Swagger API

==

API Creation/Development :: developing RestController having different methods/opertions for various http method types like GET,POST,PATCH,PUT, DELETE and etc... is called API Creation/Development.

=>We can generally take one @RestController module, So developing each RestController is called api creation..

Accounts module ---> AccountsController (@RestController) is called Acccounts API creation Daily Tx module ---> DialyTxController (@RestController) is called DailyTx API creation and etc..

EndPoints ::

The Restful webService provider/server/producer app development is called API creation.

Providing muliple details or collection of details that are required to call methods/opertions of @RestController from Client Apps or to send reqeusts from different tools like POSTMAN is called Providing end points..

Eg:: For AccountController nothing but Accounts API the end points are

the

BaseURL:: http://localhost:3030/RestProj1/accountapi

register()

/register -> POST (endpoint1)

findById(-): /find/{id} ----> GET (endpoint2)

deleteById(-): /delete/{id} --->DELETE (endpoint3)

and etc..

In End Points of any API we need to provide multiple details like

Each @RestController development is called API creation

=>Each Rest API/ Rest Controller is called is called API

URL,method names, request paths, http method types,content type path variables

(GET,POST,PUT,....)

and each method in that Rest Controller is called one end point

and etc..

store Access to Petstore orders

/store/inventory Retums pet inventories by status

`/store/order Hace an order for a pet`

(rest operation1) (rest operation2)

GET

/store/order/{orderId) Find purchase order by ID

DELETE /store/order/(orderId) Delete purchase order by ID

(rest operation3) (rest operation4)

API Documentation

============

=>we can write documentation for java classes in multiple ways a) using serperate Text docs

(/** .... */)

Here we are giving EndPoint details of Store API

note:: POSTMAN is given for API Testing where as swagger given for API Documentation cum testing (This is called Responsive Documentation)

b) using API documentation comments and javadoc tool (best) note:: Both these approaches fon-responsive documentations i.e we can read about java classes and methods but we can not test them immediately

=> After developing Rest API we can provide API documentation for Rest API

in the following ways

a) Using Seperate Text docs

b) Using API doc comments (/ ... */)**

c) Using Swagger /Swagger API (Best)|

(or) Open API

(swagger 3.x)

Creates Non-Responsive

API documentation (bad)

Creates Responsive API Documentation. (good)

w3schools.com provides the responsive document on HTML tags i.e we not only read info about HTML tags and we can also test them

note:: swagger api internallys

uses boostrap support

to showcase the

responsive documentation

note:: swagger API or Open API gives API documentation only on @RestController classes, not on service, Repository classes

note:: Open API is alternate to Swagger API .. The industry standard is still Swagger API (upto spring boot 2.x) note: Responsive documentation means we can read about classes and we can test them there itself Swagger API

eg:w3school.com, swagger api/open API docs generated for restfull API

(Good upto spring boot 2.x)

=> It is an open Source Third Party Library to provide Responsive API documentation for RestController and its methods

=> For All RestControllers of the Project we can create API documentation from single place while working Swagger API

with

=>Responsive Documentation means not only we get docs about API and its methods (End points) we can

test immediately by providing inputs and by getting outputs..

=> if this is used.. there is no need of using POSTMAN tool seperately.. and also very useful to provide Documentation based Testing env.. for Clients.

togather

=> Spring Fox +swagger

spring boot Applications

released libraries that are required to use swagger api in

=>Swagger API documentation provides the following details in the GUI Responsive docs

a) API Info (company, title, license url, and etc..)

b) End points Info

c) Model classes info

and etc..

with

=> While working swagger documentation for reading and testing we need not to remember and give URL, Http method types, content type and etc.. we just need to give required inputs and get the outputs.

note:: Using POSTMAN tool we can just test Restful APIS where as using swagger /open API we can get response docs of restful APIs using which we can also test the the rest opeations

Procedure to work with swagger API (this only in spring boot 2.x rest apps) (Legacy or outdated)

====

=========

step1) keep RestController/Rest API Project ready.. (makes sure that it is using spring boot 2.5.7 version) (Tomcat 9.x server)

step2) Add the following two jar file in pom.xml related to swagger API

a) springfox-swagger2

a) springfox-swagger-ui

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui --> <dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger-ui</artifactId>
<version>2.9.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
<dependency>
<groupId>io.springfox</groupId>
<artifactId>springfox-swagger2</artifactId>
<version>2.9.2</version>
</dependency>
```

step3) Develope seperate Configuration class Enabling enabling Swagger api

=>In configuration class create Docket object having

->Dcoumention type (screent type)

->specify base package of restControllers

->specify reqeusts paths info

->other details of API (ApiInfo obj having company name, licenseurl and etc.)

SwaggerDocConfig.java

package com.nt.config;

import java.util.Collections;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import springfox.documentation.builders.PathSelectors;

import springfox.documentation.builders.RequestHandlerSelectors;

import springfox.documentation.service.ApiInfo;

note:: From spring boot 3.x the industry is preferring to use open api (swagger 3.x api)

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 (or) @EnableSwagger2WebMvc

public class SwaggerDocsConfig {

@Bean

public Docket createDocket() {

}

return new Docket(DocumentationType.SWAGGER_2) //UI screen type

.select() //to specify RestControllers

.apis(RequestHandlerSelectors.basePackage("com.nt.controller")) //base pkg for RestContrllers
.paths(PathSelectors.regex("/tourist.*")) // to specify request paths

.build() // builds the Docket obj

.useDefaultResponseMessages(true)

.apiInfo(getApiInfo());

private ApiInfo getApiInfo() {

Contact contact=new Contact("raja","http://www.HCL.com/tourist","natarazjavaarena@gmail.com"); return new ApiInfo("Tourist API",

"Gives Info Tourist Activites", "3.4.RELEASE",

"http://www.hcl.com/license",

contact,

"GNU Public",

"http://apache.org/license/gnu",

Collections.emptyList());

}

}

step4) Run the Server app

step5) use the following url to get swagger api docs and to test the api

http://localhost:3030/SpringBoot Rest Proj12-MiniProject-SwaggerAPI/swagger-ui.html

Tourist API

3.A.RELEASE

[Base URL: localhost:8080/SpringBoot ReatProj 12-Mini Project-SwaggerAPI]

http://localhost:3030/Spring Boot Rest Proj12-MiniProject-SwaggerAPI/v2/api-docs

Gives Info Tourist Activites

Terms of service

raja - Website

Send email to raja

GNU Public

tourist-operations-controller Tourist Operations Controller

PATCH

/tourist/budgetModify/{id}/{hike} modify TouristBudgetByld

DELETE /tourist/delete/{id} removeToursit

GET /tourist/find/{id} display TouristByld

GET /tourist/findAll displayToursits

PUT

/tourist/modify modifyTourist

POST /tourist/register For Tourist registration

@ApiOperation("......") can be applied on the rest API/controller methods to provide our choice

description and that reflects in swagger docs.

```
@PostMapping("/register")
@ApiOperation("For Tourist registration")
public ResponseEntity<String> enrollTourist (@RequestBody Tourist tourist){
try {
//use service
String resultMsg-service.registerTourist(tourist);
return new ResponseEntity<String>(resultMsg,
}
HttpStatus.CREATED); //201 content created successfully
catch (Exception e) {
e.printStackTrace();
return new ResponseEntity<String>("problem in tourist enrollment",
HttpStatus.INTERNAL_SERVER_ERROR); //500 error
}
```

}//method

(Swagger 3) API

Swagger 3.x Open API

======

Spring Boot Rest API documentation using Open

===========================

=> It is compitable with spring boot 3 version (Especially spring boot rest 3.x)

=> No need of developing separate Configuraton class having the swagger api

=> The starter /jar file that is added automatically detects the @RestController classes and generates api documentation internally by giving swagger-ui.html file

Open API is the extension of swagger API

tourist-operations-controller

`PUT /tourist-api/modify`

Example App

===========

step1) Keep spring boot rest mini Project ready by taking the version 3.x+

11.x

step2) make sure that Tomcat 10.x server is configured

step3) add the following additional stater/jar file to pom.xml file

```
<!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui -->
<dependency>
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
<version>2.3.0</version>
</dependency>
```

11.x

step4) Run the Application in tomcat 10.x server

step5) Test the Application

https://localhost:3030/MiiniProject/swagger-ui.html

```
POST /tourist-api/register
PATCH /tourist-api/pmodify/{id}/{discPercen}
V
GET /tourist-api/findById/{id}
GET /tourist-api/findAll
GET /tourist-api/findAll/{city1}/{city2}/{city3)
DELETE /tourist-api/remove/{start}/{end)
```

`DELETE /tourist-api/remove/{id}`

Please collect it from mvnrepository.com

(search for springdoc -open api)

step6) we can use the following annotations to add custom info in the api documentation

Swagger 2 to Swagger 3 annotations

(Open API)

Swagger 3 is an updated version of Swagger 2 and has some changes in annotations:

2.x annotations 3.x annotations

`@Api`

@Tag

• `@ApiIgnore`

@Parameter (hidden = true) or @Operation (hidden = true) or

@Hidden

•

•

• `@ApiModel`

→

`@Schema`

`@ApiImplicitParam`

@Parameter

`@ApiImplicitParams @Parameters`

@ApiModelProperty(hidden = true)

`@Schema(accessMode = READ_ONLY)`

`@ApiModelProperty`

→ `@Schema`

•

`@ApiOperation(value = "foo", notes = "bar")`

→

@Operation(summary = "foo",

`description = "bar")`

•

`@ApiParam → @Parameter`

•

`@ApiResponse (code = 404, message = "foo")`

→

@ApiResponse(responseCode =

`"404", description = "foo")`

```
@RestController
@RequestMapping("/tourist-api")
@Tag(name = "tourist",description = "Tourist Operations controller")
public class TouristOperationsController {
@Autowired
private ITourist MgmtService touristService;
@Operation(summary = "saves the tourist info",description = "persist the tourist info")
@ApiResponses(value = {
@ApiResponse(responseCode = "201", description = "Tourist saved",
content = { @Content(mediaType = "application/json",
schema = @Schema(implementation Tourist.class))})})
@PostMapping("/save")
public ResponseEntity<String> registerTourist(@RequestBody Tourist tourist){
try {
//use service
String msg=touristService.registerTourist(tourist);
return new ResponseEntity<String>(msg,HttpStatus.CREATED); //201 created
}
catch(Exception e) {
e.printStackTrace();
}
}//method
return new ResponseEntity<String>(e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
```

tourist Tourist Operations controller

```
PUT
/tourist-api/update
POST /tourist-api/save saves the tourist Info
PATCH
/tourist-api/update/{id}/{percentage}
GET
/tourist-api/report
GET /tourist-api/find/{id}
DELETE /tourist-api/delete/{id}
```

Talking: Naresh II

Key Differences Between Postman vs Swagger

Here are 10 key differences between Postman and Swagger that you need to know to decide which tool is tailored to your requirements.

Aspect

Primary

Purpose

Postman

Swagger

API testing and development tool. API design, documentation, and testing.

Graphical user interface (GUI) for

User Interface

making API requests.

Automation of API tests and

Automation

workflows is a core feature.

Code Generation

Limited code generation for client applications.

Interactive Docs

Basic documentation capabilities.

API Monitoring

Collaboration

Supported

Methods

Limited monitoring capabilities.

Collaboration features are available, but not as central. Supports various HTTP methods

and custom scripts for testing.

GUI for designing, documenting, and exploring APIs.

While it supports automation, it is more focused on API design and documentation.

Supports extensive code generation for client SDKs and server stubs.

Offers interactive and comprehensive API documentation (Swagger UI). Responsitve documentation

Primarily designed for API design and documentation, with some integrations for monitoring.

Collaboration and version control are central features.

Provides a standardized way to define HTTP

methods and API structure.

Commonly used in API design and

Widely used by developers and

Industry Usage

documentation, especially in large-scale

projects.

testers for API testing.

Compatible with a wide range of

API Ecosystem

APIs and services.

Focuses on defining and describing APIs using

the OpenAPI Specification.

V