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.

```
<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.7.0</version>

</dependency>
```

```
<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.7.0</version>

</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

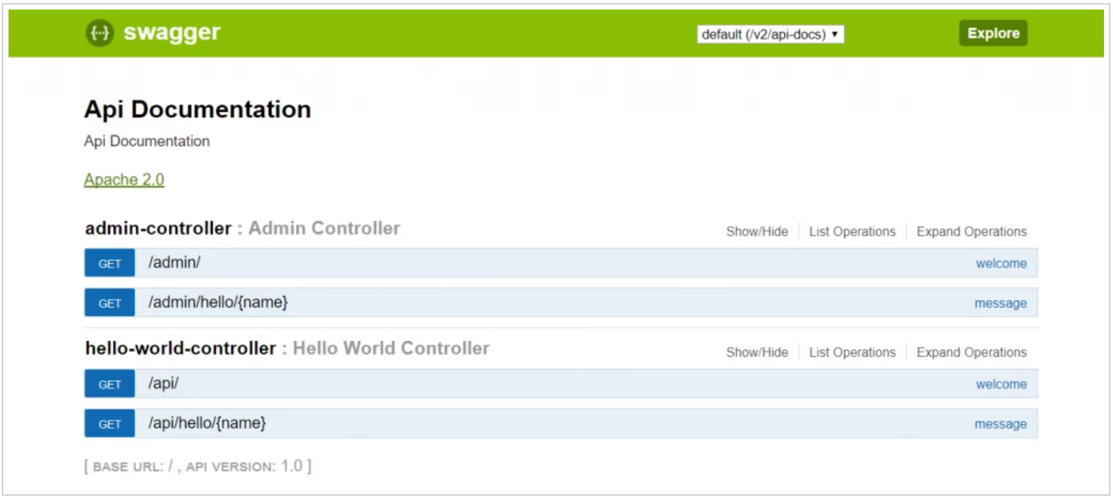
In My example, I am generating a swagger documentation based on the RestController classes I have added.

```
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class ApplicationConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.chandana.helloworld.controllers"))
                .paths(PathSelectors.any())
                .build();
    }
}
```

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



Api Documentation

Api Documentation

Apache 2.0

admin-controller : Admin Controller — Show/Hide | List Operations | Expand Operations

GET /admin/ welcome

GET /admin/hello/{name} message

hello-world-controller : Hello World Controller — Show/Hide | List Operations | Expand Operations

GET /api/ welcome

GET /api/hello/{name} message

[BASE URL: / , API VERSION: 1.0]

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

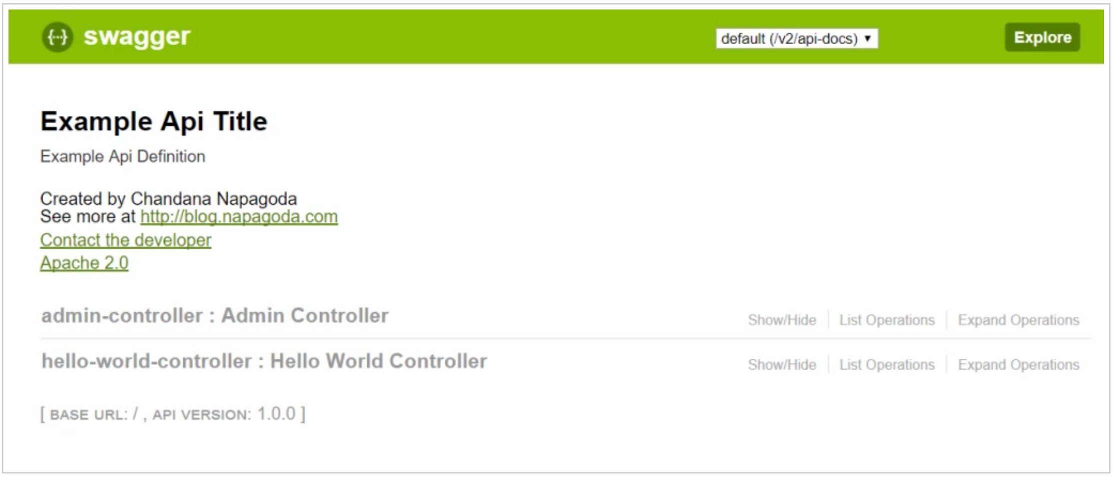
ApiInfo

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.

```
@Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(getApiInfo())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.chandana.helloworld.controllers"))
                .paths(PathSelectors.any())
                .build();
    }

    private ApiInfo getApiInfo() {
        Contact contact = new Contact("Chandana Napagoda", "http://blog.napagoda.com", "cnapagoda@gmail.com");
        return new ApiInfoBuilder()
                .title("Example Api Title")
                .description("Example Api Definition")
                .version("1.0.0")
                .license("Apache 2.0")
                .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0")
                .contact(contact)
                .build();
    }
```

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



Example Api Title

Example Api Definition

Created by Chandana Napagoda
See more at http://blog.napagoda.com
Contact the developer
Apache 2.0

admin-controller : Admin Controller — Show/Hide | List Operations | Expand Operations

hello-world-controller : Hello World Controller — Show/Hide | List Operations | Expand Operations

[BASE URL: / , API VERSION: 1.0.0]

Controller and POJO Level Documentation

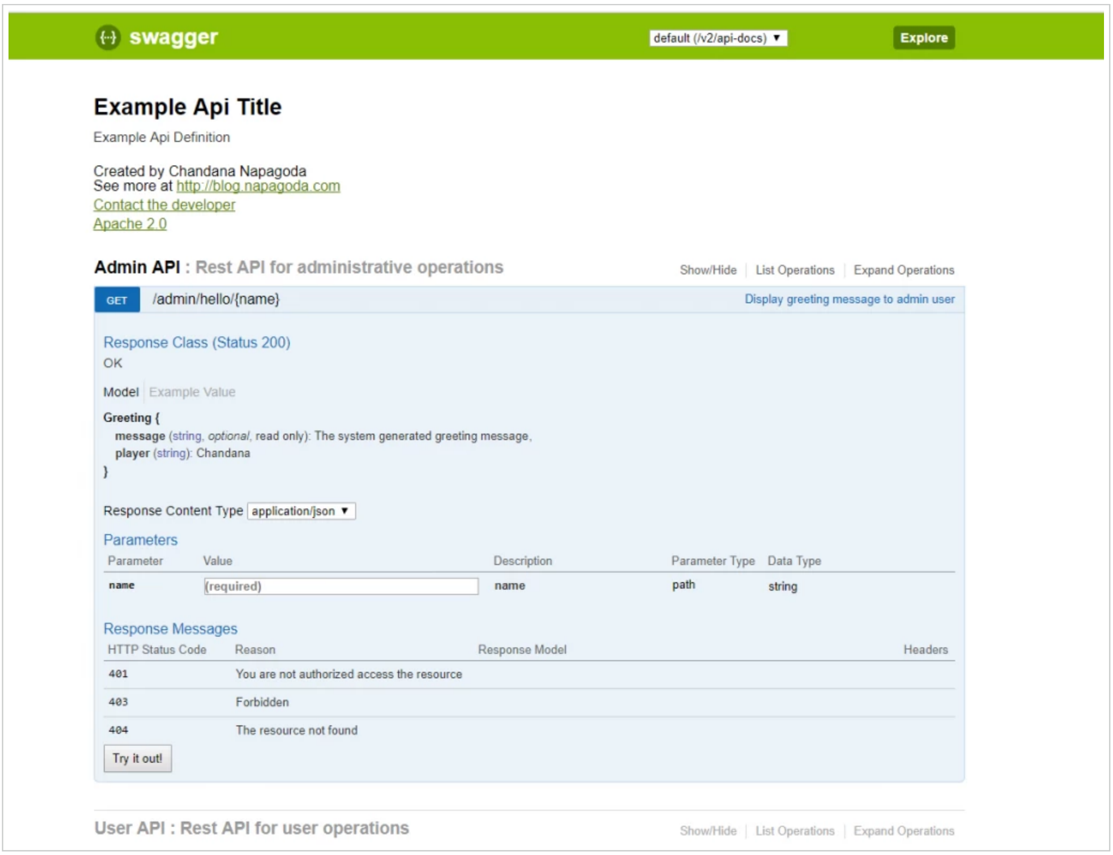
@Api annotation is used to explain each rest controller class.

@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:

```java
package com.chandana.helloworld.controllers;

import com.chandana.helloworld.bean.Greeting;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiResponse;
import io.swagger.annotations.ApiResponses;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api")
@Api(value = "user", description = "Rest API for user operations", tags = "User API")
public class HelloWorldController {

    @RequestMapping(value = "/hello/{name}", method = RequestMethod.GET, produces = "application/json")
    @ApiOperation(value = "Display greeting message to non-admin user", response = Greeting.class)
    @ApiResponses(value = {
            @ApiResponse(code = 200, message = "OK"),
            @ApiResponse(code = 404, message = "The resource not found")
    }
    )
    public Greeting message(@PathVariable String name) {
        Greeting msg = new Greeting(name, "Hello " + name);
        return msg;
    }
}
```

Greeting model class:

```java
package com.chandana.helloworld.bean;

import io.swagger.annotations.ApiModelProperty;

public class Greeting {

    @ApiModelProperty(notes = "Provided user name", required =true)
    private String player;

    @ApiModelProperty(notes = "The system generated greeting message" , readOnly =true)
    private String message;

    public Greeting(String player, String message) {
        this.player = player;
        this.message = message;
    }

    public String getPlayer() {
        return player;
    }

    public void setPlayer(String player) {
        this.player = player;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }
}
```

AppConfig class:

```java
package com.chandana.helloworld.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class ApplicationConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(getApiInfo())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.chandana.helloworld.controllers"))
                .paths(PathSelectors.any())
                .build();
    }

    private ApiInfo getApiInfo() {
        Contact contact = new Contact("Chandana Napagoda", "http://blog.napagoda.com", "cnapagoda@gmail.com");
        return new ApiInfoBuilder()
                .title("Example Api Title")
                .description("Example Api Definition")
                .version("1.0.0")
                .license("Apache 2.0")
                .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0")
                .contact(contact)
                .build();
    }
}
```

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 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!

LIKE THIS ARTICLE? READ MORE FROM JAVA CODE GEEKS

Subscribe

Join the discussion

B I U S {} [+]

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 o oauth authentication

+ 0 — Reply

Chandana Napagoda (@chandana-napagoda) 4 years ago

Author

Reply to *Diego Alcorta*

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

Wiliam Ferraciolli (@wilferraciolli) 4 years ago

Where does the swagger gets created? does it saves as yalm 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 o ultimate Java to Java developers resource center; targeted at the techn technical team lead (senior developer), project manager and junior dev JCGs serve the Java, SOA, Agile and Telecom communities with daily ne domain experts, articles, tutorials, reviews, announcements, code snipp source projects.

DISCLAIMER

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