


[ANDROID ▾](#)
[JAVA ▾](#)
[JVM LANGUAGES ▾](#)
[SOFT DEV](#)
[WEB DEVELOPMENT ▾](#)
[AGILE](#)
[CAREER](#)
[COMMS](#)
[DEVOPS](#)
[META JCG ▾](#)
[Home](#) » [Java](#) » [Enterprise Java](#) » Integrating Swagger with Spring Boot REST API

ABOUT CHANDANA NAPAGODA



Integrating Swagger with Spring Boot REST API

Posted by: Chandana Napagoda in Enterprise Java September 12th, 2017 6 Comments 4583 Views

In the last post, I talked about my experience with creating RESTful Services using Spring Boot. When creating a REST API, proper documentation is a mandatory part of it.

What is Swagger?

Swagger (Swagger 2) is a specification for describing and documenting a REST API. It specifies the format of the REST web services including URL, Resources, methods, etc. Swagger will generate documentation from the application code and handle the rendering part as well.

In this post, I am going to integrate Swagger 2 documentation into a Spring Boot based REST web service. So I am going to use Springfox implementation to generate the swagger documentation. If you want to know how to run/build Spring Boot project, please refer my previous post.

Springfox provides two dependencies to generate API Doc and Swagger UI. If you are not expecting to integrate Swagger UI into your API level, no need to add Swagger UI dependency.

```

1 <dependency>
2
3 <groupId>io.springfox</groupId>
4
5 <artifactId>springfox-swagger2</artifactId>
6
7 <version>2.7.0</version>
8
9 </dependency>

1 <dependency>
2
3 <groupId>io.springfox</groupId>
4
5 <artifactId>springfox-swagger-ui</artifactId>
6
7 <version>2.7.0</version>
8
9 </dependency>

```

@EnableSwagger2 annotation enables Springfox Swagger support in the class. To document the service, Springfox uses a Docket. The Docket helps to configure a subset of the services to be documented and group them by a name, etc. The most hidden concept is that the Springfox works by examining an application at runtime using API semantics based on spring configurations. In other words, you have to create a Spring Java Configuration class which uses spring's @Configuration

In My example, I am generating a swagger documentation based on the RestController classes I have added.

```

01 import springfox.documentation.spring.web.plugins.Docket;
02 import springfox.documentation.swagger2.annotations.EnableSwagger2;
03
04 @Configuration
05 @EnableSwagger2
06 public class ApplicationConfig {
07
08     @Bean
09     public Docket api() {
10         return new Docket(DocumentationType.SWAGGER_2)
11             .select()
12             .apis(RequestHandlerSelectors.basePackage("com.chandana.helloworld.controllers"))
13             .paths(PathSelectors.any())
14             .build();
15     }
16 }

```



Since I have added two controllers, this will group(tag) each controller related APIs separately.

The screenshot shows the Swagger UI interface. At the top, there's a green header with the Swagger logo, a dropdown menu set to 'default (/v2/api-docs)', and an 'Explore' button. Below the header, the title 'Api Documentation' is displayed, followed by 'Api Documentation' and 'Apache 2.0'. The main content area lists two controllers: 'admin-controller : Admin Controller' and 'hello-world-controller : Hello World Controller'. Each controller has a table of API endpoints. For 'admin-controller', there are two GET endpoints: '/admin/' with response 'welcome' and '/admin/hello/{name}' with response 'message'. For 'hello-world-controller', there are two GET endpoints: '/api/' with response 'welcome' and '/api/hello/{name}' with response 'message'. At the bottom, it says '[BASE URL: / , API VERSION: 1.0]'.

Out of the box, Springfox provides five predicates and they are any, none, withClassAnnotation, withMethodAnnotation and basePackage.

ApiInfo

Swagger provides some default values such as "API Documentation", "Created by Contact Email", "Apache 2.0". So you can change these default values by adding apiInfo(ApiInfo apiInfo) method. The ApiInfo class contains custom information about the API.

```

01 | @Bean
02 | public Docket api() {
03 |     return new Docket(DocumentationType.SWAGGER_2)
04 |         .apiInfo(getApiInfo())
05 |         .select()
06 |         .apis(RequestHandlerSelectors.basePackage("com.chandana.helloworld.controllers"))
07 |         .paths(PathSelectors.any())
08 |         .build();
09 | }
10 |
11 | private ApiInfo getApiInfo() {
12 |     Contact contact = new Contact("Chandana Napagoda", "http://blog.napagoda.com", "cnapagoda@gmail.com");
13 |     return new ApiInfoBuilder()
14 |         .title("Example Api Title")
15 |         .description("Example Api Definition")
16 |         .version("1.0.0")
17 |         .license("Apache 2.0")
18 |         .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0")
19 |         .contact(contact)
20 |         .build();
21 | }

```

Once ApiInfo is added, the generated documentation looks similar to this:

The screenshot shows the updated Swagger UI. The title is now 'Example Api Title' and the subtitle is 'Example Api Definition'. Below this, it says 'Created by Chandana Napagoda' and 'See more at <http://blog.napagoda.com>'. There are also links for 'Contact the developer' and 'Apache 2.0'. The controller listings for 'admin-controller' and 'hello-world-controller' remain the same as in the previous screenshot. At the bottom, it says '[BASE URL: / , API VERSION: 1.0.0]'.

Controller and POJO Level Documentation

@Api annotation is used to explain each rest controller class.

@ApiOperation annotation is used to explain to describe the resources and methods.

@ApiResponse annotation is used to explain to describe other responses that can be returned by the operation.ex: 200 ok or 202 accepted, etc.

@ApiModelProperty annotation to describe the properties of the POJO(Bean) class.

After adding above annotation, final generated swagger documentation looks like below:



Example Api Title
Example Api Definition

Created by Chandana Napagoda
See more at <http://blog.napagoda.com>
[Contact the developer](#)
[Apache 2.0](#)

Admin API : Rest API for administrative operations Show/Hide List Operations Expand Operations

GET /admin/hello/{name} Display greeting message to admin user

Response Class (Status 200)
OK

Model | Example Value

Greeting {
 message (string, optional, read only): The system generated greeting message,
 player (string): Chandana
}

Response Content Type | application/json ▼

Parameters

Parameter	Value	Description	Parameter Type	Data Type
name	(required)	name	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	You are not authorized access the resource		
403	Forbidden		
404	The resource not found		

[Try it out!](#)

User API : Rest API for user operations Show/Hide List Operations Expand Operations

Spring RestController class:

```

01 package com.chandana.helloworld.controllers;
02
03 import com.chandana.helloworld.bean.Greeting;
04 import io.swagger.annotations.Api;
05 import io.swagger.annotations.ApiOperation;
06 import io.swagger.annotations.ApiResponse;
07 import io.swagger.annotations.ApiResponses;
08 import org.springframework.web.bind.annotation.PathVariable;
09 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RequestMethod;
11 import org.springframework.web.bind.annotation.RestController;
12
13 @RestController
14 @RequestMapping("/api")
15 @Api(value = "user", description = "Rest API for user operations", tags = "User API")
16 public class HelloWorldController {
17
18     @RequestMapping(value = "/hello/{name}", method = RequestMethod.GET, produces = "application/json")
19     @ApiOperation(value = "Display greeting message to non-admin user", response = Greeting.class)
20     @ApiResponses(value = {
21         @ApiResponse(code = 200, message = "OK"),
22         @ApiResponse(code = 404, message = "The resource not found")
23     })
24     public Greeting message(@PathVariable String name) {
25         Greeting msg = new Greeting(name, "Hello " + name);
26         return msg;
27     }
28 }

```

Greeting model class:

```

01 package com.chandana.helloworld.bean;
02
03 import io.swagger.annotations.ApiModelProperty;
04
05 public class Greeting {
06
07     @ApiModelProperty(notes = "Provided user name", required = true)
08     private String player;
09
10     @ApiModelProperty(notes = "The system generated greeting message", readOnly = true)
11     private String message;
12
13     public Greeting(String player, String message) {
14         this.player = player;
15         this.message = message;
16     }
17
18     public String getPlayer() {
19         return player;
20     }
21
22     public void setPlayer(String player) {
23         this.player = player;
24     }
25
26     public String getMessage() {

```

```

27         return message;
28     }
29
30     public void setMessage(String message) {
31         this.message = message;
32     }
33 }

```

AppConfig class:

```

01 package com.chandana.helloworld.config;
02
03 import org.springframework.context.annotation.Bean;
04 import org.springframework.context.annotation.Configuration;
05 import springfox.documentation.builders.ApiInfoBuilder;
06 import springfox.documentation.builders.PathSelectors;
07 import springfox.documentation.builders.RequestHandlerSelectors;
08 import springfox.documentation.service.ApiInfo;
09 import springfox.documentation.service.Contact;
10 import springfox.documentation.spi.DocumentationType;
11 import springfox.documentation.spring.web.plugins.Docket;
12 import springfox.documentation.swagger2.annotations.EnableSwagger2;
13
14 @Configuration
15 @EnableSwagger2
16 public class ApplicationConfig {
17
18     @Bean
19     public Docket api() {
20         return new Docket(DocumentationType.SWAGGER_2)
21             .apiInfo(getApiInfo())
22             .select()
23             .apis(RequestHandlerSelectors.basePackage("com.chandana.helloworld.controllers"))
24             .paths(PathSelectors.any())
25             .build();
26     }
27
28     private ApiInfo getApiInfo() {
29         Contact contact = new Contact("Chandana Napagoda", "http://blog.napagoda.com", "cnapagoda@gmail.com");
30         return new ApiInfoBuilder()
31             .title("Example Api Title")
32             .description("Example Api Definition")
33             .version("1.0.0")
34             .license("Apache 2.0")
35             .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0")
36             .contact(contact)
37             .build();
38     }
39 }

```

You can download Swagger Spring Boot Project source code from my GitHub repo as well.

Reference: Integrating Swagger with Spring Boot REST API from our JCG partner Chandana Napagoda at the Chandana Napagoda blog.

Tagged with: SPRING SPRING BOOT



(+1 rating, 1 votes)

You need to be a registered member to rate this. 6 Comments 4583 Views Tweet it!

Do you want to know how to develop your skillset to become a **Java Rockstar**?



Subscribe to our newsletter to start Rocking right now!
To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more

☐ I agree to the Terms and Privacy Policy

LIKE THIS ARTICLE? READ MORE FROM JAVA CODE GEEKS

[Join the discussion](#)

B *I* U



This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

6 COMMENTS

Oldest ▾

**Diego Alcorta** 4 years ago

How can you input user/pass if rest api is using either basic or oauth authentication

+ 0 — Reply

**Chandana Napagoda** (@chandana-napagoda) 4 years ago

Reply to [Diego Alcorta](#)

Author

Please refer "Secure Spring Boot REST API using Basic Authentication" post on this topic.

<http://blog.napagoda.com/2017/10/secure-spring-boot-rest-api-using-basic.html>

+ 0 — Reply

**William Ferraciolli** (@wilferraciolli) 4 years ago

Where does the swagger gets created? does it saves as yaml file?

+ 2 — Reply

**Jasleen khurana** 3 years ago

how can we customize swagger's default url i.e. /swagger-ui.html to something else? Also if i configure it for any controller in my application, then to access that URL I need to login to my application. I want to access the URL without requiring to Login to my application and later on authenticate my APIs using headers. Can you please help me on that?

+ 0 — Reply

**DevLearns** 2 years ago

Hi, I want to generate Swagger document by annotating the controller interface (not the implementing class), but fail to generate the documentation. Can you please point me to any existing example?

+ 0 — Reply

**cat** 1 year ago

ApiModelProperty readOnly set to true has no expected effect. It does make sense to define it for GET only requests. It should hide property in POST/PUT/PATCH, but in swagger 2.7.0 it does not.

+ 0 — Reply

KNOWLEDGE BASE[Courses](#)[Examples](#)[Minibooks](#)[Resources](#)[Tutorials](#)**PARTNERS**[Mkyong](#)**HALL OF FAME**["Android Full Application Tutorial" series](#)[11 Online Learning websites that you should check out](#)[Advantages and Disadvantages of Cloud Computing – Cloud computing pros and cons](#)[Android Google Maps Tutorial](#)[Android JSON Parsing with Gson Tutorial](#)[Android Location Based Services Application – GPS location](#)**ABOUT JAVA CODE GEEKS**

JCGs (Java Code Geeks) is an independent online community focused on ultimate Java to Java developers resource center; targeted at the technical team lead (senior developer), project manager and junior dev. JCGs serve the Java, SOA, Agile and Telecom communities with daily news, domain experts, articles, tutorials, reviews, announcements, code snippets, source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples of Java Code Geeks are not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

THE CODE GEEKS NETWORK

.NET Code Geeks

Java Code Geeks

System Code Geeks

Web Code Geeks

Android Quick Preferences Tutorial

Difference between Comparator and Comparable in Java

GWT 2 Spring 3 JPA 2 Hibernate 3.5 Tutorial

Java Best Practices – Vector vs ArrayList vs HashSet

