链滴

# SpringCloud Alibaba 微服务实战十一 - Swagger 接口文档聚合

作者：jianzh5

原文链接：https://ld246.com/article/1581842575333

来源网站：链滴

导读：在SpringCloud体系架构中，我们需要的每个服务都需要对外输出接口文档，本篇内容主要是我们的微服务配上Swagger的接口文档，并在网关层完成接口聚合。

# Swagger2简介

在当下很多项目都会采用前后端分离的模式，前端和后端的工作由不同的开发人员完成。在这种开发式下，我们需要维护一份及时更新且完整的Rest API接口文档。传统意义上的文档都是后端人员在开相关接口后手动更新到接口文档上，但是这种方式很难保证文档的及时性，而且由于一些原因后端开人员可能会忘记更新，这样就会导致随着开发的进行接口文档会失去他本身的参考意义，反而会增加通成本。而 Swagger 给我们提供了一个全新的维护 API 文档的方式，他有以下几个作用：

- 只需要后端开发人员给接口加上几个注解，Swagger就可以根据代码自动生成API文档，节省编写口文档的工作量，而且对接口修改时只需要修改相应的注解，能保证接口的及时性；
- SwaggerUI展现出来的是一份可交互式的API文档，我们可以直接在接口页面对功能测试，省去了量接口联调的时间；

Swagger2有以下几个常见注解，大家在使用过程中会经常用到。

## @Api

此注解可以用来标记 Controller 的功能，如：

```
@Api(tags = "product模块")
public class ProductController implements ProductFeign {

}
```

## @ApiOperation

此注解用来标记一个方法的作用

```
@ApiOperation(value = "根据产品编码查找对应的产品")
public ResultData<ProductDTO> getByCode(@PathVariable String productCode){
    ...
}
```

## @ApiImplicitParam，@ApiImplicitParams

这组注解用来对参数进行描述，@ApiImplicitParam 有几个重要的参数：

name：参数名

value：参数的汉字说明、解释

required：参数是否必须传

paramType：参数放在哪个地方（header： 对应@RequestHeader的参数；query ： 对应@RequesParam的参数；path ： 对应@PathVariable的参数；body：对应 @RequestBody 的参数；form（通表单提交） ）

```
@ApiImplicitParam(name = "productCode",value = "产品编码", required = true,paramType = path")
public ResultData<ProductDTO> getByCode(@PathVariable String productCode){
```

```
    ...
}
@ApiImplicitParam(name = "productCode" , value = "产品编码",required = true, paramType =
"query")
public ResultData<String> delete(@RequestParam String productCode){
    ...
}
```

如果有多个参数，则需要使用@ApiImplicitParams 注解。

## @ApiModel，@ApiModelProperty

如果参数是一个对象，则需要在对象所在的类上加上此注解。这种一般用在post创建的时候，使用 @equestBody 这样的场景，请求参数无法使用 @ApiImplicitParam 注解进行描述的时候 。在具体字上则使用@ApiModelProperty注解。如：

```
@Data
@ApiModel(value = "产品封装类ProductDTO",description = "产品相关信息封装，用于接口传参")
public class ProductDTO {
    @ApiModelProperty(value = "产品主键")
    private Integer id;
    @ApiModelProperty(value = "产品编码")
    private String productCode;
    @ApiModelProperty(value = "产品名称")
    private String productName;
    @ApiModelProperty(value = "数量")
    private Integer count;
    @ApiModelProperty(value = "单价")
    private BigDecimal price;
}
```

# 使用

在项目中要使用Swagger2很简单，按照以下几步即可：

- 在pom文件中引入jar包

每个微服务都需要使用，我们直接将其引入在cloud-common服务中

```
<!--swagger2-->
<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>
```

- 在微服务中编写swagger2的配置类　SwaggerConfig

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {
    private static final String VERSION = "1.0.0";
    /**
     * 创建API
     */
    @Bean
    public Docket createRestApi(){
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                //指定接口包所在路径
                .apis(RequestHandlerSelectors.basePackage("com.javadaily.product.controller"))
                .paths(PathSelectors.any())
                .build();
    }

    /**
     * 添加摘要信息
     */
    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .title("product-server接口文档")
                .contact(new Contact("JAVA日知录","http://javadaily.cn","jianzh5@163.com"))
                .description("product-server接口文档")
                .termsOfServiceUrl("http://javadaily.cn")
                .license("The Apache License, Version 2.0")
                .licenseUrl("http://www.apache.org/licenses/LICENSE-2.0.html")
                .version(VERSION)
                .build();
    }
}
```

**.apis方法用于指定生成注解的范围，有以下四种取值逻辑**

①RequestHandlerSelectors.any()，为所有接口都生成API文档，这种方式不必在接口上加任何注解但是生成的文档没有任何注释，可读性不高；

②RequestHandlerSelectors.basePackage(xx.xx)，为指定包下的controller生成接口文档

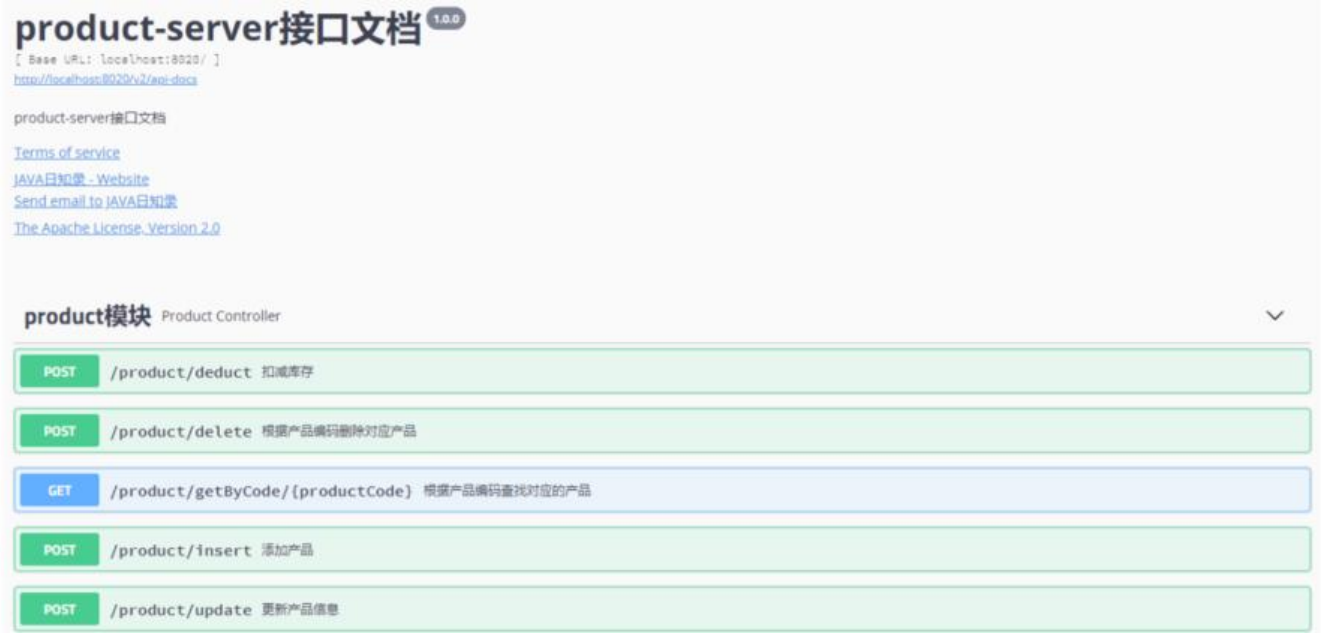
③RequestHandlerSelectors.withClassAnnotation(Api.class)，为有@api注解的接口生成api文档

④RequestHandlerSelectors.withMethodAnnotation(ApiOperation.class)，为有@ApiOperation解的方法生成API文档。

- 给相关接口加上Swagger的注解

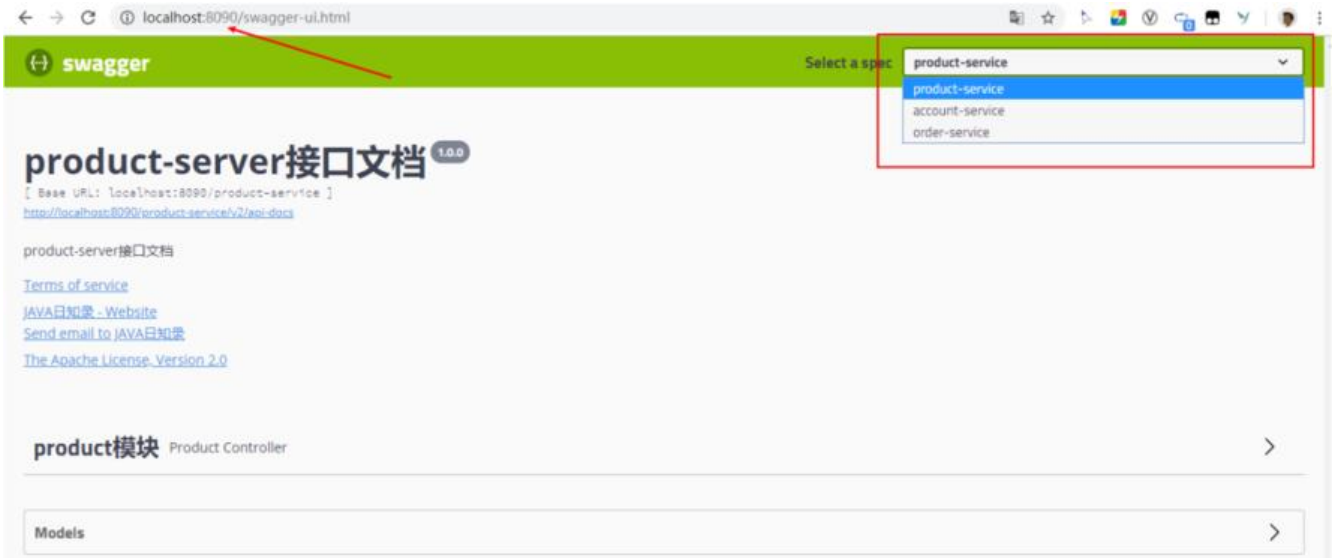
注解的详细说明参见上文。

- 打开swagger2接口API页面　http://localhost:8020/swagger-ui.html




# 网关聚合

我们已经给各个微服务加上了api文档，但是每次都需要分别输入各个微服务的文档地址，不太方便
本节内容主要是将各个微服务的api地址聚合到网关层，打开网关的api地址即可查看其它所有服务的
口文档，效果如下：




实现步骤如下：

- 编写配置类实现 SwaggerResourcesProvider 接口聚合其他微服务的api资源

```
@Component
@AllArgsConstructor
public class CustomSwaggerResourceProvider implements SwaggerResourcesProvider {
    /**
     * Swagger2默认的url后缀
     */
    public static final String SWAGGER2URL = "/v2/api-docs";
    /**
     * 网关路由
```

```
     */
    private final RouteLocator routeLocator;
    private final GatewayProperties gatewayProperties;
    /**
     * 聚合其他服务接口
     * @return
     */
    @Override
    public List<SwaggerResource> get() {
        List<SwaggerResource> resourceList = new ArrayList<>();
        List<String> routes = new ArrayList<>();
        //获取网关中配置的route
        routeLocator.getRoutes().subscribe(route -> routes.add(route.getId()));
        gatewayProperties.getRoutes().stream().filter(routeDefinition -> routes.contains(routeDef
nition.getId()))
                .forEach(routeDefinition -> routeDefinition.getPredicates().stream()
                .filter(predicateDefinition -> "Path".equalsIgnoreCase(predicateDefinition.getName(
))
                .forEach(predicateDefinition -> resourceList.add(
                        swaggerResource(
                            routeDefinition.getId(),
                            predicateDefinition
                                    .getArgs()
                                    .get(NameUtils.GENERATED_NAME_PREFIX + "0")
                                    .replace("/**",SWAGGER2URL)
//这里拼接时需要注意
//网关配置account映射到account-service，要么将网关的配置修改成account-service映射成accou
t-service
//要么就在这个拼接处解决
                        )
                )));
        return resourceList;
    }

    private SwaggerResource swaggerResource(String name, String location) {
        SwaggerResource swaggerResource = new SwaggerResource();
        swaggerResource.setName(name);
        swaggerResource.setLocation(location);
        swaggerResource.setSwaggerVersion("2.0");
        return swaggerResource;
    }
}
```

- 自定义Rest接口

```
@RestController
@RequestMapping("/swagger-resources")
public class SwaggerHandler {
    @Autowired(required = false)
    private SecurityConfiguration securityConfiguration;

    @Autowired(required = false)
    private UiConfiguration uiConfiguration;
```

```
    private final SwaggerResourcesProvider swaggerResources;

    @Autowired
    public SwaggerHandler(SwaggerResourcesProvider swaggerResources) {
        this.swaggerResources = swaggerResources;
    }

    @GetMapping("/configuration/security")
    public Mono<ResponseEntity<SecurityConfiguration>> securityConfiguration() {
        return Mono.just(new ResponseEntity<>(
                Optional.ofNullable(securityConfiguration).orElse(SecurityConfigurationBuilder.buil
er().build()), HttpStatus.OK));
    }

    @GetMapping("/configuration/ui")
    public Mono<ResponseEntity<UiConfiguration>> uiConfiguration() {
        return Mono.just(new ResponseEntity<>(
                Optional.ofNullable(uiConfiguration).orElse(UiConfigurationBuilder.builder().build())
HttpStatus.OK));
    }

    @GetMapping("")
    public Mono<ResponseEntity> swaggerResources() {
        return Mono.just((new ResponseEntity<>(swaggerResources.get(), HttpStatus.OK)));
    }
}
```

系列文章

- SpringCloud Alibaba微服务实战十 - 服务网关
- SpringCloud Alibaba微服务实战九 - Seata 容器化
- SpringCloud Alibaba微服务实战八 - Seata 整合Nacos
- SpringCloud Alibaba微服务实战七 - 分布式事务
- SpringCloud Alibaba微服务实战六 - 配置隔离
- SpringCloud Alibaba微服务实战五 - 限流熔断
- SpringCloud Alibaba微服务实战四 - 版本管理
- SpringCloud Alibaba微服务实战三 - 服务调用
- SpringCloud Alibaba微服务实战二 - 服务注册
- SpringCloud Alibaba微服务实战一 - 基础环境准备