**Spring Boot + Swagger2**

In this post we configure a spring boot application to integrate swagger2. [Spring Boot example](https://www.javainuse.com/spring/SpringBoot_HelloWorld) we had exposed a REST API . Documentation of such REST Services we develop is very important. This documentation should help consumers of the service know which all services are available, the signatures, the expected input. Also there should be some simple way to test if the service is up. The exposed services are bound to change so simultaneously the documentation would also need to be updated. If this is done manually then it will be a very tedious process, especially as the number of REST services increase. This is where swagger comes into picture. It helps automate this documentation process. Moreover, every change in the API should be simultaneously described in the reference documentation. Accomplishing this manually is a tedious exercise, so automation of the process was inevitable. In the next post we look at the [various Swagger annotations available and their use.](https://www.javainuse.com/spring/boot_swagger_annotations) In another post we look at [Using Swagger with Spring Boot Profile](https://www.javainuse.com/spring/boot_swagger_profile).  
Formerly known as the Swagger Specification, this format has been donated to the Open API Initiative (or OAI) which is a Linux Foundation Collaborative Project. We have [implemented Swagger using OpenAPI3 in another post](https://www.javainuse.com/spring/boot_swagger3).

Spring Boot Swagger- Table of Contents

[**Spring Boot + Swagger Example Hello World Example**](https://www.javainuse.com/spring/boot_swagger)

[Spring Boot + Swagger- Understanding the various Swagger Annotations](https://www.javainuse.com/spring/boot_swagger_annotations)

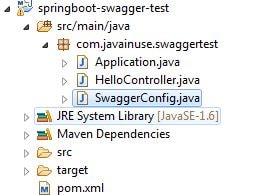
[Spring Boot + Swagger + Profile - Implementing Spring Boot Profile for a Swagger application](https://www.javainuse.com/spring/boot_swagger_profile)

[Spring Boot + Swagger 3 (OpenAPI 3) Hello World Example](https://www.javainuse.com/spring/boot_swagger3)

[Spring Boot + Swagger 3 (OpenAPI 3) + Security Example](https://www.javainuse.com/spring/boot_swaggersec)

**What is Swagger**

Swagger is widely used for visualizing APIs, and with Swagger UI it provides online sandbox for frontend developers. For the tutorial, we will use the Springfox implementation of the Swagger 2 specification. Swagger is a tool, a specification and a complete framework implementation for producing the visual representation of RESTful Web Services. It enables documentation to be updated at the same pace as the server. When properly defined via Swagger, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Thus Swagger removes the guesswork in calling the service.

The project will be as follows-  


In the Maven we need the swagger dependency.Maven will be as follows-

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.javainuse</groupId>

<artifactId>springboot-swagger-test</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.1.RELEASE</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**<dependency>**

**<groupId>io.springfox</groupId>**

**<artifactId>springfox-swagger2</artifactId>**

**<version>2.4.0</version>**

**</dependency>**

**<dependency>**

**<groupId>io.springfox</groupId>**

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

**<version>2.4.0</version>**

**</dependency>**

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**Note- Previously was using 2.2.2 version for springfox-swagger2 maven dependencies. But this**

**version has issues. If overloaded methods are used for exposing REST API it will not work properly. Only for one of the overloaded methods the**

**swagger documentation can be seen and not for the other.**

Create the Application.java as below-

package com.javainuse.swaggertest;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

@RequestMapping maps /api/javainuse request to sayHello() method.

package com.javainuse.swaggertest;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

@RequestMapping(method = RequestMethod.GET, value = "/api/javainuse")

public String sayHello() {

return "Swagger Hello World";

}

}

To enable the Swagger 2 we use the annotation @EnableSwagger2.  
A Docket bean is defined and using its select() method we get an instance of ApiSelectorBuilder. ApiSelectorBuilder we configure the endpoints exposed by Swagger.  
After the Docket bean is defined, its select() method returns an instance of ApiSelectorBuilder, which provides a way to control the endpoints exposed by Swagger.  
Using the RequestHandlerSelectors and PathSelectors we configure the predicates for selection of RequestHandlers.

package com.javainuse.swaggertest;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import com.google.common.base.Predicate;

import springfox.documentation.builders.ApiInfoBuilder;

import springfox.documentation.service.ApiInfo;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

import static springfox.documentation.builders.PathSelectors.regex;

import static com.google.common.base.Predicates.or;

@Configuration

@EnableSwagger2

public class SwaggerConfig {

@Bean

public Docket postsApi() {

return new Docket(DocumentationType.SWAGGER\_2).groupName("public-api")

.apiInfo(apiInfo()).select().paths(postPaths()).build();

}

private Predicate<String> postPaths() {

return or(regex("/api/posts.\*"), regex("/api/javainuse.\*"));

}

private ApiInfo apiInfo() {

return new ApiInfoBuilder().title("JavaInUse API")

.description("JavaInUse API reference for developers")

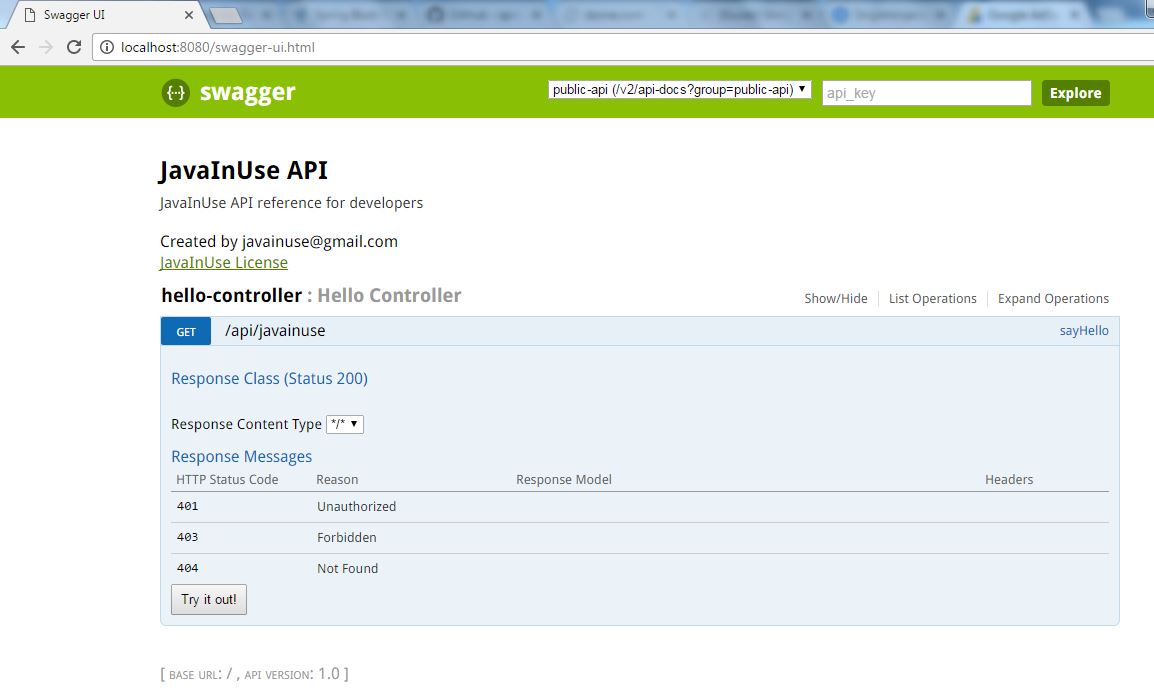
.termsOfServiceUrl("http://javainuse.com")

.contact("javainuse@gmail.com").license("JavaInUse License")

.licenseUrl("javainuse@gmail.com").version("1.0").build();

}

}

These are the only changes required. Now go to **http://localhost:8080/swagger-ui.html**.  
We will see the documentation for the exposed API as follows-  
  
  
Using the Try it button we can also check if the service is up.  
