链滴

Swagger UI 初体验

作者：flowaters

原文链接：https://ld246.com/article/1534735914420

来源网站：链滴

前言

项目组中使用了SpringBoot + Swagger UI来生成Restful API文档和测试页面。

本文来体验一下Swagger UI的用法。

简介

Swagger UI是目前最流行的RestFul接口API文档和测试工具，可以直接在官方demo上进行体验。

本文介绍下如何在SpringBoot2中集成Swagger UI。

初体验

非常简单，只需要两步即可。

依赖

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
```

第二个依赖中包含有前端js/css资源。

编写配置文件

```
package com.example.testredisreactive;

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 SwaggerConfig {
```

```
    @Bean
    public Docket createRestApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                .apis(RequestHandlerSelectors.any())
                .paths(PathSelectors.any())
                .build();
    }

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .title("后端接口标题")
                .description("后端接口描述")
                .contact(
                    new Contact("flowaters", "note.abeffect.com", "flowaters@abeffect.com")
                )
                .version("1.0.0-SNAPSHOT")
                .build();
    }
}
```

使用

打开浏览器，访问http://localhost:8080/swagger-ui.html即可

api选择器

如何只筛选出指定的API呢?

包名

一种方法是，根据包名来筛选

```
    @Bean
    public Docket createRestApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.example"))
                .paths(PathSelectors.any())
                .build();
    }
```

path

另一种方法是，根据path来筛选

```
    @Bean
    public Docket createRestApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
```

```
            .select()
            .apis(RequestHandlerSelectors.basePackage("com.example"))
            .paths(PathSelectors.ant("/get"))
            .build();
    }
```

当然也可以两者配合来筛选。

详细文档

怎么样生成详细的文档呢?

通过在接口类上增加对应的注解，如下面的示例。

- 在类上增加 Api注解
- 在方法上增加 ApiOperation注解
- 在参数上增加 ApiParam注解
- 在模型字段上 ApiModelProperty增加注解

具体如下:

```
package com.example.testredisreactive;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.core.StringRedisTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;

@RestController
@RequestMapping("/api/1.0/kv")
@Api(description = "KV存储相关接口的描述", tags = "KV存储相关接口的TAG")
public class TestController {
    private static final Logger logger = LoggerFactory.getLogger(TestController.class);

    @Autowired
    RedisTemplate redisTemplate;

    @Autowired
    StringRedisTemplate stringRedisTemplate;

    @GetMapping(value = "/set")
    @ApiOperation(notes = "使用默认的序列化方法", value = "设置KV对")
    public void set(@ApiParam(required = true, value = "key") String key,
```

```
                @ApiParam(required = true, value = "value") String value) {
        stringRedisTemplate.opsForValue().set(key, value);
    }

    @GetMapping(value = "/get")
    @ApiOperation(value = "查询KV对")
    public String get(@ApiParam(required = true, value = "KV对中的key, 字符串类型")
                      @RequestParam(defaultValue = "key", required = false) String key) {
        return stringRedisTemplate.opsForValue().get(key);
    }
    @GetMapping(value = "/getdo")
    public KVDO getDo(String key) {
        String value = stringRedisTemplate.opsForValue().get(key);
        return new KVDO(key, value);
    }

}
```

数据模型示例

```
package com.example.testredisreactive;

import io.swagger.annotations.ApiModelProperty;
import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class KVDO {

    @ApiModelProperty(required = true, value = "KVDO中的键")
    private String key;

    @ApiModelProperty(required = true, value = "KVDO中的值")
    private String value;
}
```

参考

- swagger.io