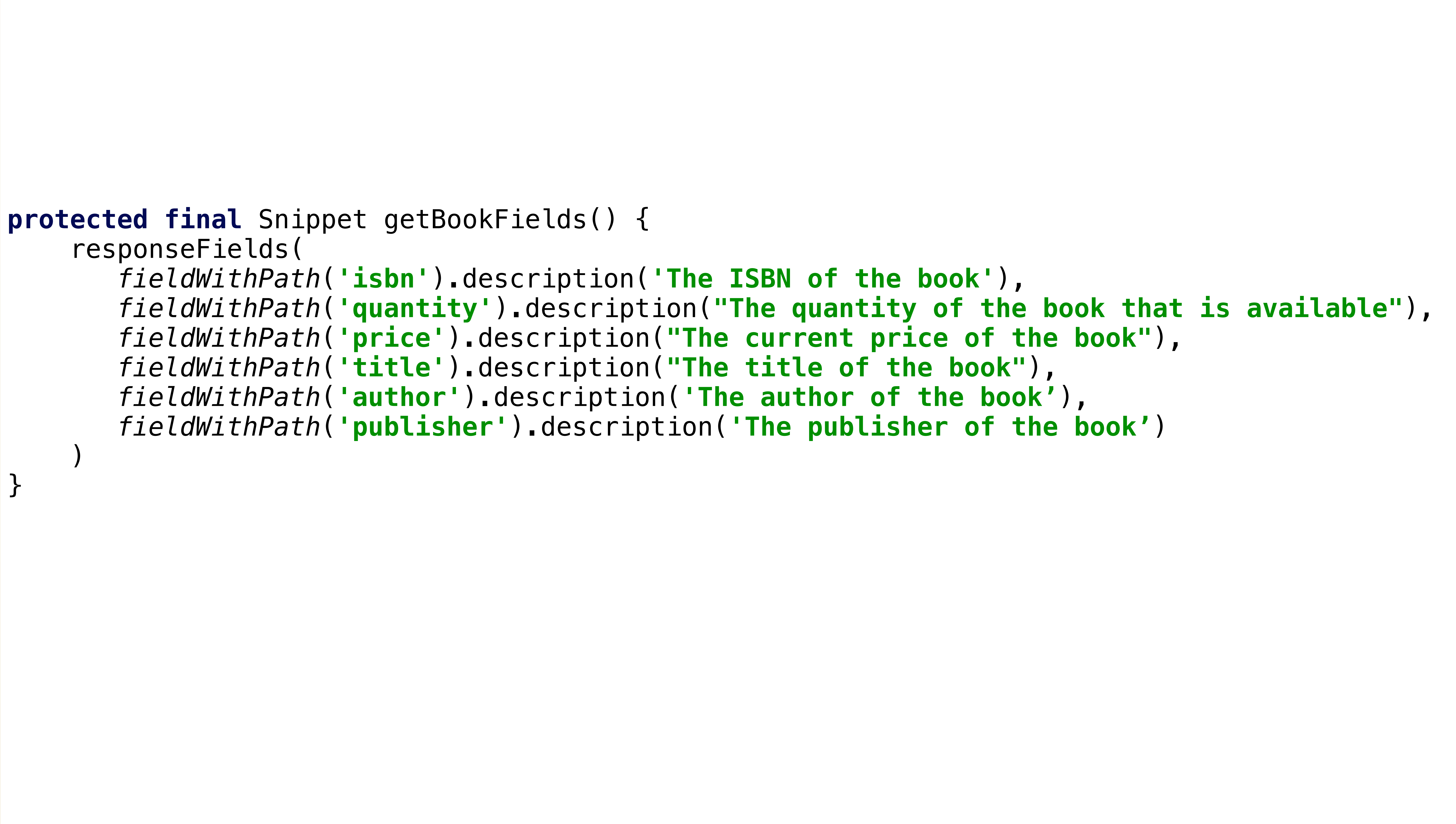
```
class BookDocumentationSpec extends BaseDocumentationSpec {

    @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 RemoteContro


    def setupSpec() {
        System.setProperty('eb.isbndb.host'                    ndb.address.port}")
        System.setProperty('eb.isbndb.apike
    }

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

    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")
        }
    }
.
.
.
```

Setup Test Data and Cleanup After Each Test

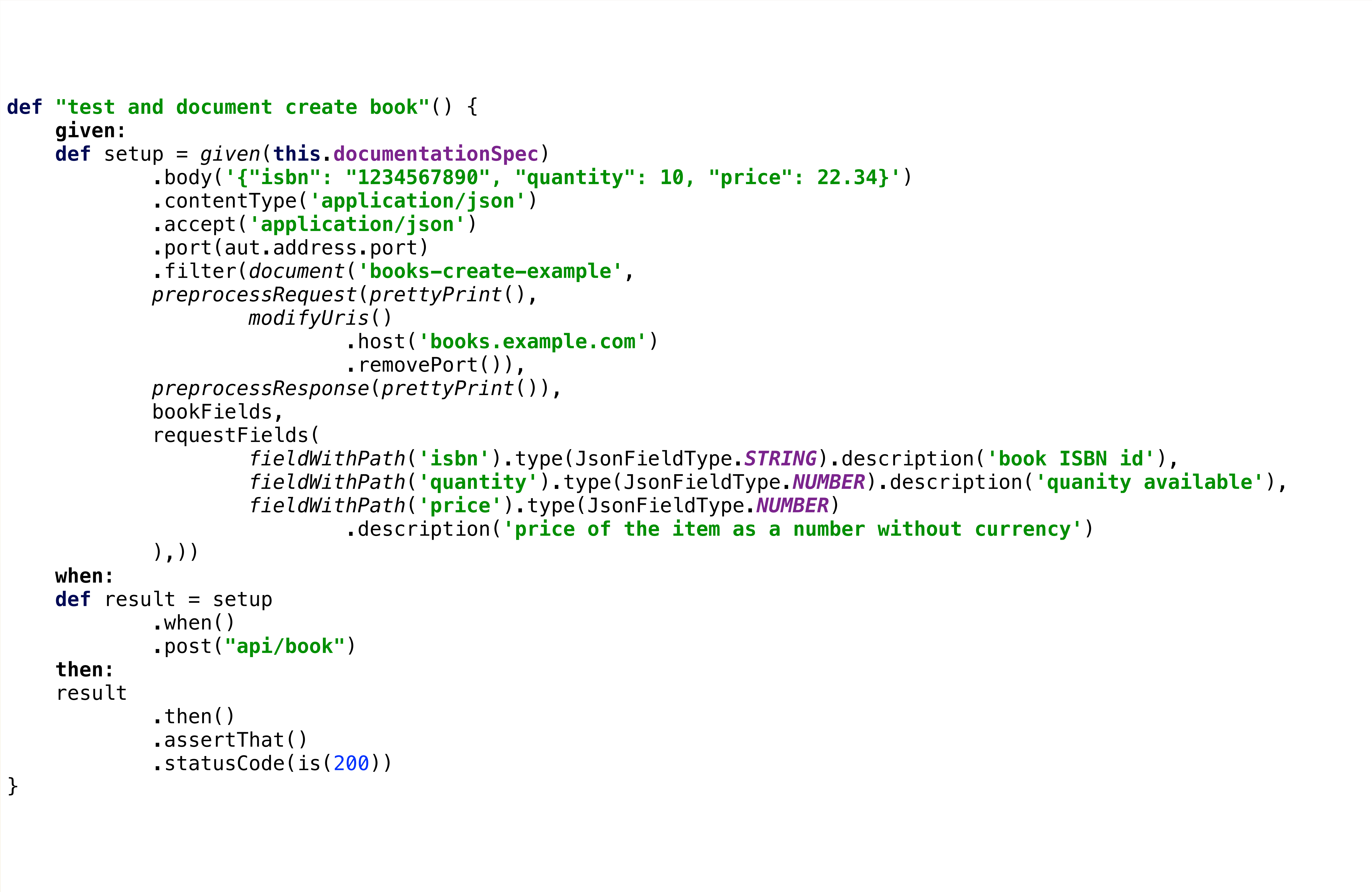


```
void 'test and document list books'() {
    setup:
    setupTestBook()

    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))
}
```

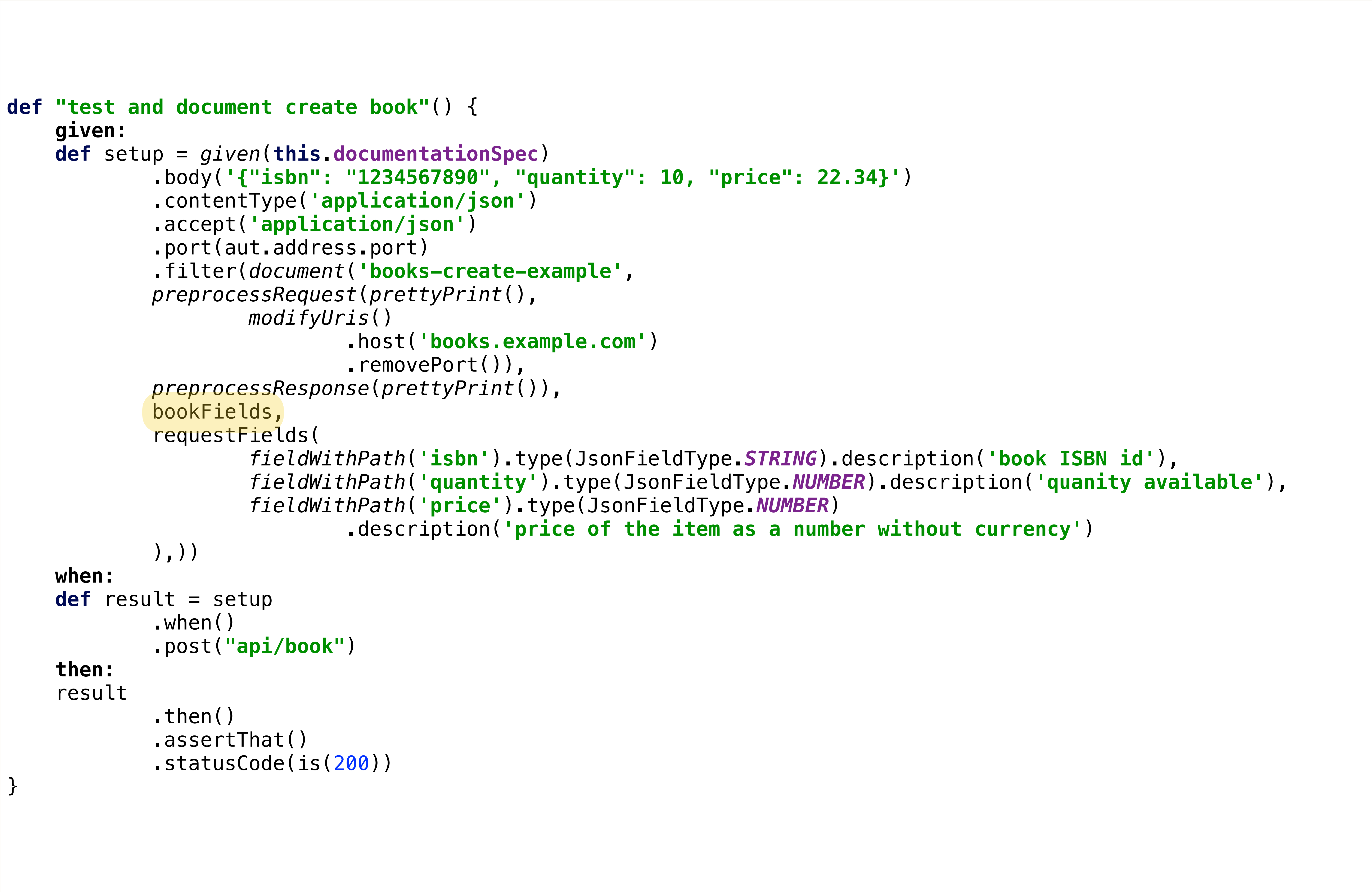
Reusable Snippets



```
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’)
    )
}
```

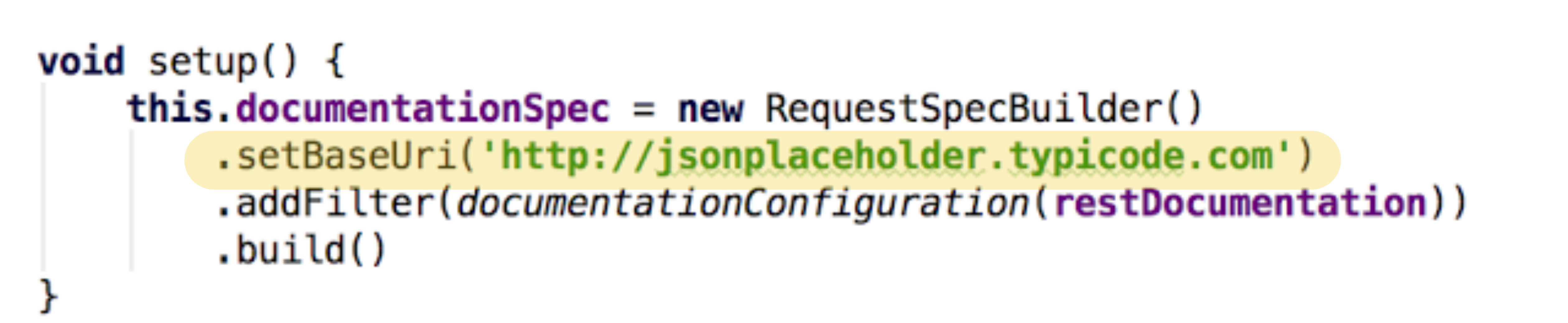


```
def "test and document create book"() {
    given:
    def setup = given(this.documentationSpec)
            .body('{"isbn": "1234567890", "quantity": 10, "price": 22.34}')
            .contentType('application/json')
            .accept('application/json')
            .port(aut.address.port)
            .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))
}
```

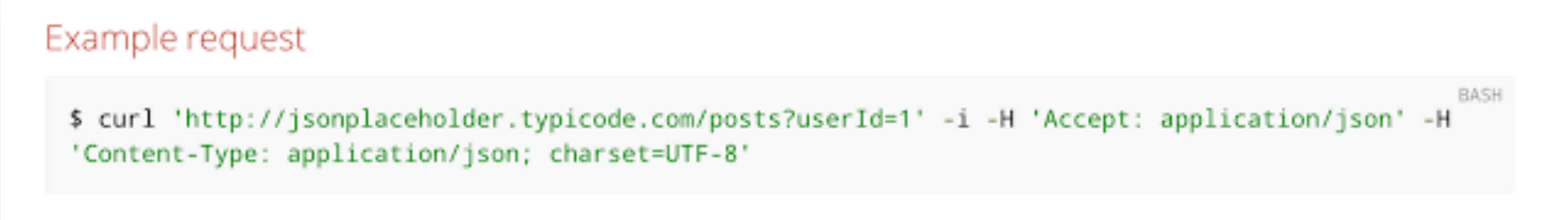


```
def "test and document create book"() {
    given:
    def setup = given(this.documentationSpec)
            .body('{"isbn": "1234567890", "quantity": 10, "price": 22.34}')
            .contentType('application/json')
            .accept('application/json')
            .port(aut.address.port)
            .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))
}
```

Public APIs



```
void setup() {
    this.documentationSpec = new RequestSpecBuilder()
        .setBaseUri('http://jsonplaceholder.typicode.com')
        .addFilter(documentationConfiguration(restDocumentation))
        .build()
}
```



Example request

```
$ curl 'http://jsonplaceholder.typicode.com/posts?userId=1' -i -H 'Accept: application/json' -H
'Content-Type: application/json; charset=UTF-8'
```

https://jennstrater.blogspot.dk/2017/01/using-spring-rest-docs-to-document.html

Groovier Spring REST docs Example - Grails

Groovier Spring REST docs Example - Grails

- Grails 3.0.15 with Web API Profile

Groovier Spring REST docs Example - Grails

- Grails 3.0.15 with Web API Profile

+ Asciidoctor Gradle plugin

Groovier Spring REST docs Example - Grails

- Grails 3.0.15 with Web API Profile

+ Asciidoctor Gradle plugin

Groovier Spring REST docs Example - Grails

- Grails 3.0.15 with Web API Profile

+ Asciidoctor Gradle plugin

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

Groovier Spring REST docs Example - Grails

- Grails 3.0.15 with Web API Profile

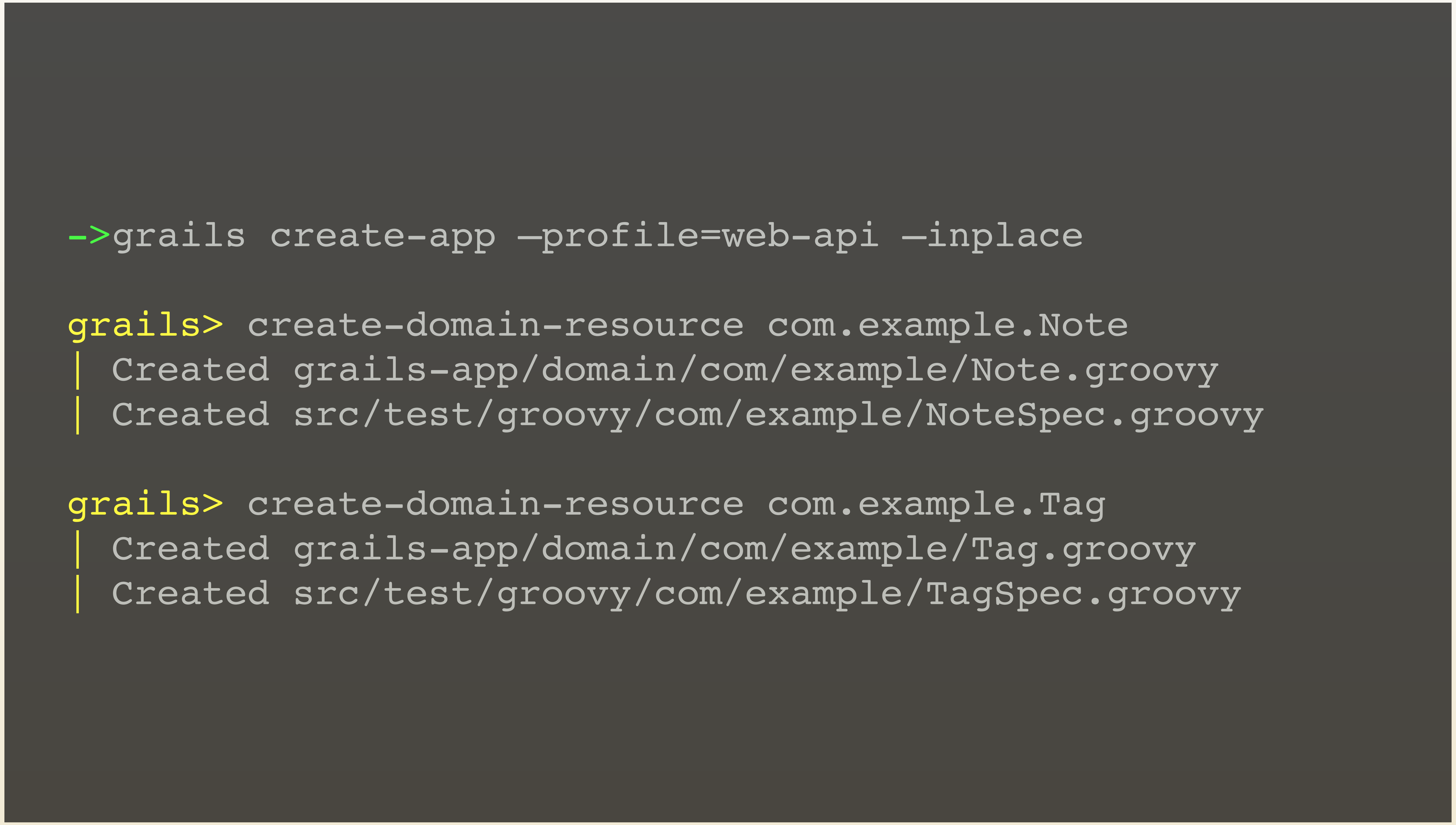
+ Asciidoctor Gradle plugin

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

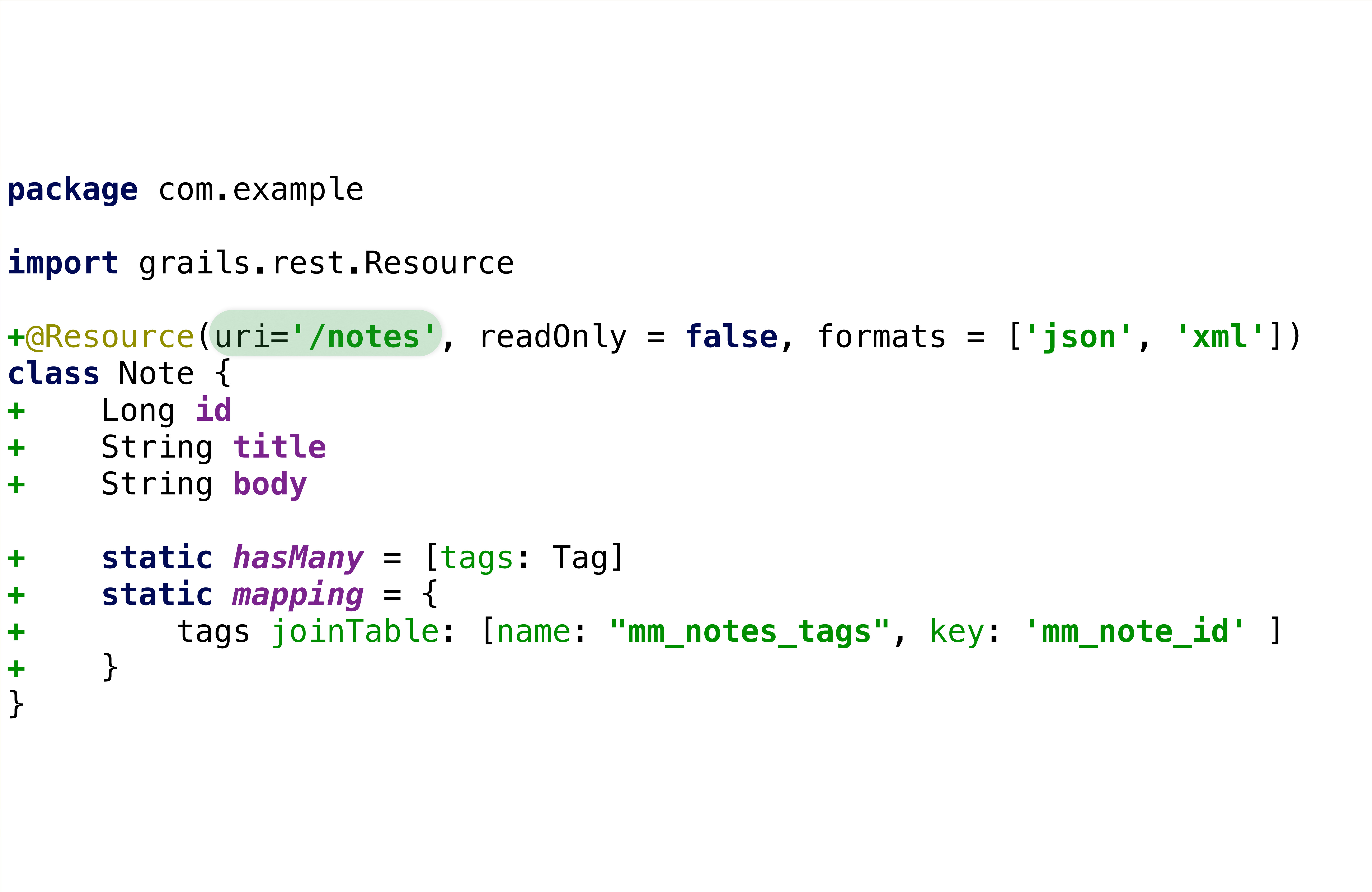
+ Publish to Github Pages

Simple Grails App

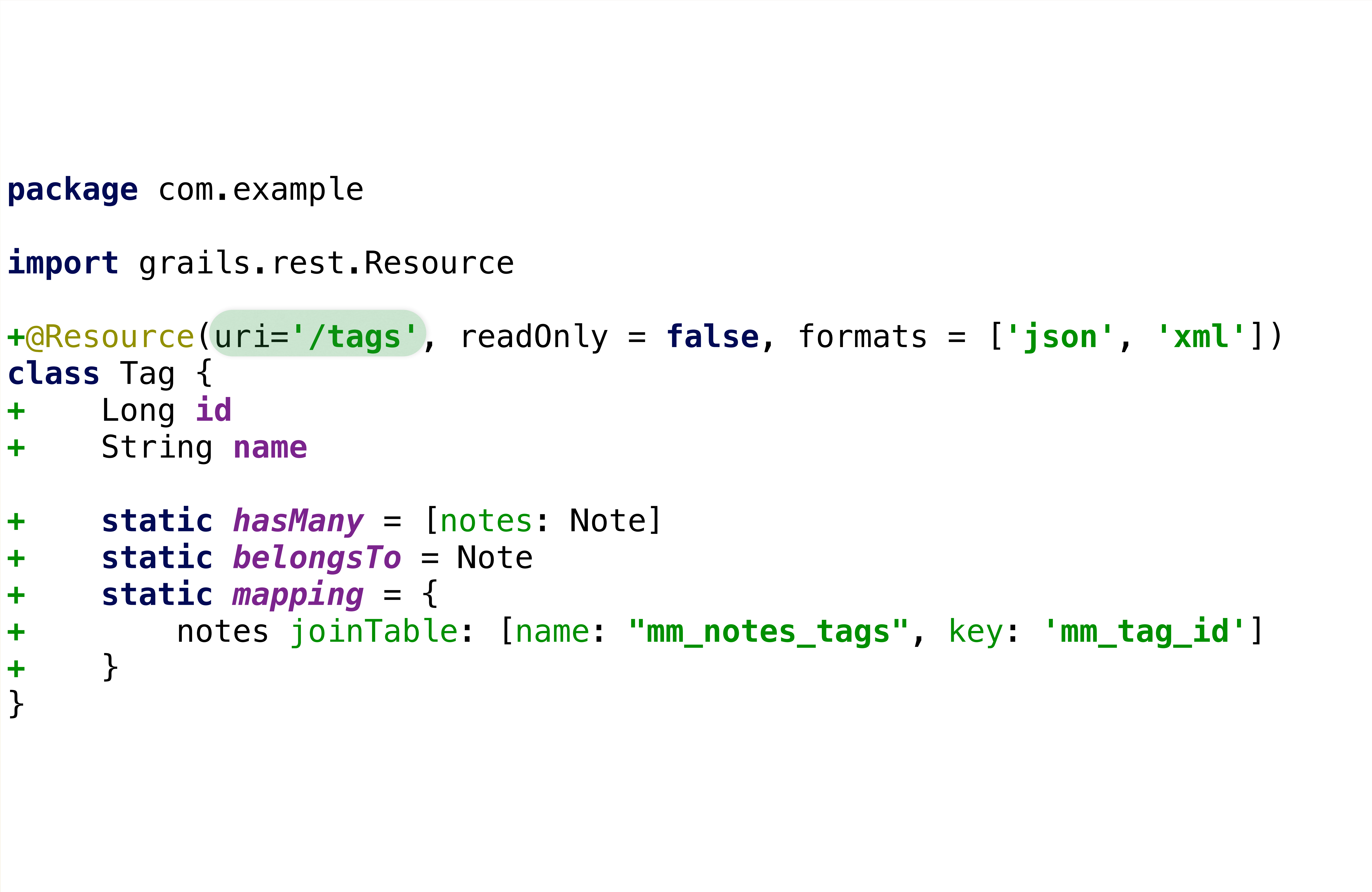
Simple Grails App



```
->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

@codeJENNerator

Spring REST docs

&

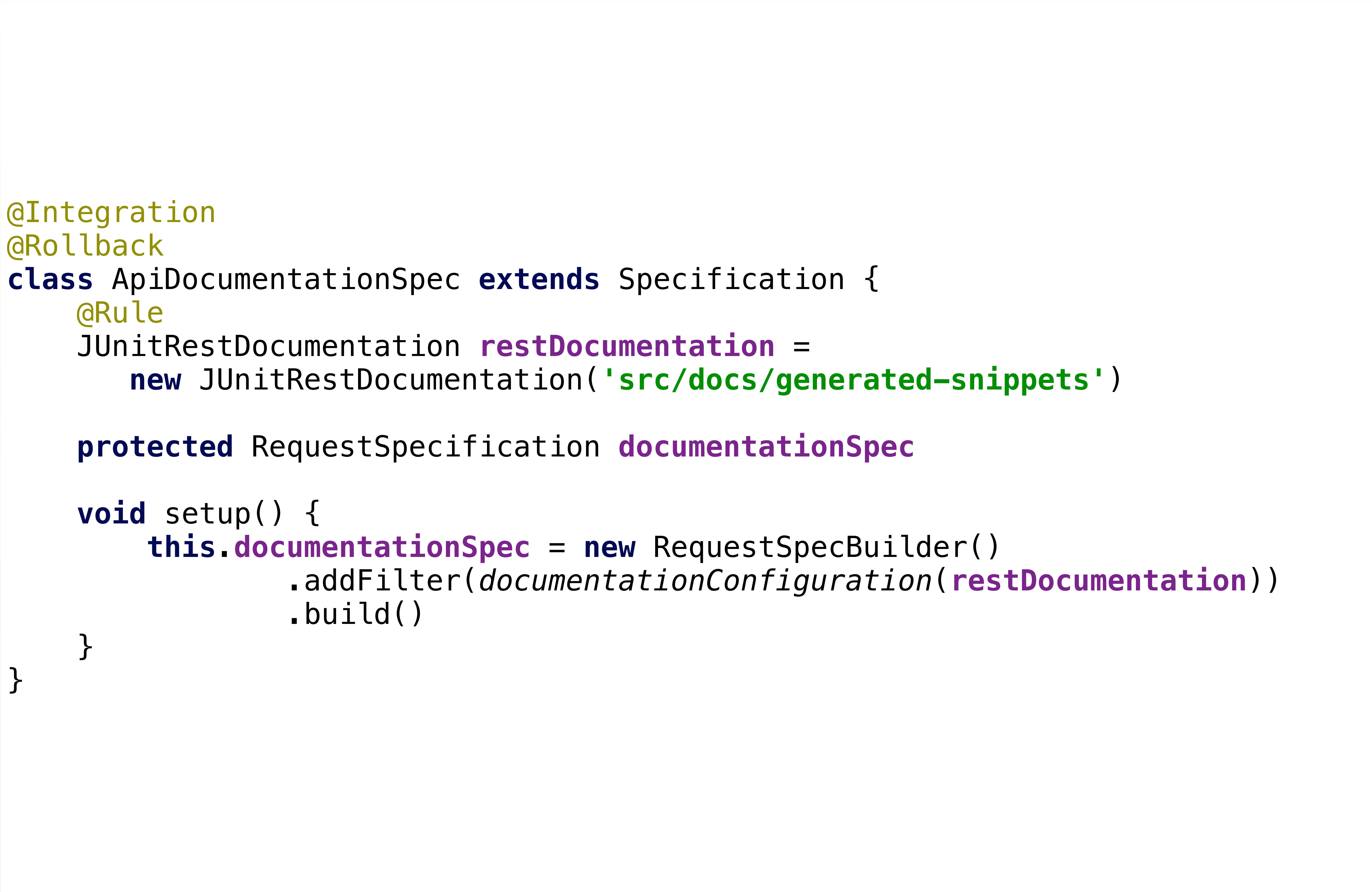
REST-assured

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

@codeJENNerator

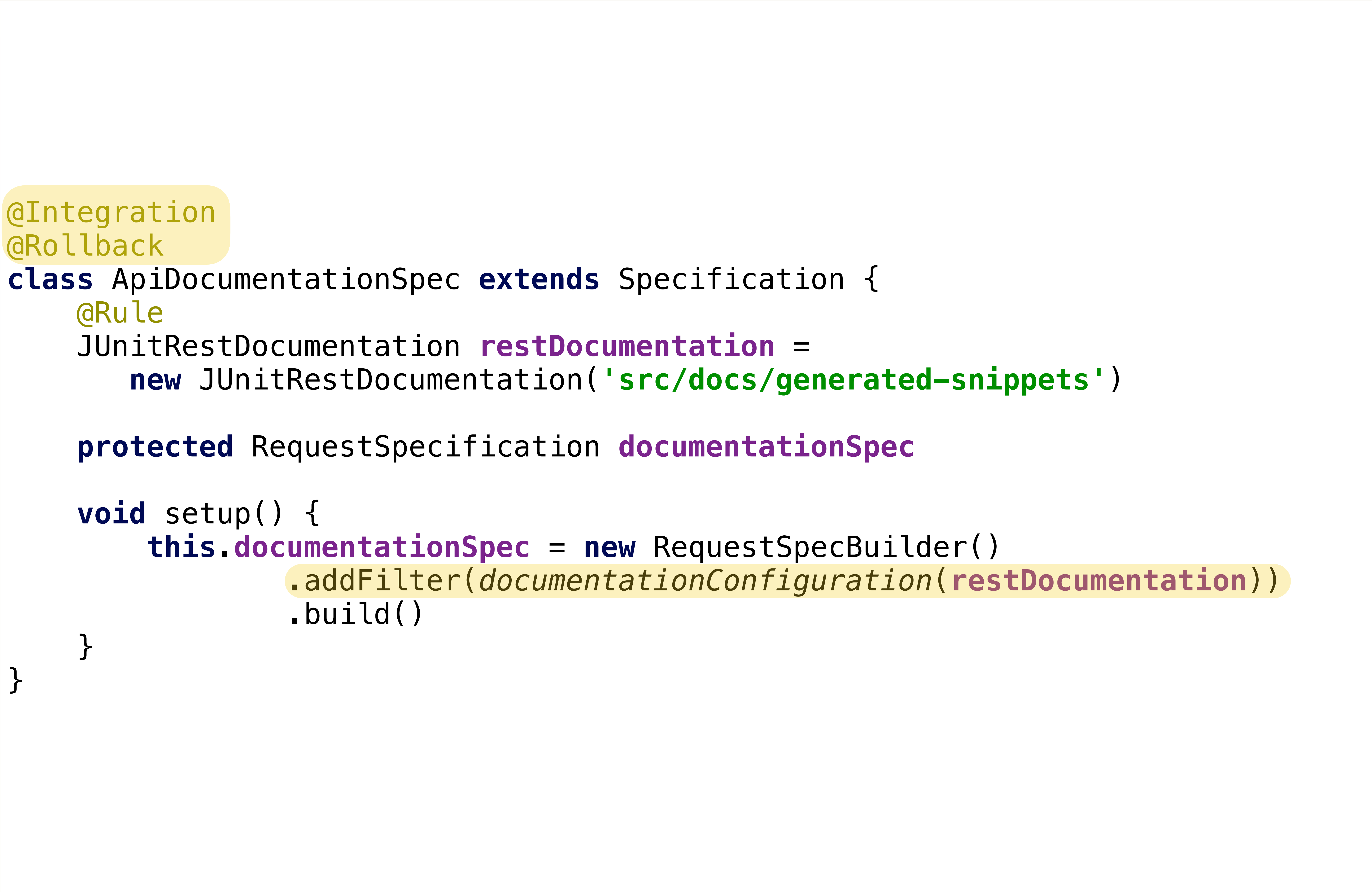
Grails Specific Setup

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

asciidoctor {
  …
+ mustRunAfter integrationTest
  …
}
```

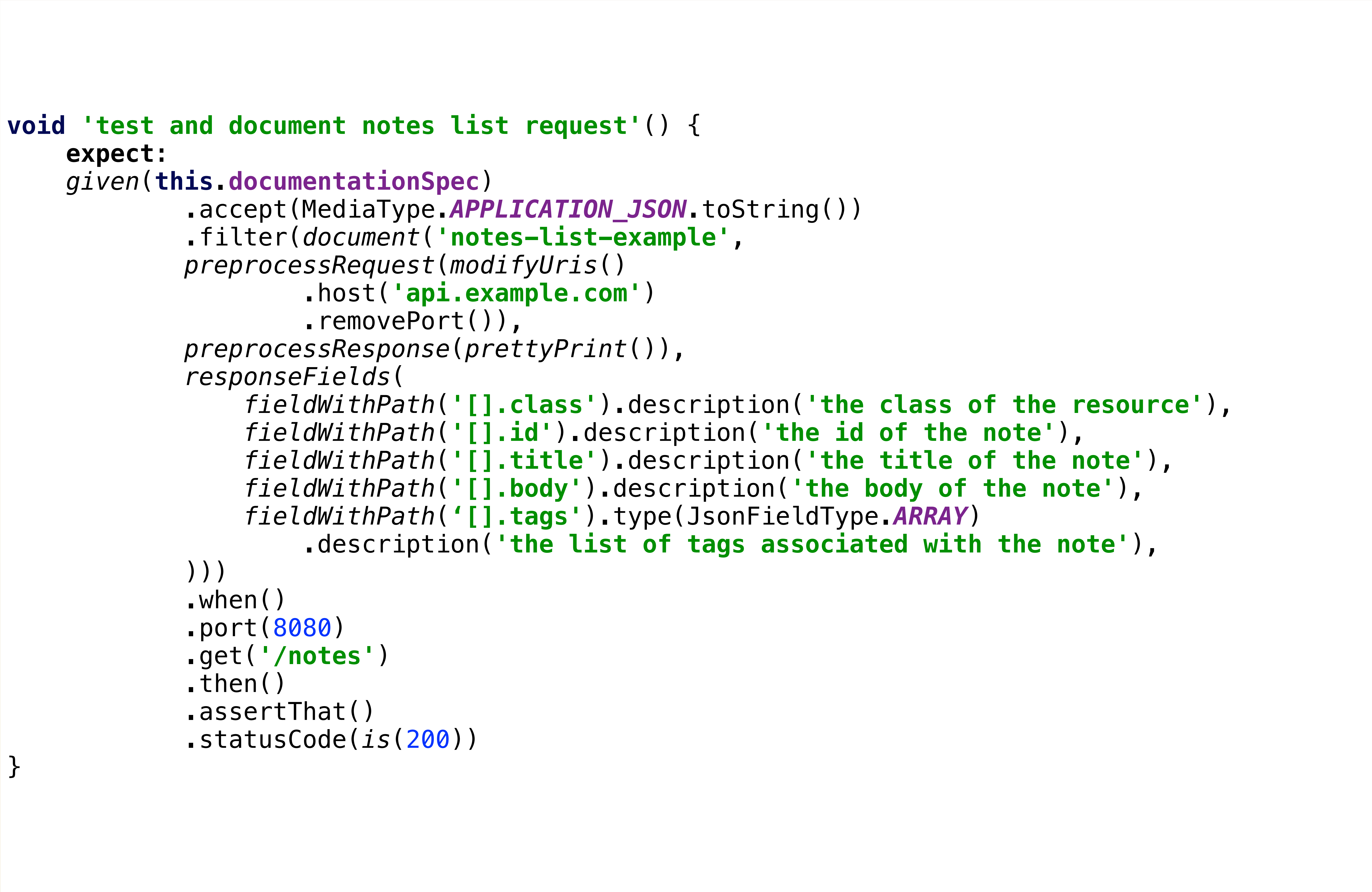


```
@Integration
@Rollback
class ApiDocumentationSpec extends Specification {
    @Rule
    JUnitRestDocumentation restDocumentation =
       new JUnitRestDocumentation('src/docs/generated-snippets')

    protected RequestSpecification documentationSpec

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

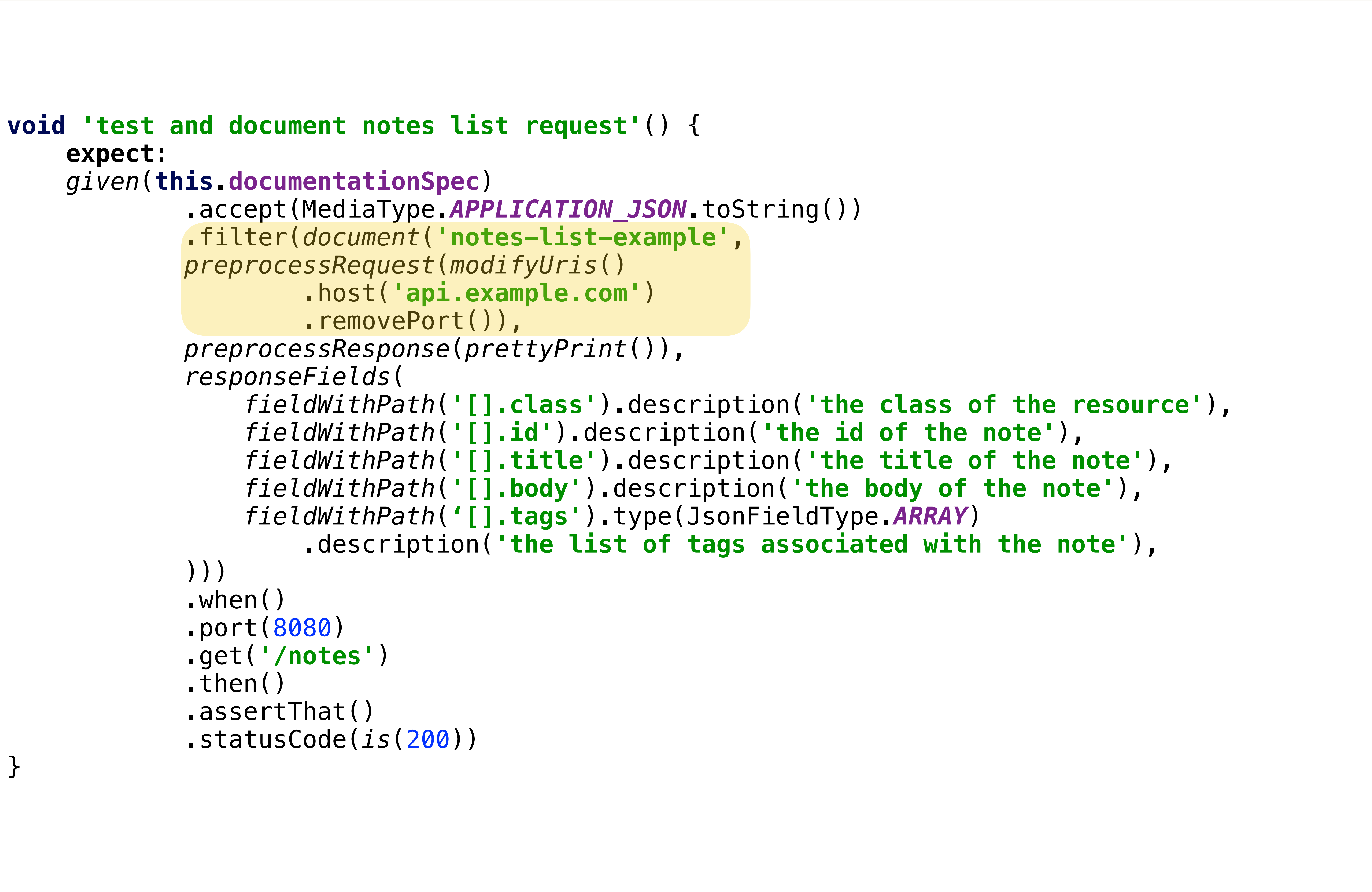
```
@Integration
@Rollback
class ApiDocumentationSpec extends Specification {
    @Rule
    JUnitRestDocumentation restDocumentation =
       new JUnitRestDocumentation('src/docs/generated-snippets')

    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()
    }
}
```

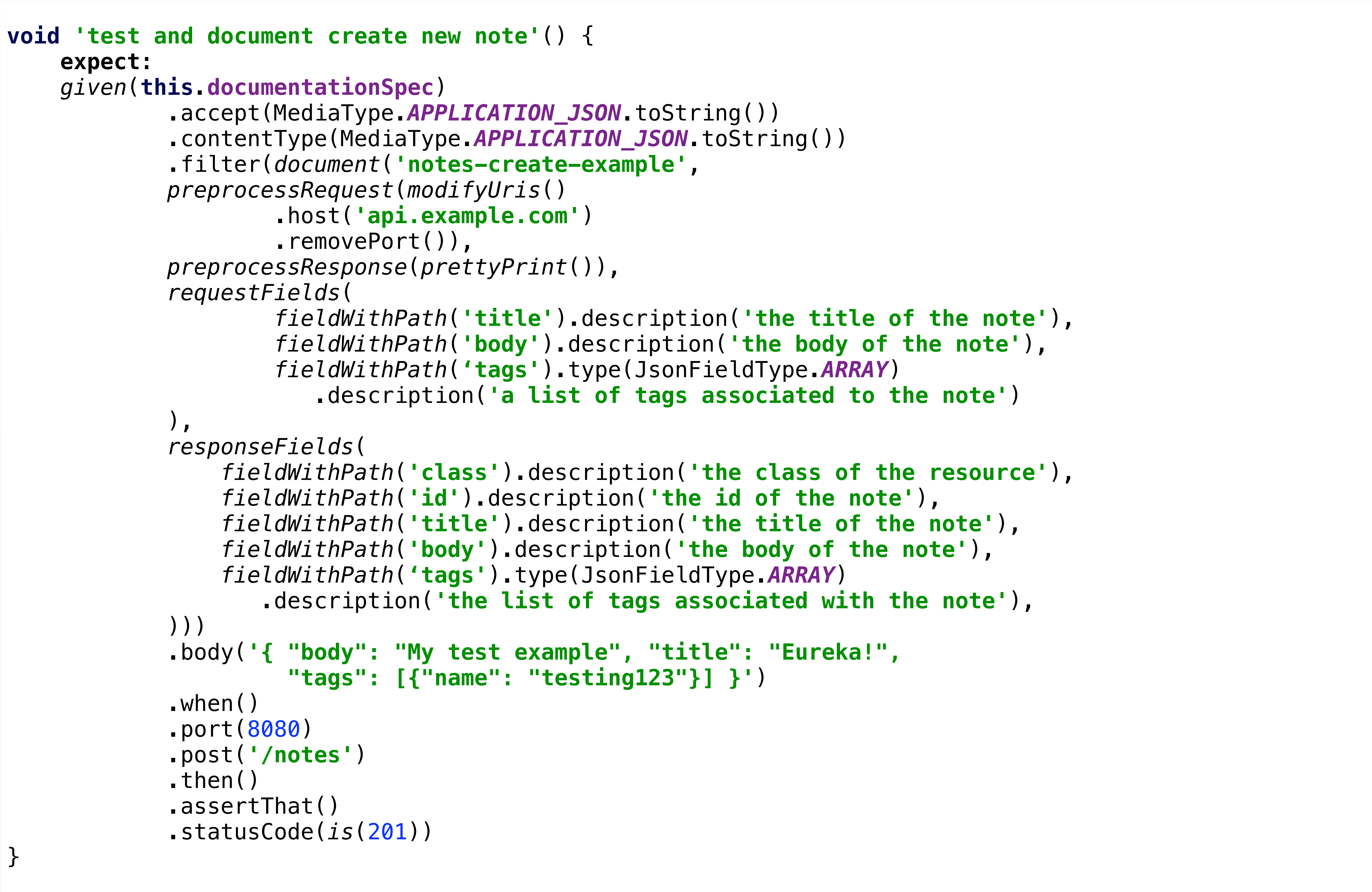


```
void 'test and document notes list request'() {
    expect:
    given(this.documentationSpec)
            .accept(MediaType.APPLICATION_JSON.toString())
            .filter(document('notes-list-example',
            preprocessRequest(modifyUris()
                    .host('api.example.com')
                    .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))
}
```



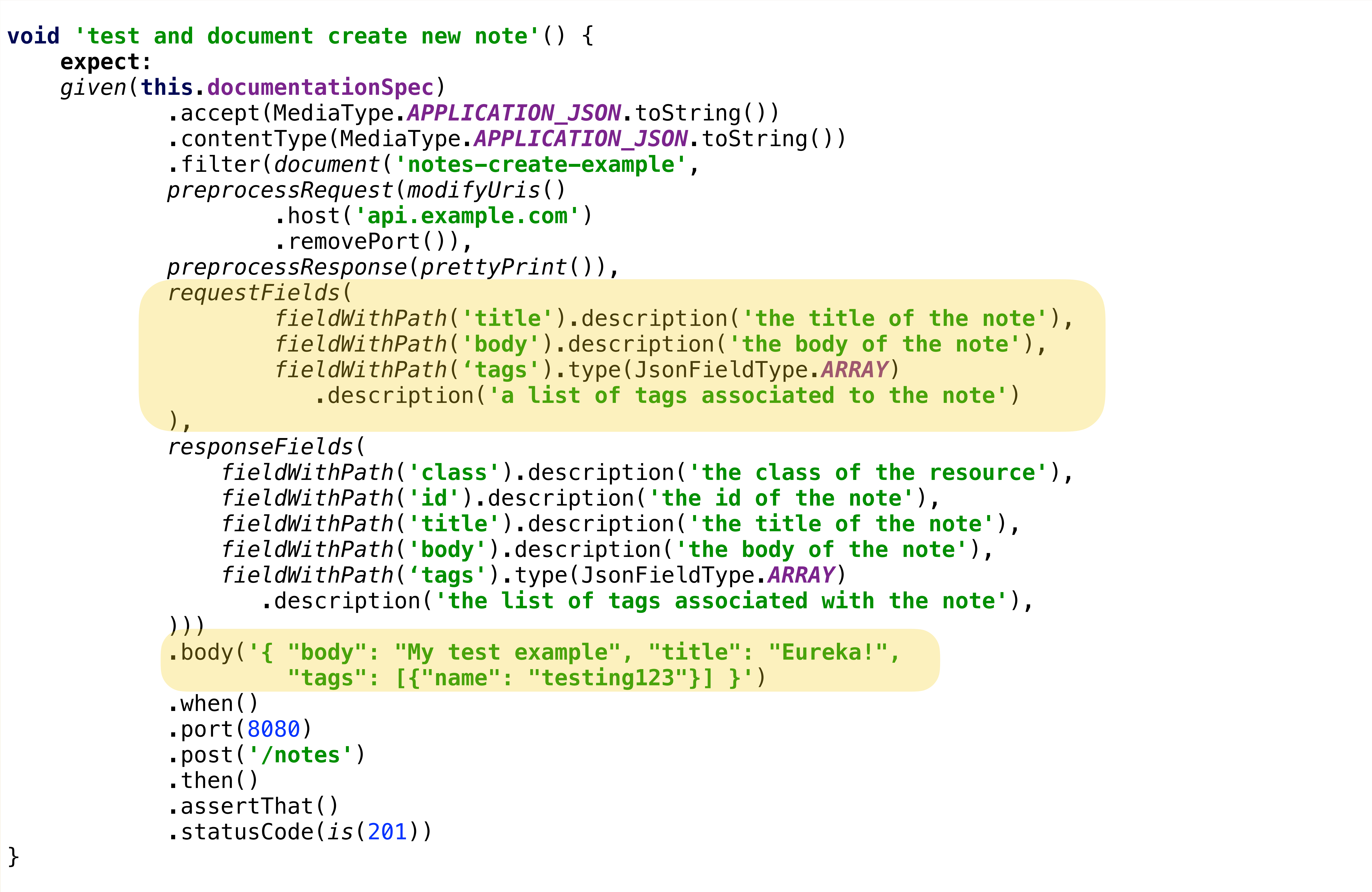
```
void 'test and document notes list request'() {
    expect:
    given(this.documentationSpec)
            .accept(MediaType.APPLICATION_JSON.toString())
            .filter(document('notes-list-example',
            preprocessRequest(modifyUris()
                    .host('api.example.com')
                    .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))
}
```

@codeJENNerator



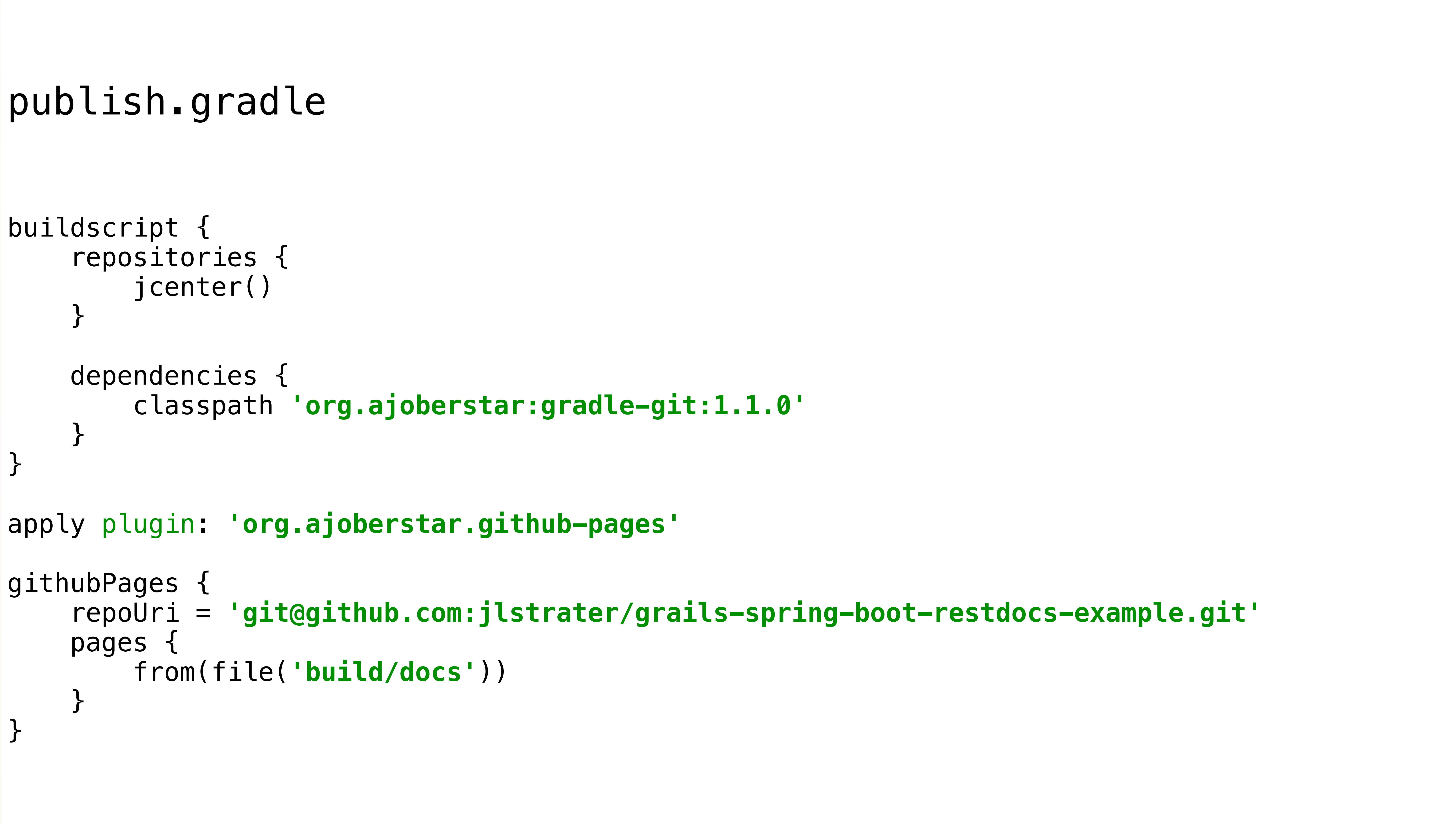
```
void 'test and document create new note'() {
    expect:
    given(this.documentationSpec)
            .accept(MediaType.APPLICATION_JSON.toString())
            .contentType(MediaType.APPLICATION_JSON.toString())
            .filter(document('notes-create-example',
            preprocessRequest(modifyUris()
                    .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))
}
```

@codeJENNerator



```
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))
}
```

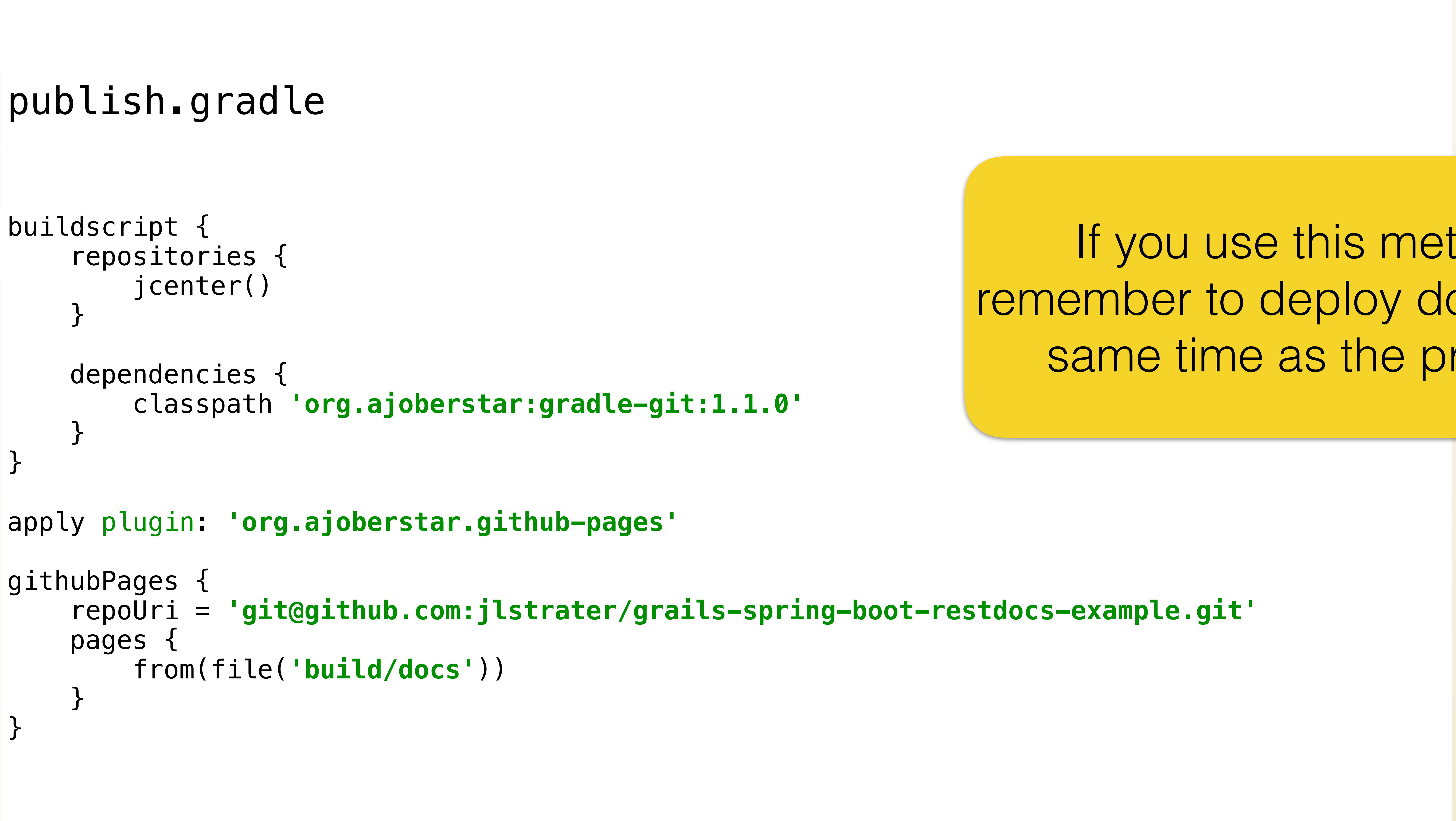
Publish Docs to Github Pages



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'))
    }
}
```

Publish Docs to Github Pages



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'))
    }
}
```

If you use this method, remember to deploy docs at the same time as the project!

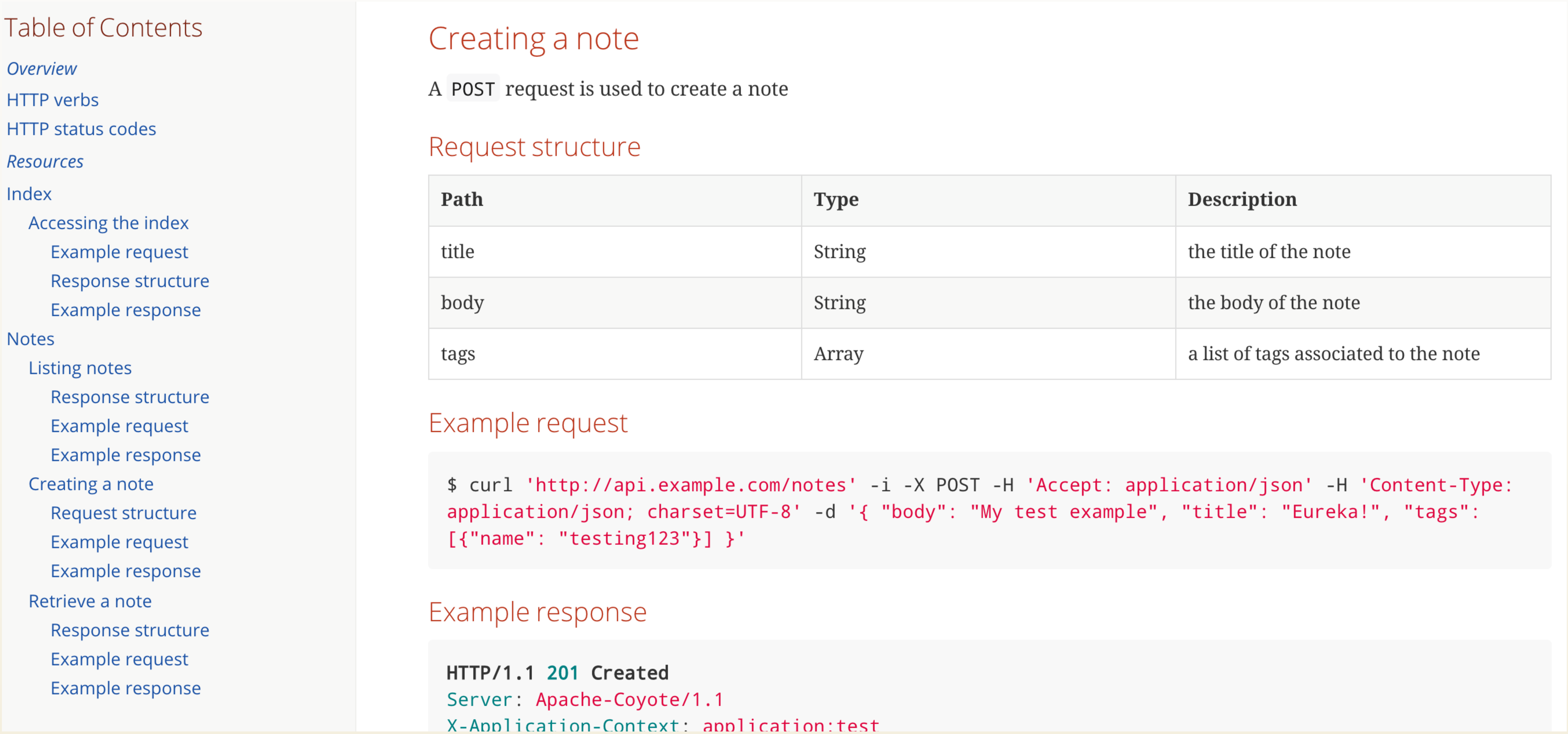


Table of Contents

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

@codeJENNerator

Outcomes

One Year Later

- Made it to production! :)
- Team still happy with Spring REST Docs
- Other dev teams like to see the examples

Read the docs for more on..

- Adding Security and Headers
- Documenting Constraints
- Hypermedia Support
- XML Support
- Using Markdown instead of Asciidoc
- Third Party Extensions for WireMock and Jersey

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

Next Steps

- Join the Groovy Community on Slack
  groovycommunity.com

- Join #spring-restdocs on gitter
  https://gitter.im/spring-projects/spring-restdocs