

# Spring Cloud Gateway集成SpringDoc, 集中管理微服务API

叹雪飞花

2023-11-29

 768

 阅读5分钟

关注

## 本文目标

Spring Cloud微服务集成SpringDoc, 在Spring Cloud Gateway中统一管理微服务的API, 微服务上下线时自动刷新SwaggerUi中的group组。

## 依赖版本

框架	版本
Spring Boot	3.1.5
Spring Cloud	2022.0.4
Spring Cloud Alibaba	2022.0.0.0
Spring Doc	2.2.0
Nacos Server	2.2.3

## 开始集成

### 项目模块

```
> spring-doc-cloud-gateway
> spring-doc-cloud-webflux
> spring-doc-cloud-webmvc
> target
.gitignore
pom.xml
```

公共模块里的配置是之前文章中提到的内容，加了一个webmvc和webflux的适配，我会将文章和代码仓库的链接放在最下边，有需要的可以去看看。

## 引入依赖，配置依赖管理

在父模块中添加lombok、测试包和服务发现与注册的包，管理Spring Cloud、Spring Cloud Alibaba依赖版本，如下

### 不要忘了SpringDoc的依赖管理

xml 复制代码

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-i
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/mave
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>3.1.5</version>
9         <relativePath/> <!-- Lookup parent from repository -->
10    </parent>
11    <groupId>com.example</groupId>
12    <artifactId>spring-doc-spring-cloud</artifactId>
13    <version>0.0.1</version>
14    <packaging>pom</packaging>
15    <name>spring-doc-spring-cloud</name>
16    <description>spring-doc-spring-cloud</description>
17    <modules>
18        <module>spring-doc-cloud-common</module>
19        <module>spring-doc-cloud-gateway</module>
20        <module>spring-doc-cloud-webflux</module>
21        <module>spring-doc-cloud-webmvc</module>
22    </modules>
23    <properties>
24        <!-- 指定Java版本为Java17 -->
```



```
28     <!-- 修复SpringBoot自带snakeyaml依赖版本的漏洞 -->
29     <snakeyaml.version>2.0</snakeyaml.version>
30     <!-- SpringDoc-OpenApi版本号 -->
31     <spring-doc.version>2.2.0</spring-doc.version>
32     <!-- SpringCloud版本 -->
33     <spring-cloud.version>2022.0.4</spring-cloud.version>
34     <!-- 指定打包插件版本 -->
35     <maven-surefire-plugin.version>3.2.2</maven-surefire-plugin.version>
36     <!-- Spring Cloud Alibaba版本号 -->
37     <spring-cloud-alibaba.version>2022.0.0.0</spring-cloud-alibaba.version>
38 </properties>
39 <dependencies>
40     <dependency>
41         <groupId>org.projectlombok</groupId>
42         <artifactId>lombok</artifactId>
43         <optional>true</optional>
44     </dependency>
45     <dependency>
46         <groupId>org.springframework.boot</groupId>
47         <artifactId>spring-boot-starter-test</artifactId>
48         <scope>test</scope>
49     </dependency>
50     <!-- 服务注册与发现 -->
51     <dependency>
52         <groupId>com.alibaba.cloud</groupId>
53         <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
54     </dependency>
55 </dependencies>
56 <dependencyManagement>
57     <dependencies>
58         <dependency>
59             <groupId>org.springframework.cloud</groupId>
60             <artifactId>spring-cloud-dependencies</artifactId>
61             <version>${spring-cloud.version}</version>
62             <type>pom</type>
63             <scope>import</scope>
64         </dependency>
65
66         <dependency>
67             <groupId>com.alibaba.cloud</groupId>
68             <artifactId>spring-cloud-alibaba-dependencies</artifactId>
69             <version>${spring-cloud-alibaba.version}</version>
70             <type>pom</type>
71             <scope>import</scope>
72         </dependency>
73
```

```
77         <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
78         <version>${spring-doc.version}</version>
79     </dependency>
80
81     <!-- 适用于webflux的SpringDoc依赖 -->
82     <dependency>
83         <groupId>org.springdoc</groupId>
84         <artifactId>springdoc-openapi-starter-webflux-ui</artifactId>
85         <version>${spring-doc.version}</version>
86     </dependency>
87 </dependencies>
88 </dependencyManagement>
89
90 </project>
91
```

## spring-doc-cloud-gateway 模块说明

引入webflux、gateway、loadbalancer负载均衡和springdoc依赖，同时引入公共模块



xml 复制代码

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven
5      <modelVersion>4.0.0</modelVersion>
6      <parent>
7          <groupId>com.example</groupId>
8          <artifactId>spring-doc-spring-cloud</artifactId>
9          <version>0.0.1</version>
10     </parent>
11
12     <artifactId>spring-doc-cloud-gateway</artifactId>
13
14     <properties>
15         <maven.compiler.source>17</maven.compiler.source>
16         <maven.compiler.target>17</maven.compiler.target>
17         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18     </properties>
19
20     <dependencies>
```



```
24         <artifactId>spring-boot-starter-webflux</artifactId>
25     </dependency>
26     <!-- 网关 -->
27     <dependency>
28         <groupId>org.springframework.cloud</groupId>
29         <artifactId>spring-cloud-starter-gateway</artifactId>
30     </dependency>
31     <!-- 负载均衡 -->
32     <dependency>
33         <groupId>org.springframework.cloud</groupId>
34         <artifactId>spring-cloud-starter-loadbalancer</artifactId>
35     </dependency>
36
37     <!-- SpringDoc -->
38     <dependency>
39         <groupId>org.springdoc</groupId>
40         <artifactId>springdoc-openapi-starter-webflux-ui</artifactId>
41     </dependency>
42
43     <dependency>
44         <groupId>io.projectreactor</groupId>
45         <artifactId>reactor-test</artifactId>
46         <scope>test</scope>
47     </dependency>
48
49     <!-- 公共包，这里是对于swagger的自定义配置，可以参考之前的文章或直接查看代码仓库的实现 -->
50     <dependency>
51         <groupId>com.example</groupId>
52         <artifactId>spring-doc-cloud-common</artifactId>
53         <version>${common.version}</version>
54     </dependency>
55 </dependencies>
56
57 <build>
58     <plugins>
59         <plugin>
60             <groupId>org.graalvm.buildtools</groupId>
61             <artifactId>native-maven-plugin</artifactId>
62         </plugin>
63         <plugin>
64             <groupId>org.springframework.boot</groupId>
65             <artifactId>spring-boot-maven-plugin</artifactId>
66             <configuration>
67                 <image>
68                     <builder>paketobuildpacks/builder-jammy-tiny:latest</builder>
69                 </image>
```



```
73         <artifactId>lombok</artifactId>
74     </exclude>
75 </excludes>
76 </configuration>
77 </plugin>
78 </plugins>
79 </build>
80
81 </project>
```

## 修改 application.yml

开启gateway自动扫描, 根据注册中心的服务自动生成路由, 路由名转小写, 添加自定义的swagger配置。



yml 复制代码

```
1  spring:
2    application:
3      name: gateway
4    cloud:
5      gateway:
6        discovery:
7          locator:
8            # 根据注册中心的服务自动生成路由
9            enabled: true
10           # 路由名转小写
11           lower-case-service-id: true
12
13 # -----以下内容可改为公共配置-----
14 # SpringDoc自定义配置
15 custom:
16   info:
17     title: ${spring.application.name}-api
18     version: 0.0.1
19     description: 这是一个使用SpringDoc生成的在线文档.
20     terms-of-service: http://127.0.0.1:8000/test01
21     gateway-url: http://127.0.0.1:8080
22   license:
23     name: Apache 2.0
24   security:
25     name: Authenticate
26     token-url: http://kwqqr48rgo.cdhttp.cn/oauth2/token
```

## 添加 InstancesChangeListener

监听微服务启、停状态，微服务状态改变后刷新Swagger UI中的组。

▼

java 复制代码

```
1 package com.example.config;
2
3 import com.alibaba.nacos.client.naming.event.InstancesChangeEvent;
4 import com.alibaba.nacos.common.notify.Event;
5 import com.alibaba.nacos.common.notify.NotifyCenter;
6 import com.alibaba.nacos.common.notify.listener.Subscriber;
7 import com.alibaba.nacos.common.utils.JacksonUtils;
8 import jakarta.annotation.PostConstruct;
9 import jakarta.annotation.Resource;
10 import lombok.extern.slf4j.Slf4j;
11 import org.springdoc.core.properties.AbstractSwaggerUiConfigProperties;
12 import org.springdoc.core.properties.SwaggerUiConfigProperties;
13 import org.springframework.cache.Cache;
14 import org.springframework.cache.CacheManager;
15 import org.springframework.cloud.gateway.route.RouteDefinition;
16 import org.springframework.cloud.gateway.route.RouteDefinitionLocator;
17 import org.springframework.context.annotation.Configuration;
18 import org.springframework.util.ObjectUtils;
19
20 import java.util.List;
21 import java.util.Set;
22 import java.util.stream.Collectors;
23
24 import static org.springdoc.core.utils.Constants.DEFAULT_API_DOCS_URL;
25 import static org.springframework.cloud.loadbalancer.core.CachingServiceInstanceListSupplier.SER
26
27 /**
28  * 监听注册中心实例注册状态改变事件，微服务实例状态改变后刷新swagger ui的组(一个组等于一个微服务)
29  *
30  * @author vains
31  */
32 @Slf4j
33 @Configuration(proxyBeanMethods = false)
34 public class InstancesChangeListener extends Subscriber<InstancesChangeEvent> {
35
36     private final String LB_SCHEME = "lb";
37
38     private final RouteDefinitionLocator locator;
```

```
42
43     private final SwaggerUiConfigProperties swaggerUiConfigProperties;
44
45     /**
46      * 获取配置文件中默认配置的swagger组
47      */
48     private final Set<AbstractSwaggerUiConfigProperties.SwaggerUrl> defaultUrls;
49
50
51     public InstancesChangeListener(RouteDefinitionLocator locator,
52                                   SwaggerUiConfigProperties swaggerUiConfigProperties) {
53         this.locator = locator;
54         this.swaggerUiConfigProperties = swaggerUiConfigProperties;
55         // 构造器中初始化配置文件中的swagger组
56         this.defaultUrls = swaggerUiConfigProperties.getUrls();
57     }
58
59     @Override
60     public void onEvent(InstancesChangeEvent event) {
61         if (log.isDebugEnabled()) {
62             log.info("Spring Gateway 接收实例刷新事件: {}, 开始刷新缓存", JacksonUtils.toJson(event));
63         }
64         Cache cache = defaultLoadBalancerCacheManager.getCache(SERVICE_INSTANCE_CACHE_NAME);
65         if (cache != null) {
66             cache.evict(event.getServiceName());
67         }
68         // 刷新group
69         this.refreshGroup();
70
71         if (log.isDebugEnabled()) {
72             log.info("Spring Gateway 实例刷新完成");
73         }
74     }
75
76     /**
77      * 刷新swagger的group
78      */
79     public void refreshGroup() {
80         // 获取网关路由
81         List<RouteDefinition> definitions = locator.getRouteDefinitions().collectList().block();
82         if (ObjectUtils.isEmpty(definitions)) {
83             return;
84         }
85
86         // 根据路由规则生成 swagger组 配置
87         Set<AbstractSwaggerUiConfigProperties.SwaggerUrl> swaggerUrls = definitions.stream()
```

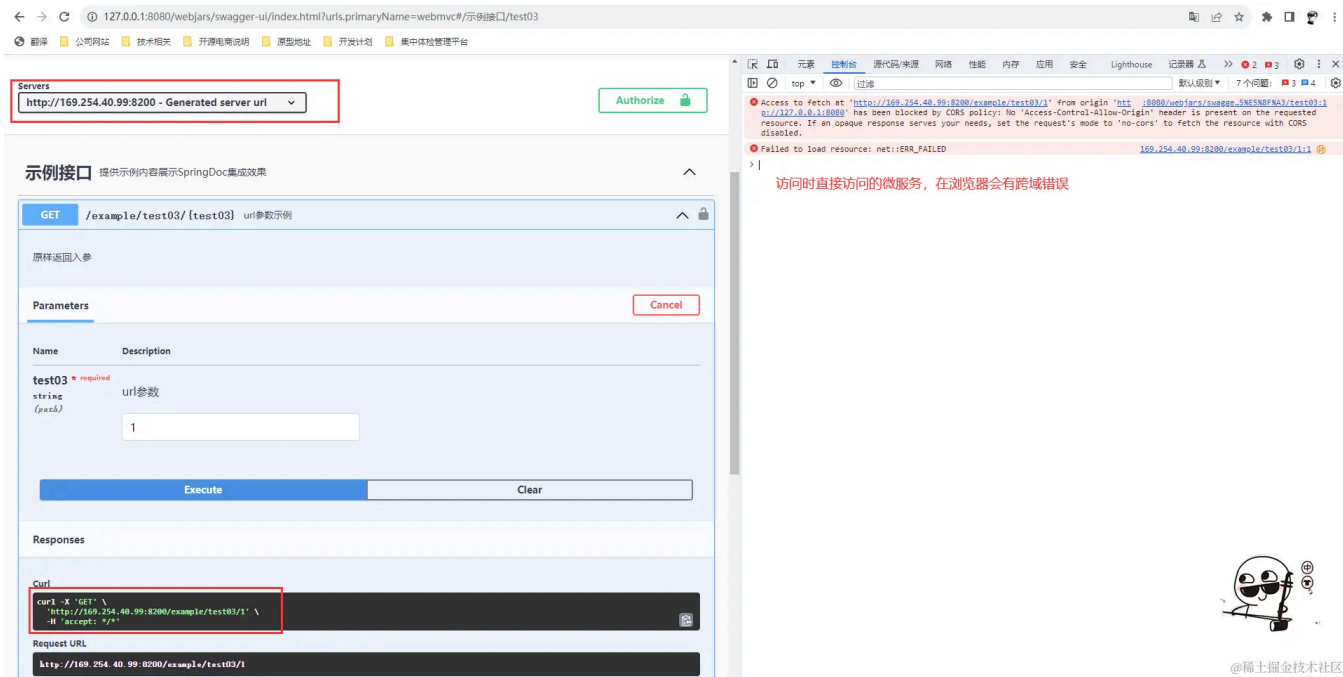
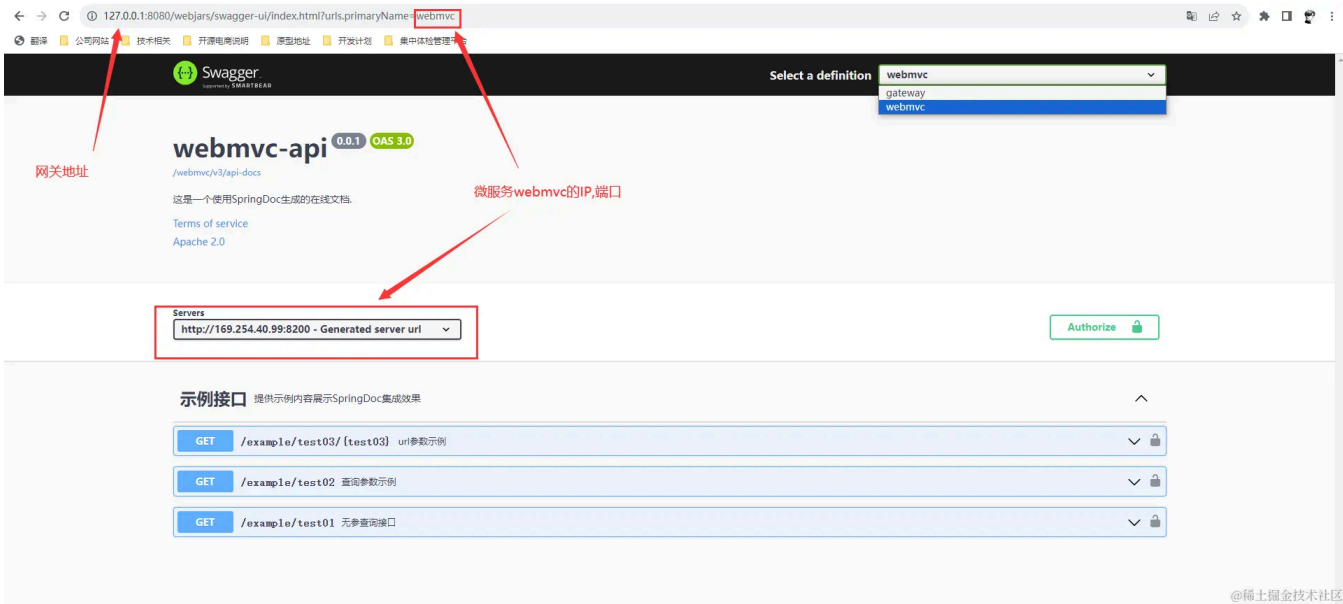


```
91         // 生成 swagger组 配置, 以微服务在注册中心中的名字当做组名、请求路径(我这里使用的是
92         String authority = definition.getUri().getAuthority();
93         return new AbstractSwaggerUiConfigProperties.SwaggerUrl(authority, authority
94     })
95     .collect(Collectors.toSet());
96
97     // 如果在配置文件中添加其它 swagger组 配置则将两者合并
98     if (!ObjectUtils.isEmpty(defaultUrls)) {
99         swaggerUrls.addAll(defaultUrls);
100    }
101
102    // 重置配置文件
103    swaggerUiConfigProperties.setUrls(swaggerUrls);
104
105    if (log.isDebugEnabled()) {
106        String groups = swaggerUrls.stream()
107            .map(AbstractSwaggerUiConfigProperties.SwaggerUrl::getName)
108            .collect(Collectors.joining(","));
109        log.debug("刷新Spring Gateway Doc Group成功, 获取到组: {}. ", groups);
110    }
111 }
112
113 @PostConstruct
114 public void registerToNotifyCenter() {
115     // 注册监听事件
116     NotifyCenter.registerSubscriber((this));
117 }
118
119 @Override
120 public Class<? extends Event> subscribeType() {
121     return InstancesChangeEvent.class;
122 }
123 }
```

网关启动时、微服务停止、微服务启动时网关会从注册中心获取最新的服务列表, 然后根据服务列表生成路由配置, 路由的代理路径就是微服务的名字, 使用 **http://网关ip:网关端口/微服务名/\*\*** 访问对应的微服务。

在注册中心(Nacos)的服务列表更新时会会有一个SpringEvent事件通知, 也就是上边类中的监听实现, 每次收到通知时就会根据网关的路由生成 **SwaggerUrl** 列表, 其中name是微服务的名字(application.name), 路径是 **/application.name/v3/api-docs**, 这样实际上就是通过网关将请求代理至各微服务了, 获取到的api信息实际上也是各微服务的, 如果某个微服务禁用

当然，虽然可以通过网关代理获取到微服务的api信息，但是在测试接口时还是会出现问题，请求会直接发送至微服务，并不会经过网关代理，如下所示



所以说需要修改各微服务配置，指定当前服务访问的url，在SpringDoc配置中添加 **servers** 属性，并设置值为被网关代理的路径，如下所示



```
17  * Spring Doc OpenApi 注解配置
18  *
19  * @author vains
20  */
21  @vains-Sofia
22  @OpenAPIDefinition(
23      info = @Info(
24          // 标题
25          title = "${custom.info.title}",
26          // 版本
27          version = "${custom.info.version}",
28          // 描述
29          description = "${custom.info.description}",
30          // 首页
31          termsOfService = "${custom.info.termsOfService}",
32          // license
33          license = @License(
34              name = "${custom.license.name}",
35              // license 地址
36              url = "http://127.0.0.1:8080/example/test01"
37          ),
38          // 这里的名字是引用下边 @SecurityScheme 注解中指定的名字, 指定后发起请求时会在请求头中按照OAuth2的规范添加token
39          security = @SecurityRequirement(name = "${custom.security.name}"),
40          servers = @Server(url = "${custom.info.gateway-url}")
41      )
42      @SecuritySchemes({@SecurityScheme(
```

这里是网关代理的路径  
http://网关ip:网关端口/当前服务名

@稀土掘金技术社区

在引用的微服务中设置自定义配置 `custom.info.gateway-url` , 相信看到这里就明白为什么上方网关的yml中会有这么一个配置了。

## 修改微服务yml

添加类似如下配置, 设置 `spring.application.name` , 设置SpringDoc自定义配置, 设置 `custom.info.gateway-url`



yml 复制代码

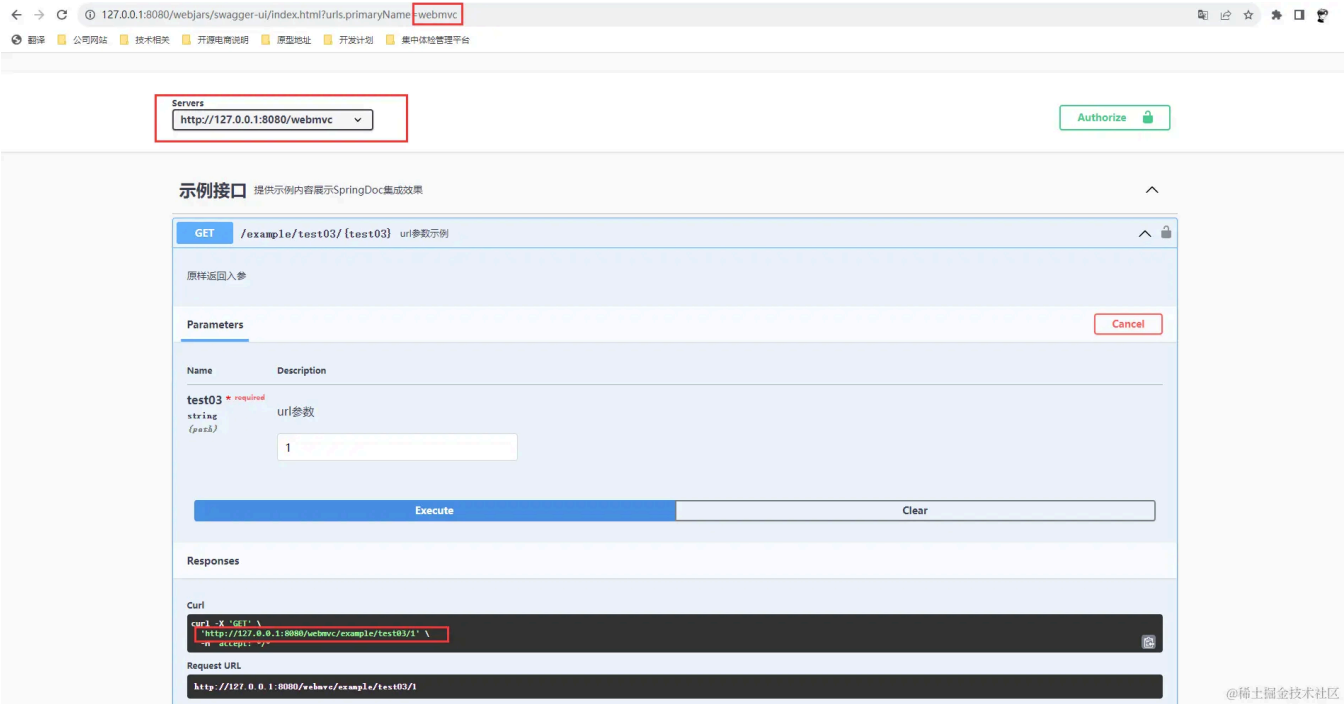
```
1  spring:
2    application:
3      name: webmvc
4
5  # -----以下内容可改为公共配置-----
6  # SpringDoc自定义配置
7  custom:
8    info:
9      title: ${spring.application.name}-api
```

```
13 # 设置当前服务在网关中的代理路径
14 gateway-url: http://127.0.0.1:8080/${spring.application.name}
15 license:
16   name: Apache 2.0
17 security:
18   name: Authenticate
19   token-url: http://kwqqr48rgo.cdhttp.cn/oauth2/token
20   authorization-url: http://kwqqr48rgo.cdhttp.cn/oauth2/authorize
21 server:
22   port: 8200
```

## 查看效果

webflux访问地址, 默认会有一个 webjars 前缀

<http://127.0.0.1:8080/webjars/swagger-ui/index.html>



其它微服务都是一些测试接口, 没必要贴了, 大家用自己的就好, 或者去代码仓库拉取代码看看。

## 附录



3. 代码仓库：[Gitee](#)、[Github](#)

标签： Spring Boot    Spring Cloud    Java    话题： 每天一个知识点

本文收录于以下专栏



SpringDoc 专栏目录  
适用于SpringBoot3.x版本的在线文档工具——SpringDoc框架。  
14 订阅 · 4 篇文章

订阅

上一篇 SpringDoc枚举字段处理与SpringBoot接...

评论 0



抢首评，友善交流



0 / 1000 ? 发送

暂无评论数据

开始集成

- 项目模块
- 引入依赖, 配置依赖管理
- spring-doc-cloud-gateway模块说明
  - 修改application.yml
  - 添加InstancesChangeListener
  - 修改微服务yml
  - 查看效果

附录

相关推荐

Spring Authorization Server + Oauth2 配置认证服务器与资源服务器

2.9k阅读 · 11点赞

 Spring Authorization Server (1) 认证、授权、oauth2概念和流程初步介绍

474阅读 · 3点赞

Spring Authorization Server的使用

7.2k阅读 · 21点赞

 Spring Authorization Server (2) 授权服务、资源服务、客户端核心配置讲解

557阅读 · 3点赞

 Spring Authorization Server (10) 授权服务的JWK密钥对生成和JWT信息扩展

368阅读 · 1点赞

精选内容

Java多线程的常用方法和使用

薛慕昭 · 228阅读 · 1点赞

聊聊基于传统 Client 、Server 架构的企业级软件中的消息显示机制的实现

JerryWang\_sap · 202阅读 · 0点赞

Go 中的空结构体和空字符串

胡译胡说 · 236阅读 · 2点赞

SpringBoot项目重构思路

花哥编程 · 212阅读 · 0点赞

为你推荐

- Spring Authorization Server入门 (二) Spring Boot整合Spring Authorization Server

叹雪飞花 9月前 👁 6.8k 👍 31 💬 86 Java
- Spring Authorization Server入门 (十二) 实现授权码模式使用前后端分离的登录页面

叹雪飞花 8月前 👁 4.8k 👍 24 💬 65 后端 Spring ... Spring
- Spring Authorization Server入门 (十) 添加短信验证码方式登录

叹雪飞花 9月前 👁 3.4k 👍 20 💬 9 Spring Spring ...
- Spring Authorization Server入门 (八) Spring Boot引入Security OAuth2 Client对接认...

叹雪飞花 9月前 👁 2.8k 👍 13 💬 43 Spring ... Spring
- Spring Authorization Server入门 (十六) Spring Cloud Gateway对接认证服务

叹雪飞花 7月前 👁 2.6k 👍 17 💬 44 Spring ... Spring ... 安全
- Spring Authorization Server入门 (十三) 实现联合身份认证, 集成Github与Gitee的OAu...

叹雪飞花 8月前 👁 2.3k 👍 13 💬 51 Spring Spring ... 安全
- Spring Authorization Server入门 (十一) 自定义grant\_type(短信认证登录)获取token

叹雪飞花 9月前 👁 2.5k 👍 15 💬 39 Spring Spring ... 安全
- Spring Authorization Server入门 (七) 登录添加图形验证码

叹雪飞花 9月前 👁 2.9k 👍 18 💬 4 Spring ...
- SpringBoot3.x最简集成SpringDoc-OpenApi

叹雪飞花 4月前 👁 2.3k 👍 17 💬 评论 后端 Spring ... Java
- Spring Authorization Server入门 (九) Spring Boot引入Resource Server对接认证服务

叹雪飞花 9月前 👁 1.9k 👍 13 💬 8 Spring Spring ...
- Spring Authorization Server优化篇: 添加Redis缓存支持和统一响应类

Spring Authorization Server入门 (十五) 令牌授权确认与设备验证页面

叹雪飞花7月前

 1.6k

 14

 8

Spring ...


Spring

Vue.js

Spring Authorization Server入门 (十九) 基于Redis的Token、客户端信息和授权确认信...

叹雪飞花4月前

 1.3k

 8

 19

Spring ...

后端

Redis

Spring Authorization Server入门 (二十) 实现二维码扫码登录

叹雪飞花2月前

 937

 16

 6

Spring ...

Spring

Java

Spring Authorization Server入门 (十七) Vue项目使用授权码模式对接认证服务

叹雪飞花6月前

 762

 9

 12

Vue.js

安全

Spring ...