http://localhost:port/swagger-ui.html

What is Swagger?

* **Swagger** is a tool that enables the documentation of RESTful services
* It provides an implementation for producing the visual representation of the RESTful services
* It specifies the RESTful services format including service url, method signatures, inputs etc.

Now, open the eclipse ide and let’s see how to integrate Swagger2 into Spring Boot RESTful Api example. Do note, we are going to use the [Springfox](https://springfox.github.io/springfox/" \t "_blank) implementation to generate the Swagger documentation.

Spring Boot REST API with Swagger Example

Maven Dependencies

Here, we specify the dependencies for the Spring Boot and Swagger2. Maven will automatically resolve the other dependencies. The **updated** file will have the following code.

*pom.xml*

<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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.springboot.swagger</groupId>

<artifactId>Springbootswagger</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>Springbootswagger Maven Webapp</name>

<url>http://maven.apache.org</url>

<!-- Spring boot parent dependency jar -->

<parent>

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

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

<version>2.1.1.RELEASE</version>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.github.javafaker</groupId>

<artifactId>javafaker</artifactId>

<version>0.16</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

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

<version>2.9.2</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

</dependencies>

<build>

<finalName>Springbootswagger</finalName>

</build>

</project>

3.2.2 Swagger Configuration class

Add the following the Swagger Configuration class. The class is annotated with @EnableSwagger2 annotation to enable the swagger support in your application. Here we have added a metadata() method that returns the ApiInfo object with the information about the application information.

*Myapplication.java*

// The static import is used for the regex(..) method.

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

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;

@Configuration

@EnableSwagger2

public class Myswaggerconfig {

@Bean

public Docket postsApi() {

return new Docket(DocumentationType.SWAGGER\_2). apiInfo(metadata()).select().paths(regex("/emp.\*")).build();

}

@SuppressWarnings("deprecation")

private ApiInfo metadata() {

return new ApiInfoBuilder().title("Capgemini").description("API reference guide for developers").termsOfServiceUrl("https://www....../").contact("Batra, Yatin").version("1.0").build();

}

}

3.2.3 Controller class

Add the following code to the controller class designed to handle the incoming requests. This class defines the REST API endpoints which are configured by the @RequestMapping annotation.

*Mycontroller.java*

package com.example.howtodoinjava.springhystrixstudentservice.controller;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

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;

import com.example.howtodoinjava.springhystrixstudentservice.domain.Student;

@RestController

public class EmployeeController {

//api-docs

//swagger-ui.html

@RequestMapping(value = "/emp/{id}", method = RequestMethod.GET)

public String getEmp() {

return "Emp1";

}

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

public String getAllEmp() {

return "All Emp1";

}

@RequestMapping(value = "/emp/{id}", method = RequestMethod.DELETE)

public String deleteEmp() {

return "DeleteEmp1";

}

@RequestMapping(value = "/emp/{id}", method = RequestMethod.POST)

public String addEmp() {

return "AddEmp1";

}

}

4. Run the Application

As we are ready with all the changes, let us compile the spring boot project and run the application as a java project. Right click on the Myapplication.java class, Run As -> Java Application.

Fig. 6: Deploy the Application

Developers can debug the example and see what happens after every step. Enjoy!

5. Project Demo

Open your favorite browser and hit the following URL to display the index page of the swagger2 tool.

http://localhost:8080/swagger-ui.html

Fig. 7: Swagger documentation

The index page will display the documentation for the exposed api as shown in Fig. 2. Users can click the Try button to check if the service is up or not.