



土味儿~ 已于 2022-08-18 10:16:37 修改 阅读量3k 收藏 17 点赞数

分类专栏：

# SpringSecurity OAuth2

# SpringCloud

# SpringBoot

 文章标签：

eureka

spring

gateway

SpringBoot 同时被 3 个专栏收录

20 订阅

## 目概述

### 概述

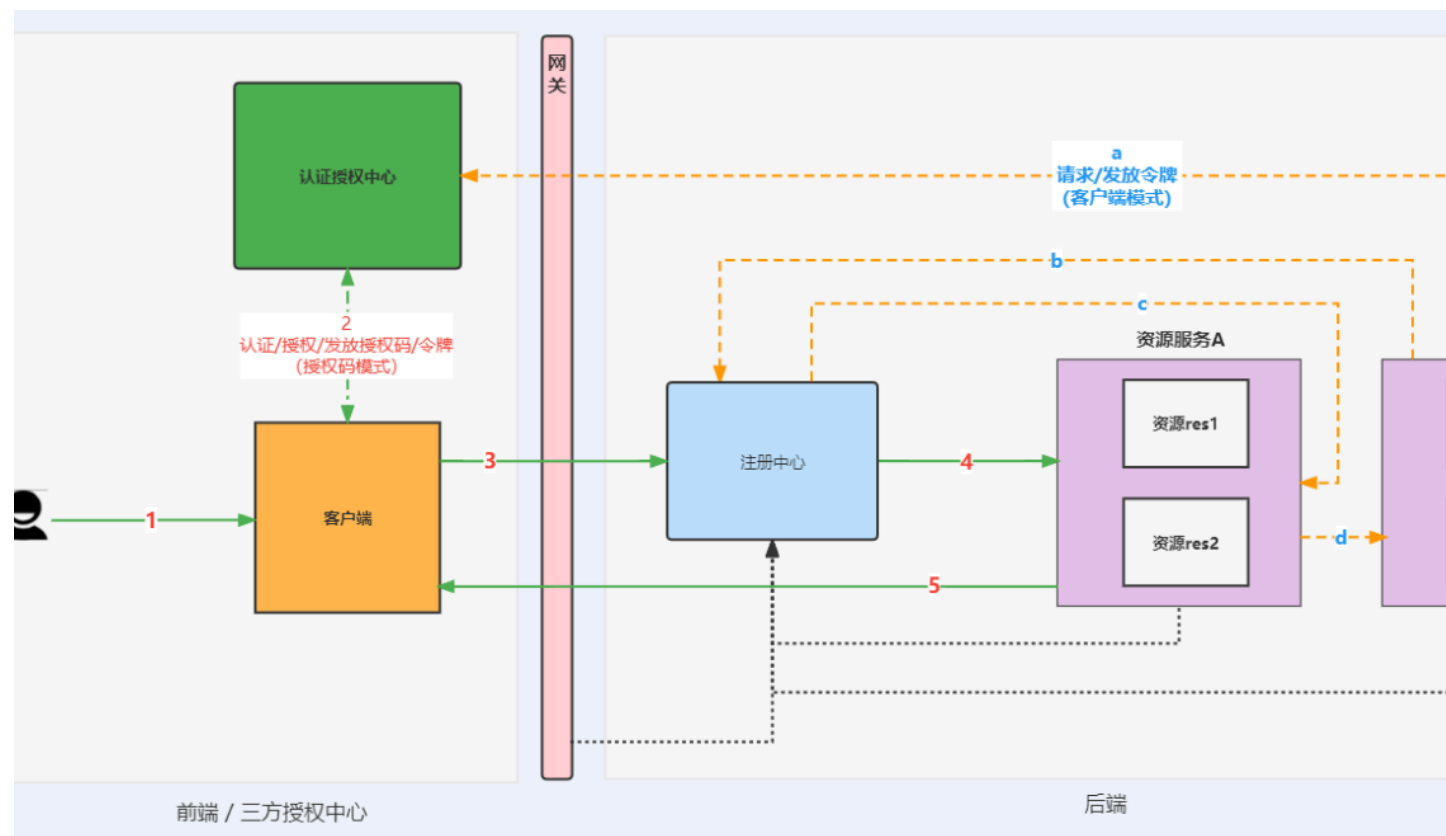
是在前面章节的基础上，进行的升级改造。增加了 注册中心 、网关，更加贴近于实际需求。

本节之前，请先搭建前面项目：

【图文详解】搭建 Spring Authorization Server + Resource + Client 完整Demo

【oauth2 客户端模式】Spring Authorization Server + Resource + Client 资源服务间的相互访问

### 整体架构图



### 流程解析

#### 总体结构

分为三个部分：

端和第三方授权中心

关：把前后两部分隔离开，所有的请求访问都通过网关转发

端：所有的资源服务（微服务）、注册中心、配置中心等

**1端** 需要用户认证/授权，并请求 **授权中心** 发放令牌

**1端** 拿到令牌后，通过 **网关**，去 **注册中心** 查找 **资源服务**

**1中心** 把资源服务告之 **客户端**，**客户端** 携带令牌去访问 **资源服务**

**1服务** 验证令牌后，把资源数据，通过 **网关**，返回 **客户端**，再呈现给 **用户**

资源服务间调用

**1服务**之间的访问也需要经过网关（图中略），减少服务间的耦合

**1服务B** 希望调用 **资源服务A**；**资源服务B** 通过 **网关**，向 **授权中心** 申请令牌；**授权中心** 验证ID/密钥(客户端模式)后，通过 **网关**，直接发放令牌给 **资源服**

**1服务B** 通过 **网关**，向 **注册中心** 查找 **资源服务A**

**1中心** 把 **资源服务A**，再通过 **网关**，告之 **资源服务B**，**资源服务B** 携带令牌访问 **资源服务A**

**1服务A** 验证令牌后，把资源数据，通过 **网关**，返给 **资源服务B**

额外说明

**有资源服务** 都需要注册到 **注册中心**，访问者只需要知道 **网关地址** 和 **服务名称**，就可以访问；不用关心资源服务的具体部署位置，不管是单机部署还是集

**关** 是前后端访问的桥梁，网关也可以注册到注册中心；所有的访问都应该经过网关转发，减少服务间的耦合

**1中心** 作为第三方存在，不应该注册到注册中心

**1端** 作为工程的前端，也不用注册到注册中心

搭建环境

ring Security 5.6.3 (Client/Resource)

ring Authorization Server 0.2.3

ring Boot 2.6.7 (gateway、eureka...)

. 1.8

sql 5.7

nbok、log4j、fastjson2 ...

构搭建

模块	端口	说明
oauth2-server-resource-client-gateway-eureka	—	父工程
oauth2-client-8000	8000	项目前端（oauth2客户端）
oauth2-server-9000	9000	认证授权中心（oauth2服务端）
oauth2-resource-a-8001	8001	微服务A（oauth2资源服务器），受保护对
oauth2-resource-b-8002	8002	微服务B（oauth2资源服务器），受保护对
eureka-7000	7000	注册中心
gateway-9999	9999	网关

父工程

通 **meven** 工程 **oauth2-server-resource-client-gateway-eureka**；打包格式 **pom**，删除 **src**

**m.xmll**

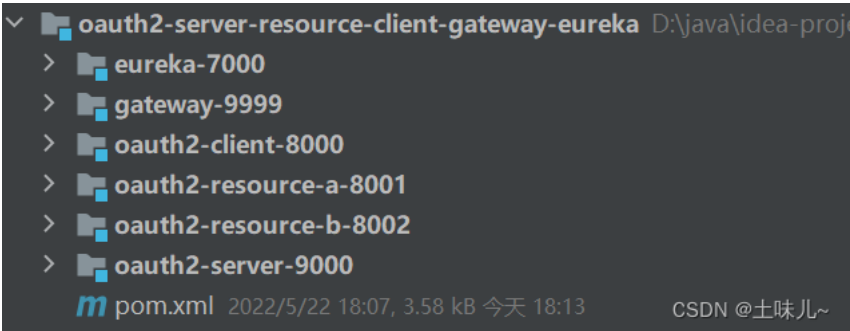
土味儿~

已关注

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.tuwer</groupId>
  <artifactId>oauth2-server-resource-client-gateway-eureka</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>pom</packaging>
```

子模块



注册中心

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <parent>
    <artifactId>oauth2-server-resource-client-gateway-eureka</artifactId>
    <groupId>com.tuwer</groupId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
```

application.yml

```
server:
  port: 7000

spring:
  application:
    # 注册中心
    name: eureka-server-7000

# Eureka配置
eureka:
  instance:
```

土味儿~

已关注

```
package com.tuwer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

/**
 * @author 土味儿
 * Date 2022/5/21
 * @version 1.0
 */
```

## 关

### om.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>oauth2-server-resource-client-gateway-eureka</artifactId>
        <groupId>com.tuwer</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
```

### pplication.yml

```
server:
  port: 9999

spring:
  application:
    # 网关应用名称
    name: gateway-9999
  # 配置 Spring Cloud 相关属性
  cloud:
    # 配置 Spring Cloud Gateway 相关属性
    gateway:
```

## 自动类

```
package com.tuwer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

/**
 * @author 土味儿
 * Date 2022/5/21
 * @version 1.0
 */
@SpringBootApplication
```

土味儿~

已关注

## MyInfo.java

一步可以省略；作用：给外部一个查看该模块基本信息的接口；

需要导入 `actuator` 依赖

application.yml 中配置

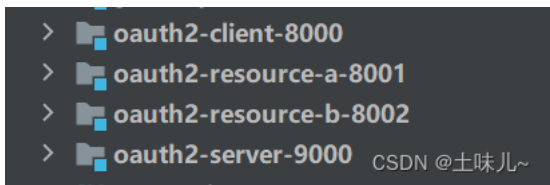
如果有security安全设置，需要放开口 `/actuator/info`

```
@Component
public class MyInfo implements InfoContributor {
    @Override
    public void contribute(Info.Builder builder) {
        HashMap<String, Object> map = new HashMap<>();
        // 可以从数据库获取信息
        map.put("ServiceName", "路由网关");
        map.put("version", "1.0-SNAPSHOT");
        map.put("author", "tuwer");
        builder.withDetails(map);
    }
}
```

```
# 监控端口配置
management:
  endpoints:
    web:
      exposure:
        # 开启 info,health; 新版本中只默认开启了 health
        include: info,health
```

## 合 OAuth2 三元素

### 复制旧项目中OAuth2三元素模块



### 记置资源服务到注册中心等

`oauth2-resource-a-8001` 为例，其它资源模块操作方法一样；

授权中心不需要添加到注册中心，它作为第三方存在

客户端作为工程前端，也不用添加到注册中心

### 添加依赖

```
<!-- 注册中心客户端 -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

### 配置注册中心

```
# 配置 Eureka
eureka:
  client:
    # 默认值: true 需要从注册中心拉取其他的服
```





```
# 注册中心地址
defaultZone: http://localhost:7000/eureka/
```

```
instance:
# 修改Eureka上的默认描述信息
instance-id: oauth2-resource-a-8001
# 以IP地址注册到服务中心
prefer-ip-address: true
```

## 启动类添加 @EnableEurekaClient

```
@SpringBootApplication
@EnableEurekaClient
public class Resource_a_8001 {
    public static void main(String[] args) {
        SpringApplication.run(Resource_a_8001.class, args);
    }
}
```

## 配置MyInfo.java

一步可以省略；作用：给外部一个查看该模块基本信息的接口；

## 入依赖

```
<!-- actuator完善监控信息 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

## Info.java

```
@Component
public class MyInfo implements InfoContributor {
    @Override
    public void contribute(Info.Builder builder) {
        HashMap<String, Object> map = new HashMap<>();
        // 可以从数据库获取信息
        map.put("ServiceName", "资源服务器A");
        map.put("version", "1.0-SNAPSHOT");
        map.put("author", "tuwer");
        builder.withDetails(map);
    }
}
```

## application.yml

```
# 监控端口配置
management:
  endpoints:
    web:
      exposure:
        # 开启 info,health; 新版本中只默认开启了 health
        include: info,health
```

## 全策略中放行端点

```
// 第一种方法
@Bean
SecurityFilterChain oauth2SecurityFilterChain(HttpSecurity http) throws Exception {
    http.authorizeRequests(requests ->
```



已关注



```
// 让所有请求都通过认证
requests
    .antMatchers("/actuator/**").permitAll()
    .anyRequest().authenticated()
)

// ...
return http.build();
}

// 第二种方法
@Bean
WebSecurityCustomizer webSecurityCustomizer() {
    return web -> web.ignoring().antMatchers("/actuator/health", "/actuator/info");
}
```

## 配置网关中路由规则

`spring.cloud.gateway.routes` 节点下配置

### 原服务A

```
# 资源服务A
# 把对 【网关/o2_resource_a/**】 的请求，转发到 【lb://oauth2-resource-a-8001】
# 转发时去掉第1节 o2_resource_a
# lb: 表示负载均衡 Loadbalance
# oauth2-resource-a-8001 是【资源服务A】在注册中心中的名称
- id: oauth2-resource-a-8001
  uri: lb://oauth2-resource-a-8001
  predicates:
    - Path=/o2_resource_a/**
    - Method=GET
  filters:
    - name: StripPrefix
      args:
        # 过滤掉第1节
        parts: 1
```

### 原服务B

```
# 资源服务B
- id: oauth2-resource-b-8002
  uri: lb://oauth2-resource-b-8002
  predicates:
    - Path=/o2_resource_b/**
    - Method=GET
  filters:
    - name: StripPrefix
      args:
        # 过滤掉第1节
        parts: 1
```

### 客户端

```
# 客户端
- id: oauth2-client-8000
  uri: http://127.0.0.1:8000
  predicates:
    - Path=/o2_client/**
    - Method=GET
  filters:
    - name: StripPrefix
      args:
```



土味儿~ 已关注





## 认证中心

```
# 认证中心
- id: oauth2-server-9000
  uri: http://os.com:9000
  predicates:
    - Path=/o2_server/**
    - Method=GET,POST
  filters:
    - name: StripPrefix
      args:
        # 过滤掉第1节
        parts: 1
```

## 旧项目部分内容修改

### 授权中心

改注册的两个客户端ID和名称；为了与旧项目区分。修改完之后，在客户端引用位置也要相应修改

```
//String clientId_1 = "my_client";
String clientId_1 = "my_client_2";

//String clientId_2 = "micro_service";
String clientId_2 = "micro_service_2";

// 客户端名称: 可省略
//.clientId("my_client_name")
.clientId("my_client_name_2")

//.clientId("micro_service")
.clientId("micro_service_2")
```

### 客户端

application.yml

```
#client-id: my_client
client-id: my_client_2
```

sourceController.java

把对资源的直接调用，改为通过网关调用

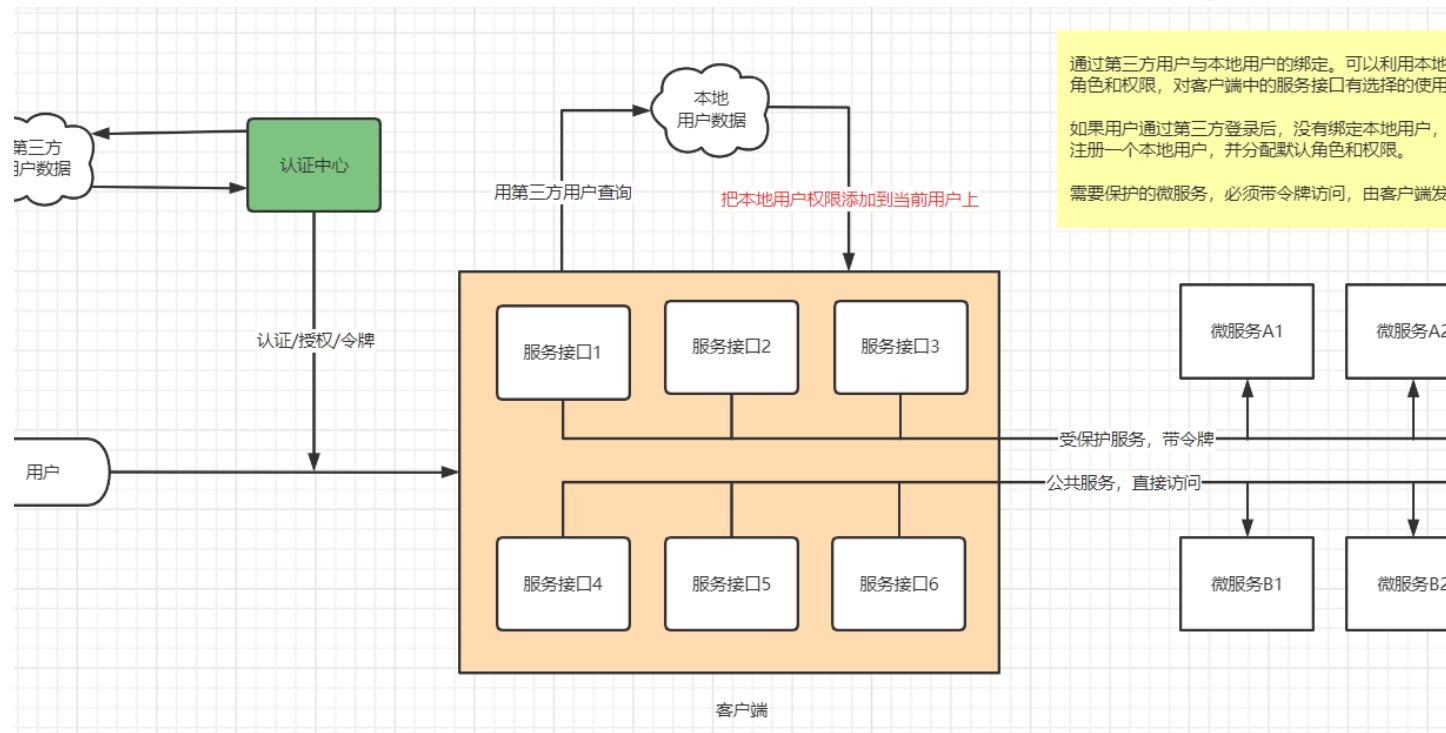
```
// ...
// 网关地址
private String BASE_URL = "http://192.168.62.1:9999";

@GetMapping("/server/a/res1")
public String getServerAres1(@RegisteredOAuth2AuthorizedClient
    OAuth2AuthorizedClient oAuth2AuthorizedClient) {
    // 网关地址/路由路径...
    return getServer(BASE_URL + "/o2_resource_a/res1", oAuth2AuthorizedClient);
}
// ...
```



土味儿~

已关注



也模拟客户

```
/**
 * 虚拟一个本地用户
 *
 * @return UserDetailsService
 */
@Bean
UserDetailsService userDetailsService() {
    return username -> User.withUsername("local_admin")
        .password("123456")
        .roles("TEST", "ABC")
        //.authorities("ROLE_ADMIN", "ROLE_USER")
        .build();
}
```

也权限提升

```
@Controller
public class IndexController {
    @Autowired
    UserDetailsService userDetailsService;

    /**
     * 权限提升
     * 第三方用户进入本系统后，绑定本地用户，获取本地用户的角色和权限
     * @param model
     * @return
     */
}
```


资源服务器

项目中需要 资源服务B 调用 资源服务A；只需要修改 资源服务B

```
// ...
// 网关地址
private String BASE_URL = "http://192.168.62.1:9999";

@GetMapping("/res1")
public String getRes1(HttpServletRequest request) {
    //return getServer("http://127.0.0.1:8001/res2", request);
    return getServer(BASE_URL + "/o2_resource_a/res2", request);
    //return JSON.toJSONString(new Result(200, "服务B -> 资源1"));
}
```

式



The screenshot shows a web browser's address bar. The address bar contains the URL `http://192.168.62.1:9999/actuator/info`, which is highlighted with a red rectangular box. To the left of the address bar, there are navigation icons: back, forward, refresh, home, and star. Above the address bar, there are tabs for 'Eureka' and '192.168.62.1:9999/actuator/in'.

```
{ "ServiceName": "路由网关", "author": "tuwer", "version": "1.0-SNAPSHOT" }
```

## 网关基本信息

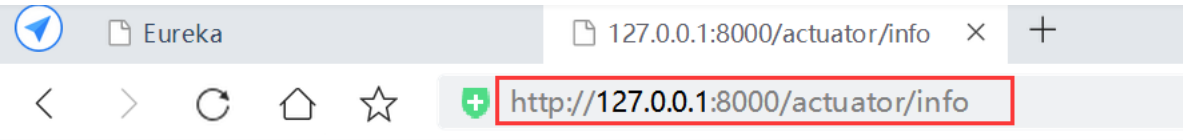
CSDN @土味儿~

Figure 1. Screenshot of the browser address bar showing the URL `http://os.com:9000/actuator/info`.

```
{ "ServiceName": "认证中心", "author": "tuwer", "version": "1.0-SNAPSHOT" }
```

## 认证中心基本信息

CSDN @土味儿~



{"ServiceName":"oauth2客户端","author":"tuwer","version":"1.0-SNAPSHOT"}

客户端基本信息

CSDN @土味儿~



{"ServiceName":"资源服务器A","author":"tuwer","version":"1.0-SNAPSHOT"}

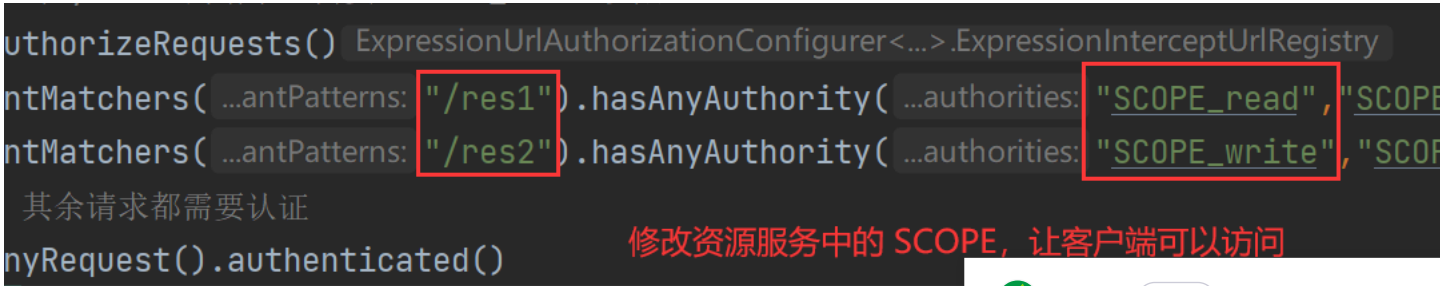
资源服务A 基本信息

CSDN @土味儿~



测试  
端中需要加入本地用户，第三方用户绑定本地用户，再把本地用户角色/权限赋予给第三方用户，实现客户端的角色管理。  
设计及说明见：[OAuth2在分布式微服务架构下基于角色的权限设计\(RBAC\)](#)

原服务器放开客户端的访问权限，只要是合法的客户端请求都通过



< > ↺ 🏠 ☆

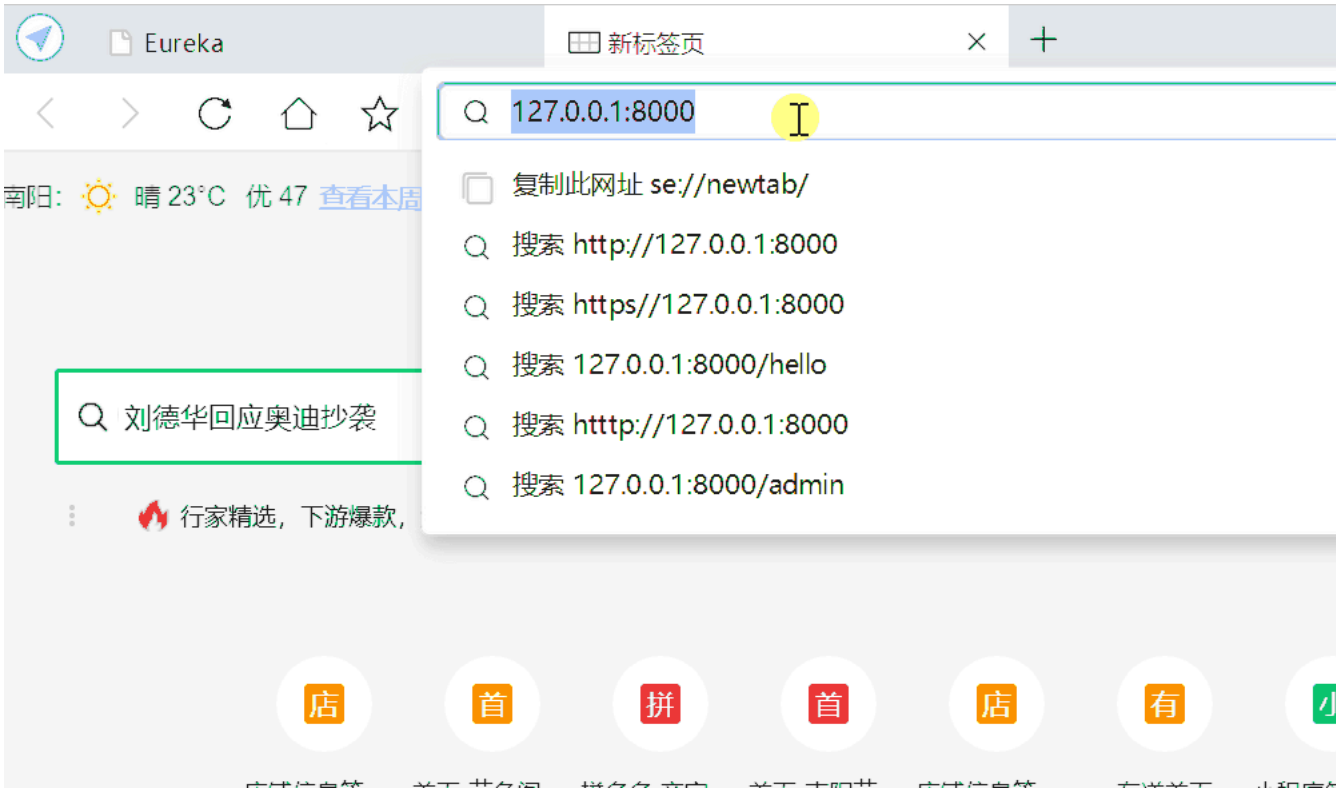
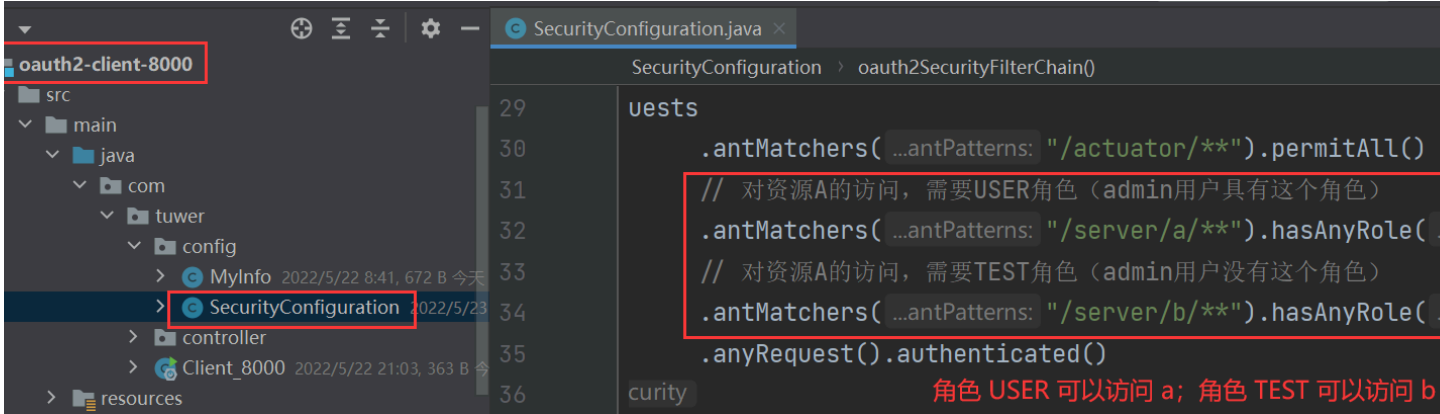
⚡ http://127.0.0.1:8000/server/a/res2

{"code":200,"data":"服务A -> 资源2","time":"2022-05-23T10:05:03.905"}

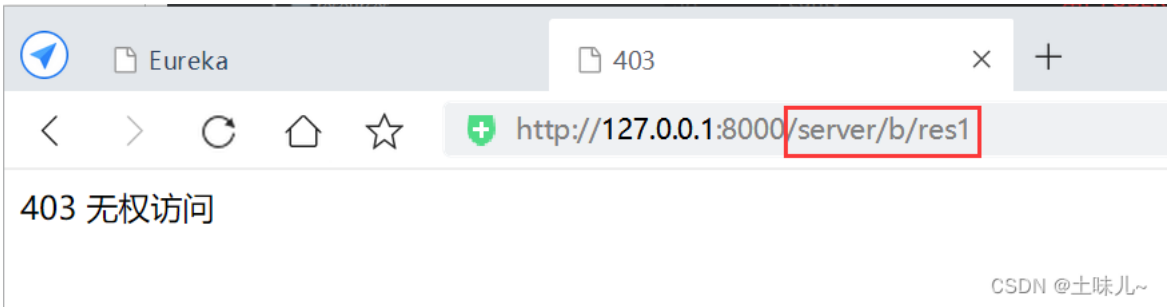
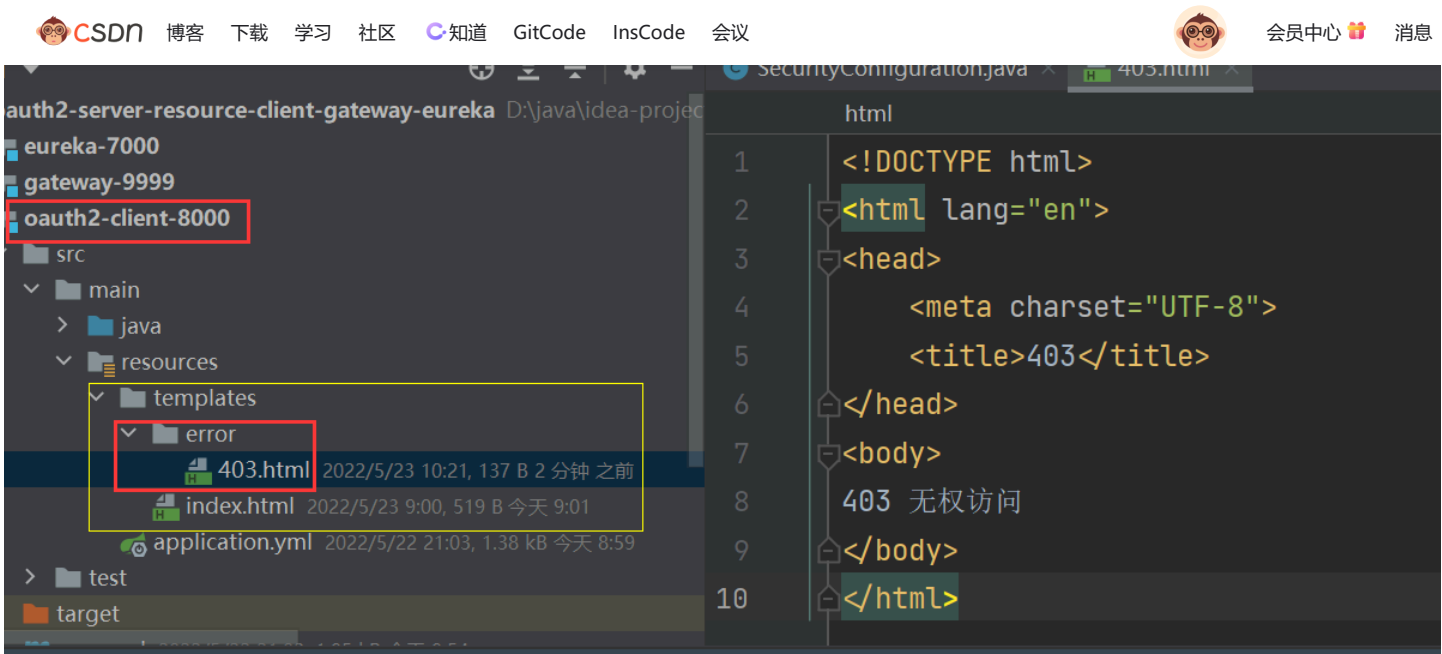
原来不能直接访问的，已经可以了

CSDN @土味儿~

➡端根据角色/权限控制访问；修改客户端中安全策略；重启客户端测试



加403页面



仓库: <https://gitee.com/tuwer/oauth2>

章知识点与官方知识档案匹配, 可进一步学习相关知识

能树 首页 概览 149710 人正在系统学习中

ng Authorization Server 0.2.3 的变化

pring Authorization Server又发新版本了, 现在的版本是0.2.3。本次都有什么改动呢? 我们来了解一下。0.2.3版本特性 本次更新的新特性不少。为公开客户端提供默认的设置

boot-2-oauth2-resource-server: 带有用户和客户端数据库 (JPA, Hibernate, MySQL) 的Spring Boot 2 OAuth2资源和授权服务器实现

boot 2 OAuth2资源和授权服务器 使用用户和客户端数据库... git clone <https://github.com/indrekru/spring-boot-2-oauth2-resource-server>.git 您需要在您的环境中安装Maven: M

论  weixin\_61557632 热评 博主, 单点登出, 怎么实现, a应用登出, 只能清空本地会话、全局会话, 其它应用的本地会话还存在, 如果不调用a/oauth2/authorize

gateway 整合spring authorization server

cloud Gateway 作为服务网关,可以与认证服务(如Spring Authorization Server或其他OAuth2服务)集成,实现对所有API请求的统一认证和授权。以下是对接认证服务的基本步骤: 1

Cloud 服务和网关整合OAuth2实现权限控制实战全流程

cloud OAuth2 authorization server 实现源码在gihub上,确保nacos和mysql数据库启动即可。资源服务器 使用Spring Security 5.2.x来实现的。必须依赖spring-boot-starter-secu

Authorization Server入门 (十六) Spring Cloud Gateway对接认证服务

单独讲过Security Client和Resource Server的对接, 但是都是基于Spring webmvc的, Gateway这种非阻塞式的网关是基于webflux的, 对于集成Security相关内容略有不同, 且

!.0 实践 Spring Authorization Server 搭建授权服务器 + Resource + Client

0 实践 Spring Authorization Server 搭建授权服务器 + Resource + Client

starter-oauth2-authorization-server

boot 3.1 提供了一个 spring-boot-starter-oauth2-authorization-server 启动器,可以支持 Spring Authorization Server 的自动配置,轻松配置基于 Servlet 的 OAuth2 授权服务器,同时

ring Authorization Server配合Spring Gateway授权登录后重定向错...

的授权码模式有误,设计授权接口为需要传入accessToken,修改时因为去掉了传入令牌认证所以添加了默认的表单登陆。表现 表现为通过网关访问授权接口,重定向到login登陆

security最新学习, [spring-security-oauth2-authorization-server](#) 【spring-security-oauth2升级】



CSDN

博客 下载 学习 社区

知道

GitCode

InsCode

会议



会员中心



消息

新的授权服务器Spring Authorization Server入门

Spring官方已经强烈建议使用Spring Authorization Server替换已经过时的Spring Security OAuth2.0[1]，距离Spring Se...

Cloud Gateway 整合OAuth2.0 实现统一认证授权

配置 @EnableAuthorizationServer 新建AuthorizationServerConfig.java UserDetailsServiceImpl 从数据库获取用户信息。 身份验证提供者 DaoAuthenticationProvider JwtAuth...

g Cloud Gateway 与OAuth2模式一起使用\_gateway集成oauth2...

Connect 定义了一种基于 OAuth2 授权代码流的最终用户身份验证机制。下图是Spring Cloud Gateway与授权服务进行身份验证完整流程,为了清楚起见,其中一些参数已被省略。

12 客户端模式】Spring Authorization Server + Resource + Client 资源服务间的相互访问

上一节中介绍了项目的搭建,并实现了授权码模式的访问。在上一节的基础上,再来实现客户端模式。【图文详解】搭建 Spring Authorization Server + Resource + Client 完整

authorization-server系列 (1) ----demo

cret\_basic: 将 clientId 和 clientSecret 通过 ':' 号拼接, ( clientId 和 clientSecret 都在上面配置中, ) 并使用 Base64 进行编码得到一串字符, 再在前面加个 注意有个 Basic 前

Cloud 之 Gateway、Spring Security、Oauth2 的整合

orizationManager 3、ResourceServerConfig 4、配置文件 源码地址:看这里,看这里,看这里 gateway为微服务的网关,所有的访问服务的api都要通过网关转发到内网对应的服务。

Cloud OAuth2 搭建授权服务器 + 客户端 + 令牌中继

oot 版本2.1.4.RELEASE、Spring Cloud版本Greenwich.RELEASE 说明: token采用redis存储, 用户信息采用数据库存储 oauth2官网整合springboot的例子(含服务端配置和客户

boot-oauth2-demo

该项目展示了如何利用Spring Boot集成OAuth2, 创建一个认证服务器 (Authorization Server) 以及资源服务器 (Resource Server)。 OAuth2是一个开放标准, 用于授权第三

auth2-server:完整, 合规且经过良好测试的模块, 用于在node.js中使用express实现OAuth2 ServerProvider

合规且经过良好测试的模块, 用于在实现OAuth2服务器。 注意: 经过一段时间的中断后, 该项目现在可以正常维护。 依赖关系已更新, 错误修复将在v3 (当前版本) 中发布。 叉

security-oauth2-authorization-server.zip

pring-security-oauth2-authorization-server"将带你深入理解如何利用Spring Security OAuth2构建一个授权服务器, 以保护你的API并提供安全的访问控制。 OAuth2是一种开放

security-oauth2文档

人员提供了一种简单的方式去实现OAuth2协议的支持, 使得开发者能够轻松地在应用程序中集成授权服务器 (Authorization Server) 和资源服务器 (Resource Server)。 ##

单的Spring Authorization Server示例代码

定的OAuth2的基础 需要有一定的Spring Security基础 Spring Authorization Server 官方简介: Spring Authorization Server is a framework that provides implementations of the

羊解】搭建 Spring Authorization Server + Resource + Client 完整Demo 热门推荐

的Demo, 有认证端, 有资源端, 有客户端; 采用当前最新的技术。 非常感谢 码农小胖哥, 仔细阅读了他的很多文章。本项目中的很多逻辑和代码都源自于他。如果想深入学

Authorization Server 1.1 扩展实现 OAuth2 密码模式与 Spring Cloud 的整合实战

开源微服务商城项目 youlai-mall、Spring Boot 3 和 Spring Authorization Server 1.1 版本, 演示了如何扩展密码模式, 以及如何将其应用于 Spring Cloud 微服务实战。

Authorization Server实战

一个开放标准的授权协议, 允许用户授权第三方应用访问其在某个服务提供者上的受保护资源, 而无需将其实际的凭证 (比如用户和密码) 分享给第三方应用。 OAuth2.0协议的

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

从0到1搭建了一个简单认证服务, 解释了认证服务的各项配置用意, 如何设置自己的登录页和授权确认页, 如何让认证服务解析请求时携带的token, 文章过长难免有遗漏的地

级环境中部署Java程序: Docker命令实用指南 最新发布

Java应用程序的部署提供了一种快速、一致且可移植的方式。掌握这些基本的Docker命令, 可以帮助你在企业级环境中更高效地部署和管理Java应用。

boot-starter-oauth2-authorization-server 3.1.0 简单使用

boot-starter-oauth2-authorization-server"是Spring Boot提供的OAuth2认证服务器的Starter, 可以用于构建安全的OAuth2认证授权服务器。 以下是使用 'spring-boot-starter-oautl

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号 经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务 中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证 营业执照

©1999-2024北京创新乐知网络技术有限公司

土味儿~

冯龄3年 暂无认证

6729 38万+ 36万+

哥排名 总排名 访问 等级

1万+ 435 194 2109

粉丝 获赞 评论 收藏

土味儿~

已关注

https://blog.csdn.net/tu\_wer/article/details/124972455

15/17





私信

已关注




知道，每日最新动态  
AI新鲜事！




流量券免费送  
一篇就可获得！

文章



哈夫曼编码与压缩比  22635

步骤 用例图 类图 对象图 包图 顺序图 状态图 活动图 协作图 

作  19754

【图解】搭建 Spring Authorization  
Resource + Client 完整Demo 

司机7种耦合关系  11795

buntu	4篇
lasticSearch	6篇
端	7篇
 Bootstrap	5篇
pring	5篇
 SpringCloud	25篇



【图解】搭建 Spring Authorization ...  
你这个是springcloud项目吗？前  
下的实现方案吗？不太理解这’ ...

【图解】搭建 Spring Authorization ...  
感谢作者，我在一个节点卡主  
成功获取用户信息的时候，Aut ...

【图解】搭建 Spring Authorization ...  
：访问资源的时候只带了token并  
stoken，确定可以自动刷新to ...

网络编程深入剖析【尚硅谷】  
ly: FileChannelDemo1 这个demo  
没有写入的操作 还要进行读写 ...

司机7种耦合关系  
5566: 大佬讲的太清晰了，能再讲

 土味儿~ 

已关注



ot中java操作Excel【EasyExcel】

安装与使用 (SpringCloud)

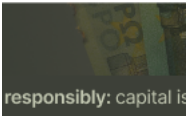
牌桶

19篇 2022年 126篇

34篇



Start trading



概述

概述

整体架构图

2.1、总体结构

2.2、用户访问

2.3、资源服务间调用

2.4、额外说明

搭建环境

搭建

父工程

子模块

中心

pom.xml