<https://github.com/SpringForAll/spring-boot-starter-swagger>

//下面这个博客说明简单点

https://blog.csdn.net/sanyaoxu\_2/article/details/80555328

Swagger 默认访问地址 http://localhost:8080/swagger-ui.html

第一步 导入依赖

<dependency>  
 <groupId>com.spring4all</groupId>  
 <artifactId>swagger-spring-boot-starter</artifactId>  
 <version>1.7.0.RELEASE</version>  
</dependency>

第二步 在pojo controller 同级目录下建立 config目录 配置SwaggerConfiguration



对应代码

@Configuration  
@EnableSwagger2

public class SwaggerConfiguration {  
 //通过多个docket 每个组查阅不同的controller  
 @Bean  
 //Environment是springframework里的Environment  
 public Docket createRestApia(Environment environment) {  
 //获取环境 如果环境是下面写的两个其中一个 flag为true如果都不是则为false  
 Profiles profiles = Profiles.*of*("dev","test");  
 //传递环境  
 boolean flag = environment.acceptsProfiles(profiles);  
 return new Docket(DocumentationType.*SWAGGER\_2*)  
 .apiInfo(apiInfo())  
 .enable(flag)  
 .groupName("B");  
 }  
  
 @Bean  
 public Docket createRestApimf(Environment environment) {  
 //获取环境 如果环境是下面写的两个其中一个 flag为true如果都不是则为false  
 Profiles profiles = Profiles.*of*("dev","test");  
 //传递环境  
 boolean flag = environment.acceptsProfiles(profiles);  
 return new Docket(DocumentationType.*SWAGGER\_2*)  
 .apiInfo(apiInfo())  
 //代表右上角的分组 多个Docket有多个组显示  
 .groupName("mf")  
 // .enable(false)表示不启动  
 .enable(flag)  
 .select()  
 //是swagger scan base package,这是扫描注解的配置，即你的API接口位置。  
 // 用这个才能把有api接口的东西显示在网上  
 //指定扫描的包  
 .apis(RequestHandlerSelectors.*basePackage*("com.tianzej9.swaggerpractise.controller"))  
 //any()扫描全部  
 .paths(PathSelectors.*any*())  
 .build();  
 }  
  
 */\*\*  
 \* 创建该API的基本信息（这些基本信息会展现在文档页面中）  
 \* 访问地址：http://项目实际地址/swagger-ui.html  
 \** ***@return*** *\*/* private ApiInfo apiInfo() {  
 return new ApiInfoBuilder()  
 .title("Spring Boot中使用Swagger2构建RESTful APIs")  
 .description("更多请关注http://www.baidu.com")  
 .termsOfServiceUrl("http://www.baidu.com")  
 .version("1.0")  
 .build();  
 }  
  
  
}

## **4：添加文档内容**

在完成了上述配置后，其实已经可以生产文档内容，但是这样的文档主要针对请求本身，描述的主要来源是函数的命名，对用户并不友好，我们通常需要自己增加一些说明来丰富文档内容。

****Swagger使用的注解及其说明：****

[@Api](https://my.oschina.net/u/2396174)：用在类上，说明该类的作用。

@ApiOperation：注解来给API增加方法说明。

@ApiImplicitParams : 用在方法上包含一组参数说明。

@ApiImplicitParam：用来注解来给方法入参增加说明。

@ApiResponses：用于表示一组响应

@ApiResponse：用在@ApiResponses中，一般用于表达一个错误的响应信息

    l   ****code****：数字，例如400

    l   ****message****：信息，例如"请求参数没填好"

    l   ****response****：抛出异常的类

@ApiModel：描述一个Model的信息（一般用在请求参数无法使用@ApiImplicitParam注解进行描述的时候）

    l   ****@ApiModelProperty****：描述一个model的属性

注意：@ApiImplicitParam的参数说明：

|  |  |
| --- | --- |
| ****paramType****：指定参数放在哪个地方 | header：请求参数放置于Request Header，使用@RequestHeader获取  query：请求参数放置于请求地址，使用@RequestParam获取  path：（用于restful接口）-->请求参数的获取：@PathVariable  body：（不常用）  form（不常用） |
| name：参数名 |  |
| dataType：参数类型 |  |
| required：参数是否必须传 | true | false |
| value：说明参数的意思 |  |
| defaultValue：参数的默认值 |  |

类如传实体类显示

@ApiModel("用户实体类") //用在类上  
public class User {  
 @ApiModelProperty("用户名")  
 public String user;  
  
}

@RestController  
public class SuanshuController {  
//必须要这样子传过去才会在swagger有显示  
@PostMapping("/User")//post是增加内容  
 public User user(){  
 return new User();  
 }  
  
  
}

具体展示



方法测试

//给方法注释  
@ApiOperation("一个测试方法")  
@GetMapping("/test")  
//给参数注释  
public String test(@ApiParam("用户名") String name){  
 return "Hello"+name;  
}

