OAuth2在分布式微服务架构下基于角色的权限设计(RBAC)

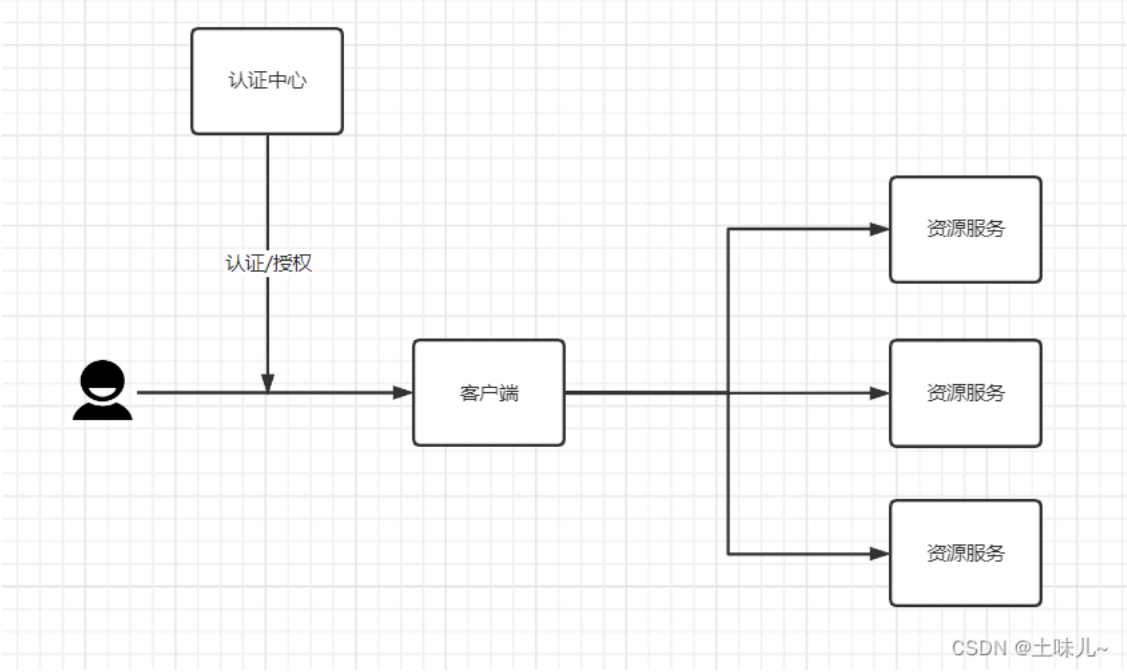
原创 土味儿~ 已于 2022-05-25 13:54:13 修改 阅读量3.5k 收藏 17 点赞数 6

分类专栏： # SpringSecurity OAuth2 # SpringCloud 文章标签： 架构 分布式 微服务

SpringCloud 同时被 2 个专栏收录 12 订阅 25 篇文章

> 在前两节的基础上，对 权限控制 作进一步的分析与设计。
> RBAC (Role-Base Access Control，基于角色的访问控制)
> 本篇内容基于个人理解，不当之处，欢迎批评指正。
> 前两篇内容：
>
> - 【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

1、OAuth2中用户访问的基本流程

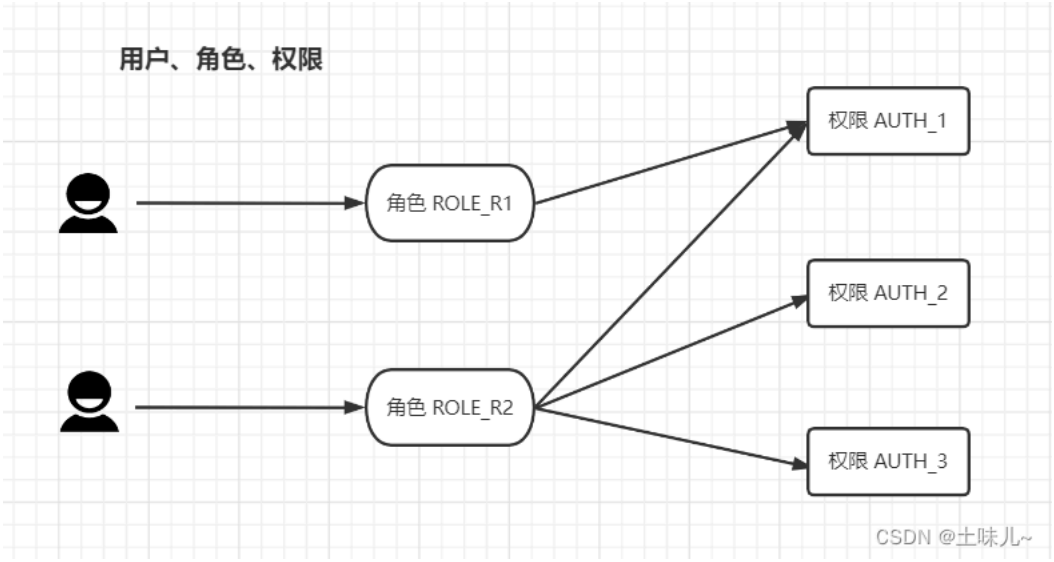


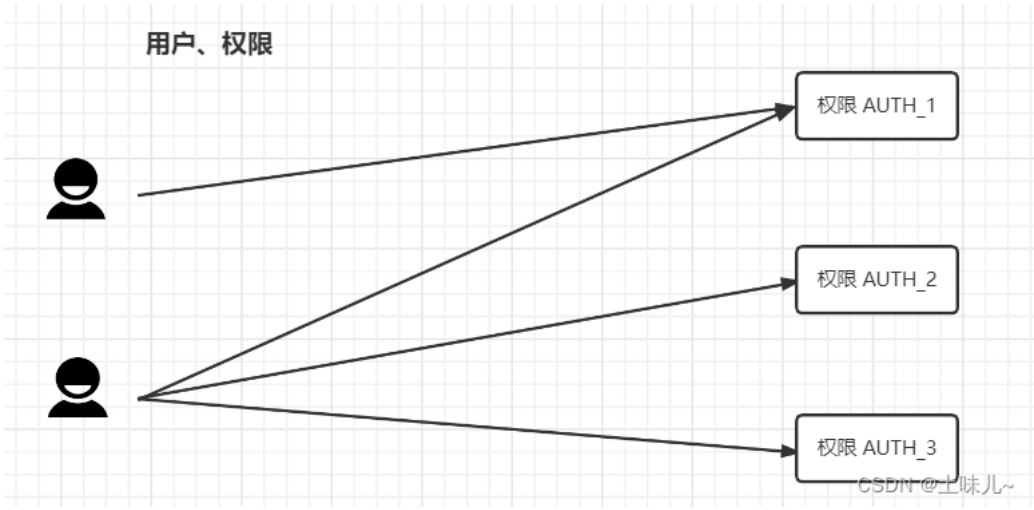
- 用户经过认证/授权后，进入客户端（认证中心给客户端发放令牌），客户端携带令牌访问对应的资源。
- 客户端是用户和资源之外的第三方，要想访问资源必须得到用户的允许。
- 用户拥有资源，通过客户端去访问，把访问权限赋于给了客户端。

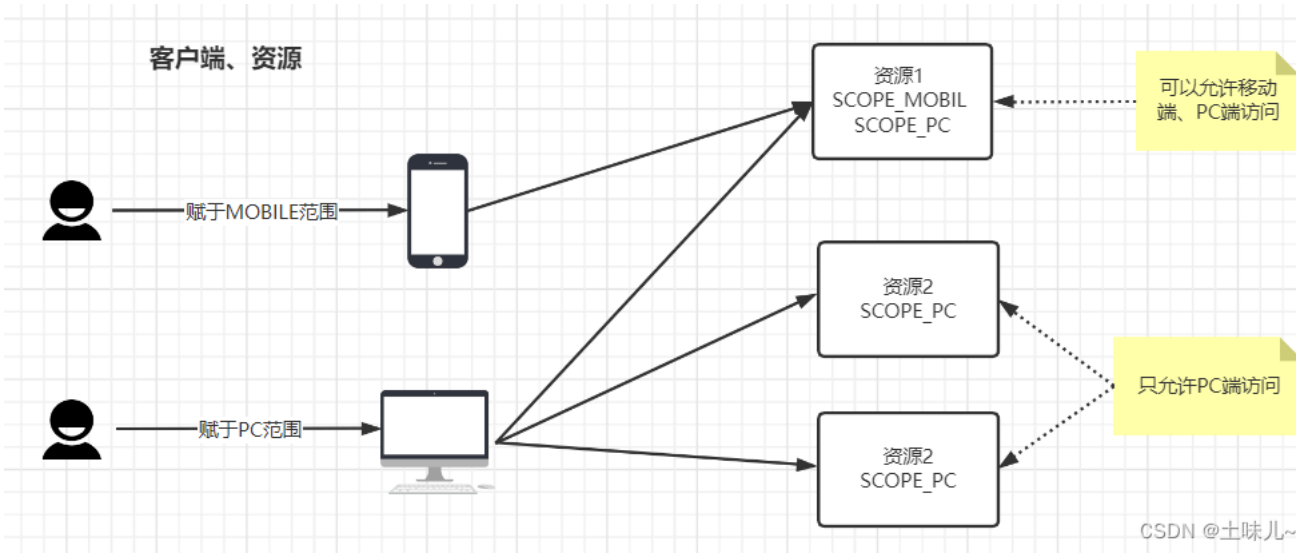
2、SCOPE、ROLE、AUTH 区别

- **SCOPE**：范围；指用户授权客户端可以访问的范围。客户端只能在这个范围内去访问。是针对客户端来说的。
- **ROLE**：角色；是用户的身份。是针对用户来说的。
- **AUTH**：权限；是角色所拥有的。角色与权限是多对多的关系；一个角色可以有多个权限，一个权限也可以同时被多个角色所拥有。权限也可以直用户，如果用户不指定角色，可以直接把权限赋于用户。

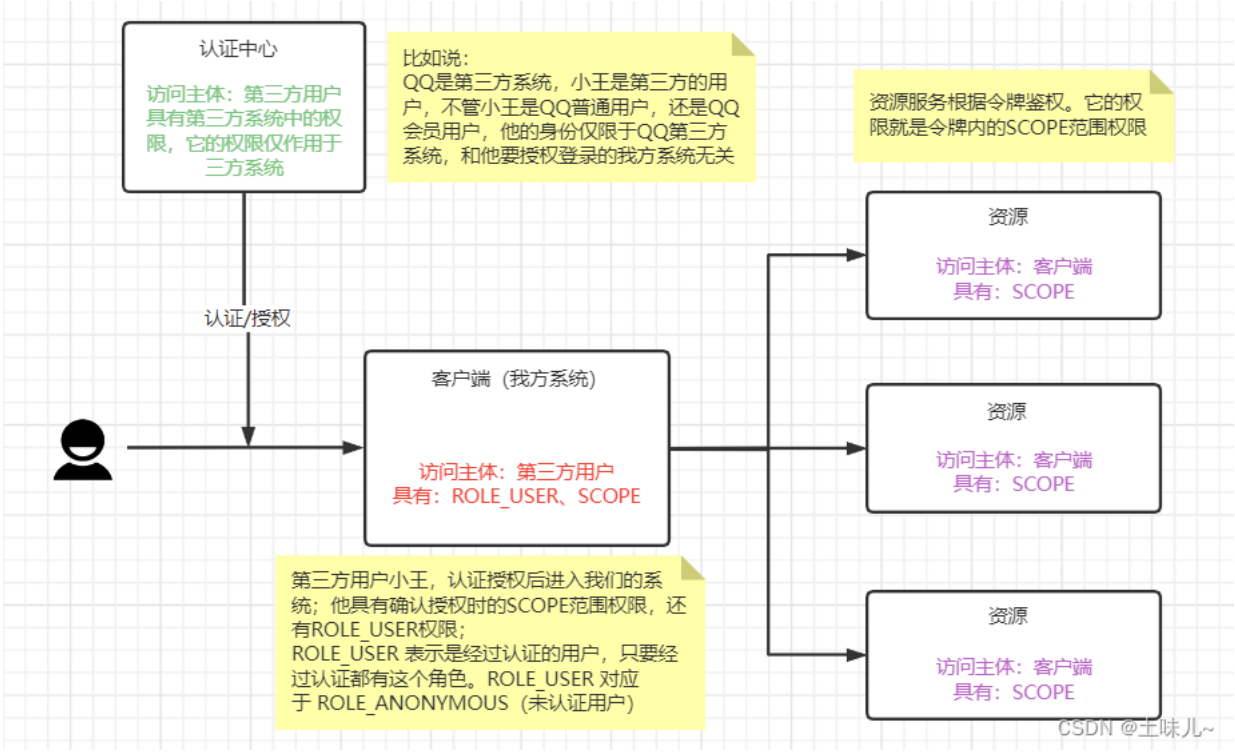
区别	含义	面向对象
SCOPE	范围	客户端
ROLE	角色	用户
AUTH	权限	角色 或 用户







3、server、resource、client 中访问主体的区别

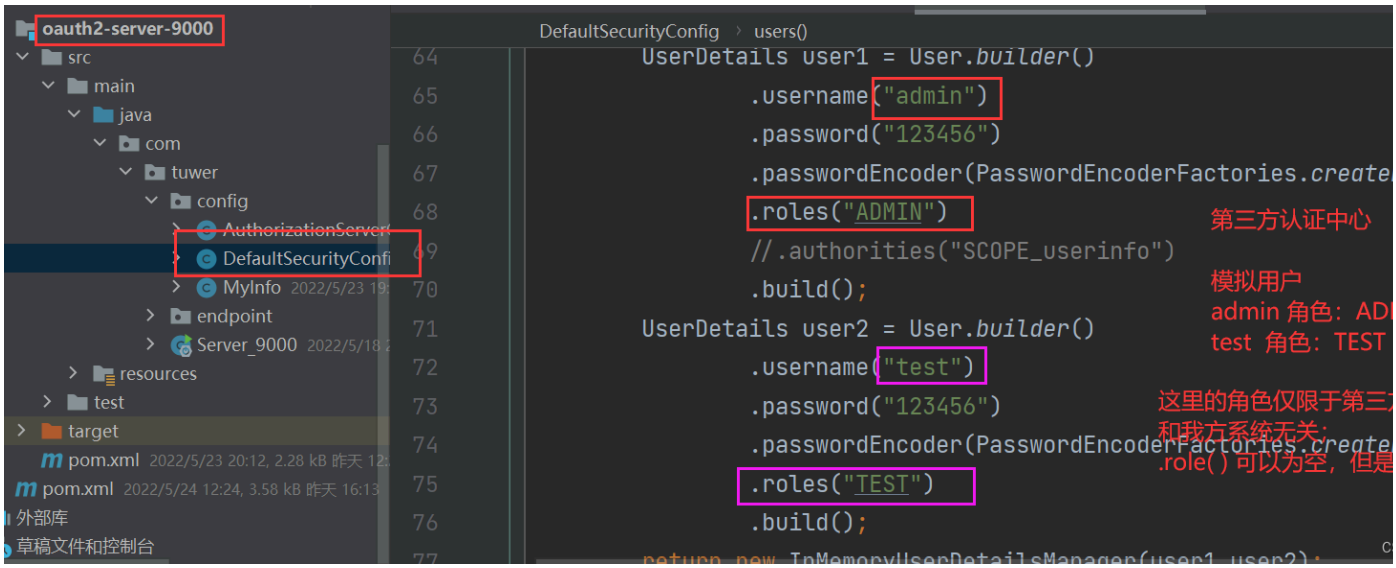


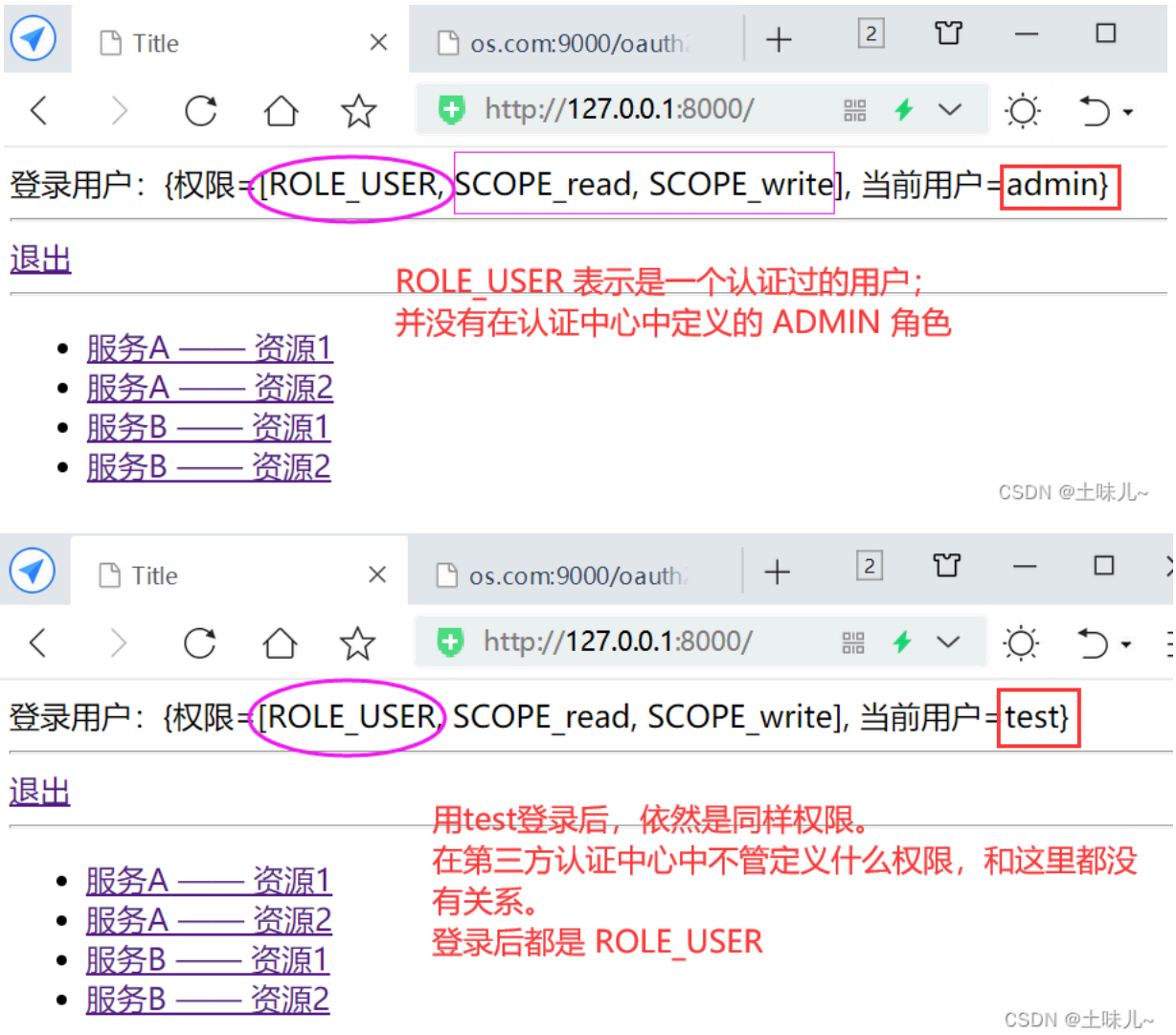
从图中可以看出，在每个系统中的访问主体及权限是不同的（这里的权限是统称，包括SCOPE、ROLE、AUTH，不仅仅指AUTH）

- 当用户登录后，在认证中心内，访问主体就是 **第三方用户**，它的权限是他在认证中心中的权限，和我方系统无关
- 在客户端中，访问主体还是 **第三方用户**，权限包括：用户授于客户端的 `SCOPE` ，以及 `ROLE_USER` ；

> `ROLE_USER` 表示这是一个经过认证的用户，不管第三方用户在第三方系统中是什么身份，只要进入到我方系统中，就是 `ROLE_USER` 身份；对
> `ROLE_ANONYMOUS` （未认证用户）

- 客户端携带令牌访问资源，在资源服务器中访问主体就是 **客户端**，权限只有： `SCOPE` ；因为客户端是在用户授权下去访问的，所以在认证中心生只包括了用户授于的 SCOPE，不可能把用户的身份ROLE也赋于客户端。





4、访问控制 分析

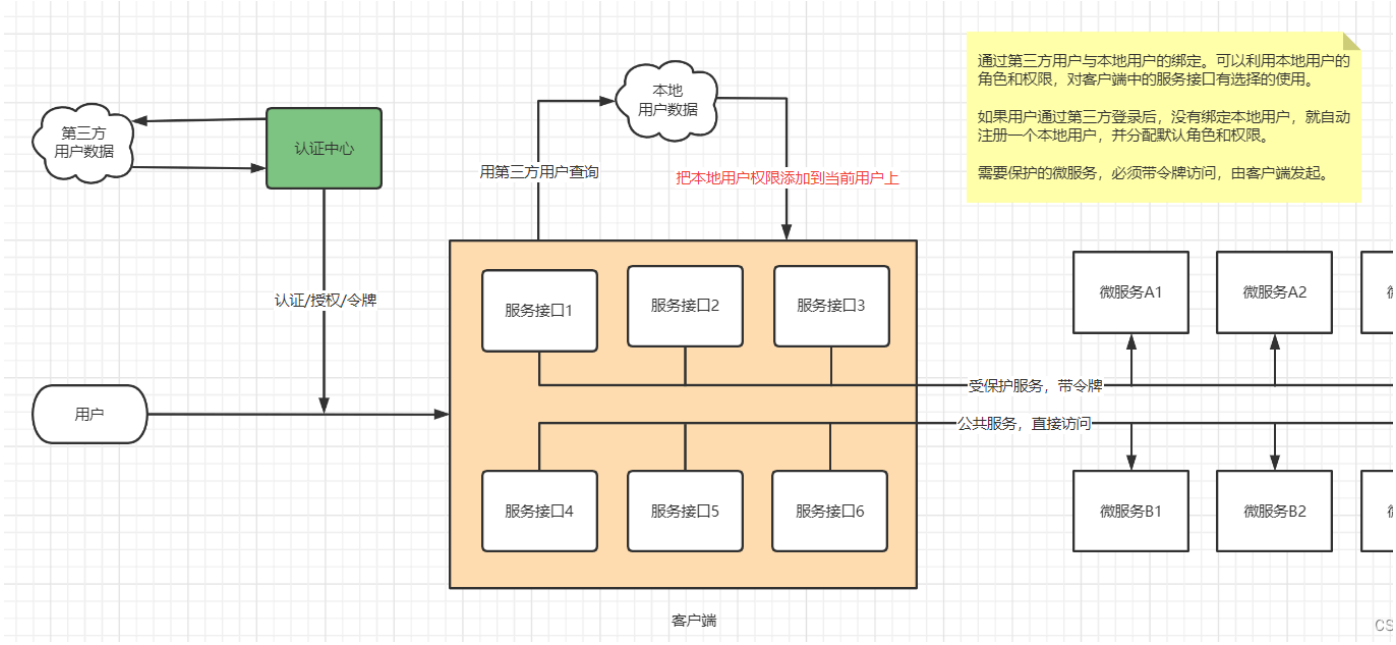
通过上面的分析可以发现，资源端只有 SCOPE，不可能用 ROLE 或 AUTH 去控制用户的访问。认证中心不负责访问资源，要想通过 ROLE 或 AUTH 去访问资源，只能在 客户端 去操作。资源API在客户端有对应的接口，要想控制资源API，就控制客户端的对应接口就可以了。只要用户能访问客户端的某接口，它就能访问与之对应的资源API。

- **资源API 面向 SCOPE 开放**；
- **客户端API 面向 ROLE 或 AUTH 开放**；

但是，所有第三方用户，进入我方系统后，都具有 ROLE_USER 身份，身份是一样的，如何在客户端中通过 ROLE 或 AUTH 去控制用户访问资源呢？

解决方案：添加本地用户，赋于不同的 ROLE 或 AUTH ；第三方用户与本地用户实现绑定；通过本地用户的 ROLE 或 AUTH 去控制用户访问资源。这是的一个通用做法。

那第三方用户进入我方系统后，如何改变他的身份？把本地的 ROLE 或 AUTH 赋给他呢？办法就是**权限提升**！

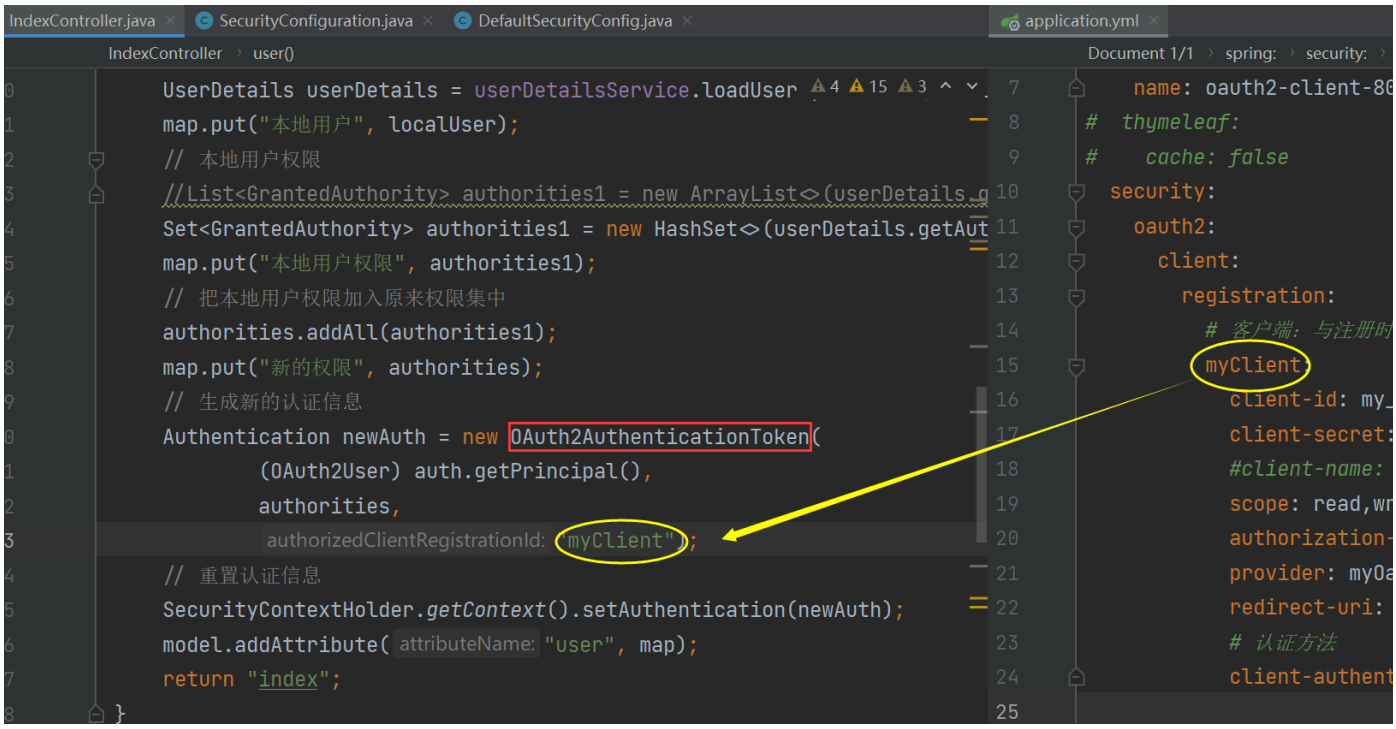


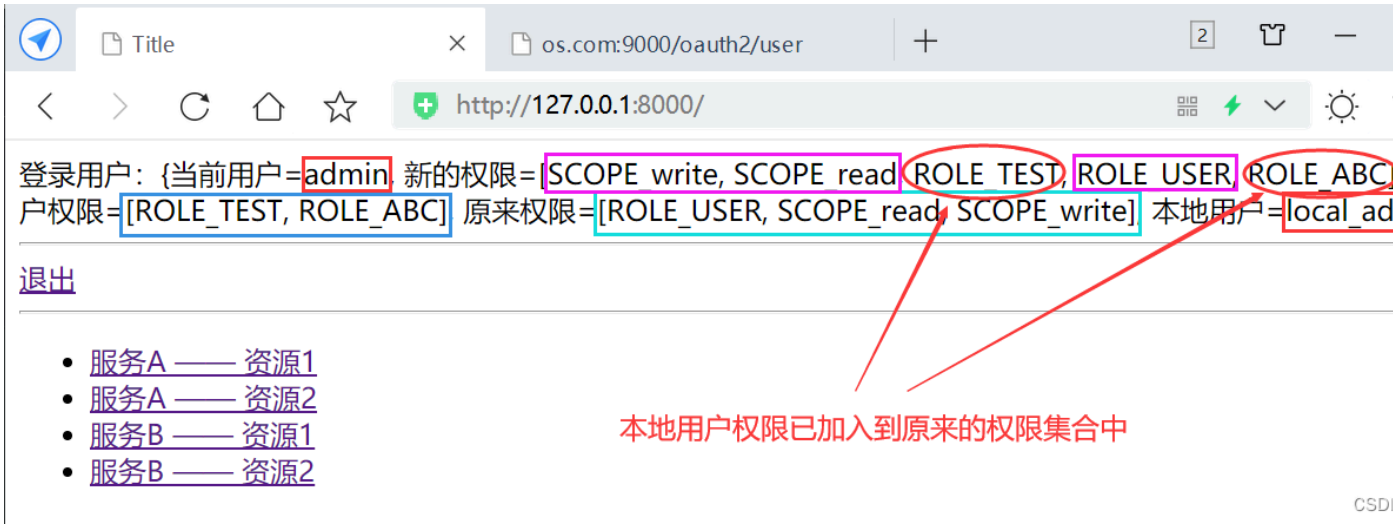
5、客户端权限提升

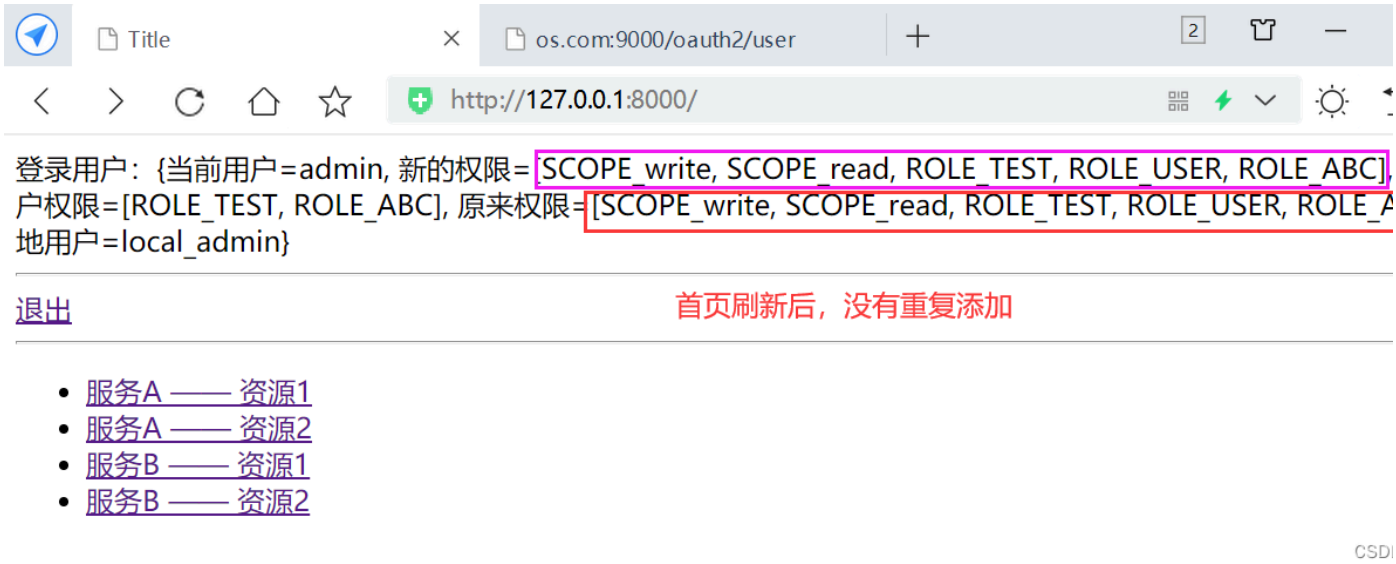
- 第三方用户进入我方系统后，从 `SecurityContextHolder` 中获取第三方用户的 `name` 和 `authorities`
- 根据第三方用户的 `name` ，查询绑定的本地用户，进而得到本地用户的 `authorities`
- 把本地 `authorities` 加入到 第三方用户的 `authorities` 中
- 重新生成新的 `Authentication`
- 注入 `SecurityContextHolder` 中，替换原来的 `authorities` ，完成权限提升

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

    @GetMapping("/")
    public String user(Model model) {
        // 从安全上下文中获取登录信息，返回给model
        Map<String, Object> map = new HashMap<>(5);

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();
        String username = auth.getName();
```

```
@Configuration
public class SecurityConfiguration {
    /**
     * 虚拟一个本地用户
     *
     * @return UserDetailsService
     */
    @Bean
    UserDetailsService userDetailsService() {
        return username -> User.withUsername("local_admin")
                .password("123456")
```


IndexController.java
SecurityConfiguration.java
DefaultSecurityConfig.java
application.yml
IndexController › user()
Document 1/1 › spring: › security: ›
UserDetails userDetails = userDetailsService.loadUser
map.put("本地用户", localUser);
// 本地用户权限
//List<GrantedAuthority> authorities1 = new ArrayList<>(userDetails.g
Set<GrantedAuthority> authorities1 = new HashSet<>(userDetails.getAut
map.put("本地用户权限", authorities1);
// 把本地用户权限加入原来权限集中
authorities.addAll(authorities1);
map.put("新的权限", authorities);
// 生成新的认证信息
Authentication newAuth = new OAuth2AuthenticationToken(
(OAuth2User) auth.getPrincipal(),
authorities,
authorizedClientRegistrationId: "myClient");
// 重置认证信息
SecurityContextHolder.getContext().setAuthentication(newAuth);
model.addAttribute(attributeName: "user", map);
return "index";
}
name: oauth2-client-80
thymeleaf:
cache: false
security:
oauth2:
client:
registration:
客户端：与注册时
myClient:
client-id: my_
client-secret:
#client-name:
scope: read,wr
authorization-
provider: myOa
redirect-uri:
认证方法
client-authent


Title
os.com:9000/oauth2/user
http://127.0.0.1:8000/
登录用户：{当前用户=admin, 新的权限=[SCOPE_write, SCOPE_read, ROLE_TEST, ROLE_USER, ROLE_ABC]
户权限=[ROLE_TEST, ROLE_ABC] 原来权限=[ROLE_USER, SCOPE_read, SCOPE_write] 本地用户=local_ad
退出
服务A —— 资源1
服务A —— 资源2
服务B —— 资源1
服务B —— 资源2
本地用户权限已加入到原来的权限集合中

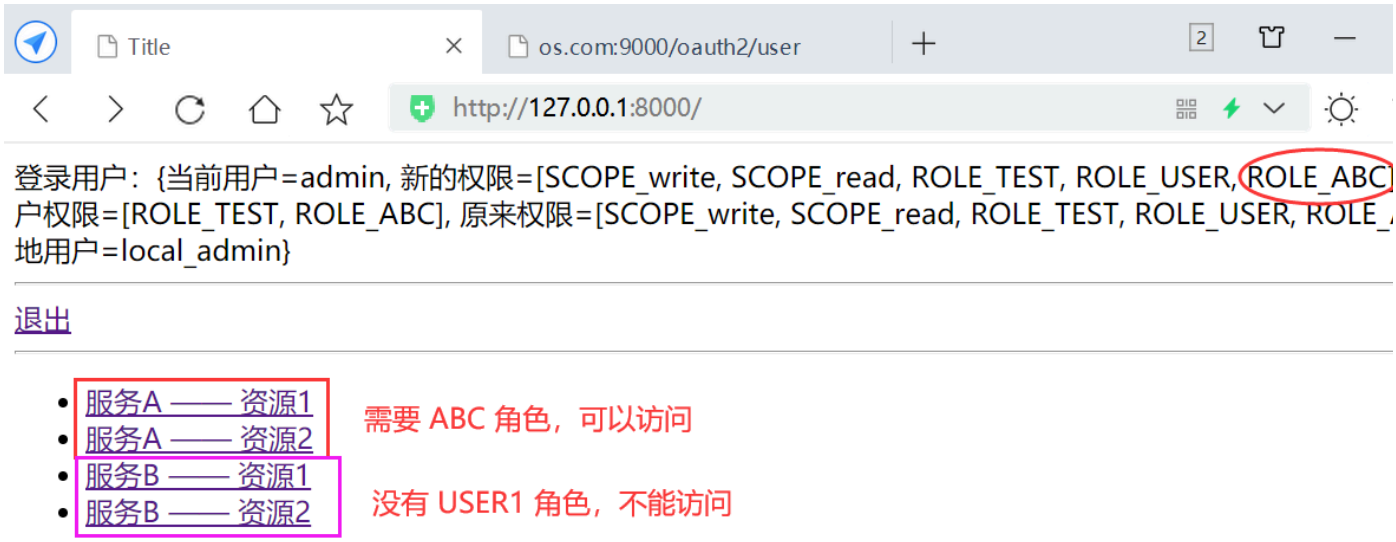

Title
os.com:9000/oauth2/user
http://127.0.0.1:8000/
登录用户：{当前用户=admin, 新的权限=[SCOPE_write, SCOPE_read, ROLE_TEST, ROLE_USER, ROLE_ABC],
户权限=[ROLE_TEST, ROLE_ABC], 原来权限=[SCOPE_write, SCOPE_read, ROLE_TEST, ROLE_USER, ROLE_A
地用户=local_admin}
退出
首页刷新后，没有重复添加
服务A —— 资源1
服务A —— 资源2
服务B —— 资源1
服务B —— 资源2

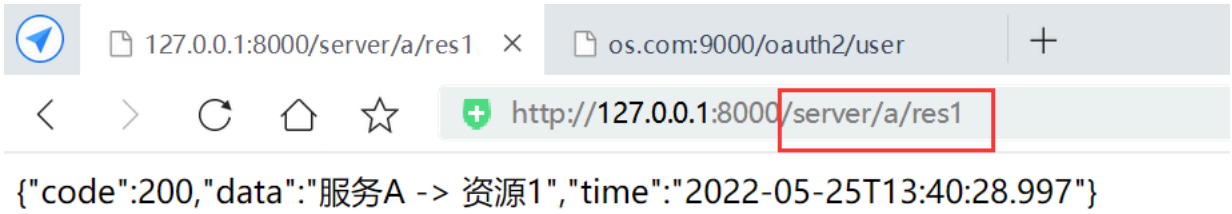
- 访问测试

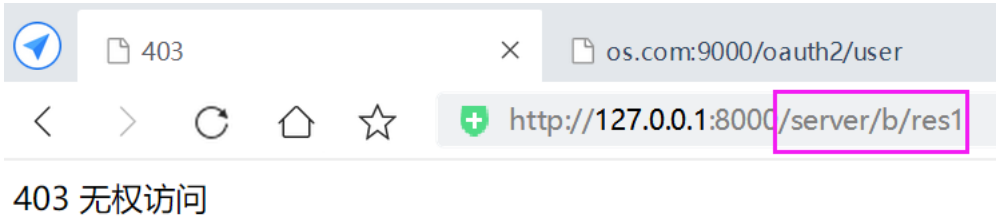
```
http.authorizeRequests(requests →
        // 任何请求都需要认证
        requests
                .antMatchers( ...antPatterns: "/actuator/**").permitAll()
                // 对资源A的访问，需要USER角色（admin用户具有这个角色）
                .antMatchers( ...antPatterns: "/server/a/**").hasAnyRole( ...roles: "ABC
                // 对资源A的访问，需要TEST角色（admin用户没有这个角色）
                .antMatchers( ...antPatterns: "/server/b/**").hasAnyRole( ...roles: "USE
                .anyRequest().authenticated()
    );
http.oauth2Login();
http.oauth2Client();
return http.build();
```

ABC 是本地用户的角色，权限提升前无法访问，提升后可以；

权限提升前后都不具有 USER1 角色，不能访问







6、权限设计

- 客户端

客户端分类	被授于的 SCOPE
电脑端	SCOPE_1
手机端	SCOPE_2
内部资源服务	SCOPE_0

- 资源端

资源分类	允许访问的 SCOPE	说明
r1/res1	SCOPE_0、SCOPE_1、SCOPE_2	资源服务器1 中的 资源1，可以被三个客户端访问
r1/res2	SCOPE_0、SCOPE_1	资源服务器1 中的 资源2，只可以被电脑端、内部资源访问
r2/res1	SCOPE_0、SCOPE_2	资源服务器2 中的 资源1，只可以被手机端、内部资源访问
r2/res2	SCOPE_1、SCOPE_2	资源服务器2 中的 资源2，只可以被电脑端、手机端访问
r3/res1	SCOPE_2	资源服务器3 中的 资源1，只可以被手机端访问
r3/res2	SCOPE_0	资源服务器3 中的 资源2，只可以被内部资源访问

- 用户与角色

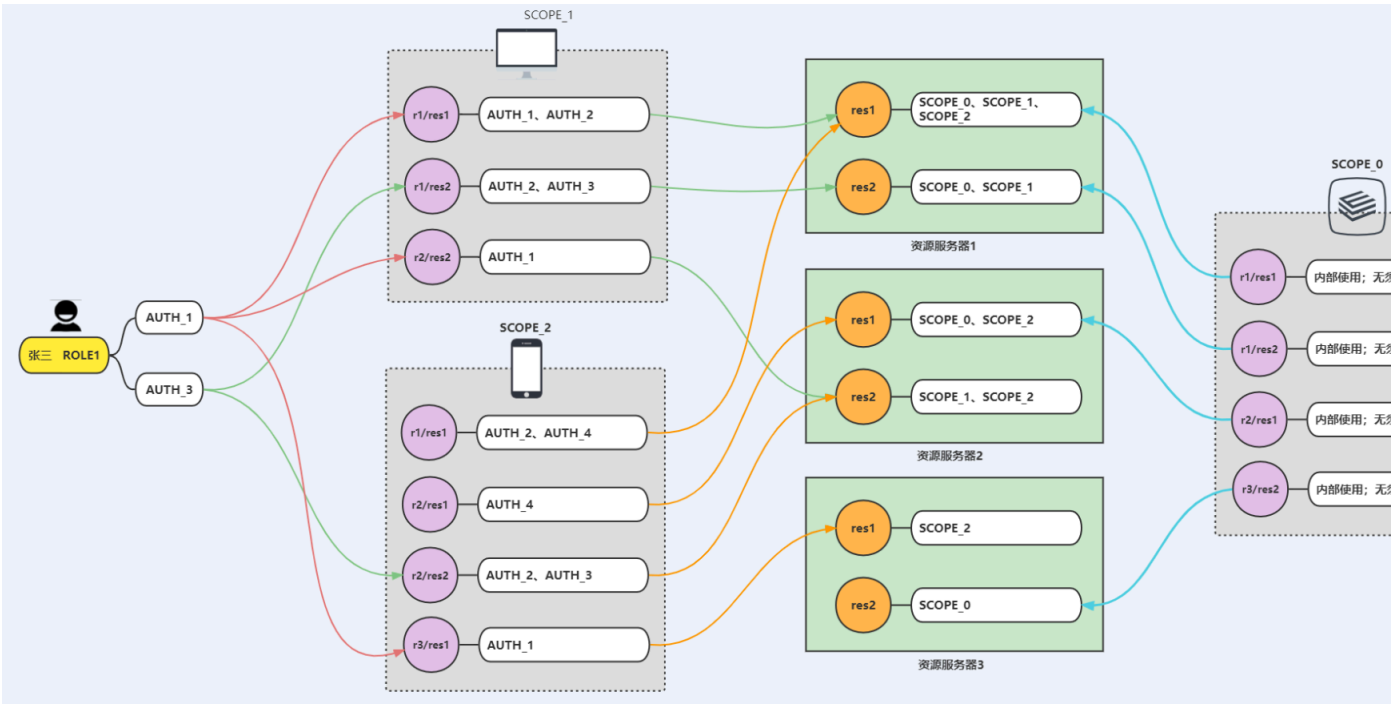
用户	角色
张三	ROLE_1
李四	ROLE_2

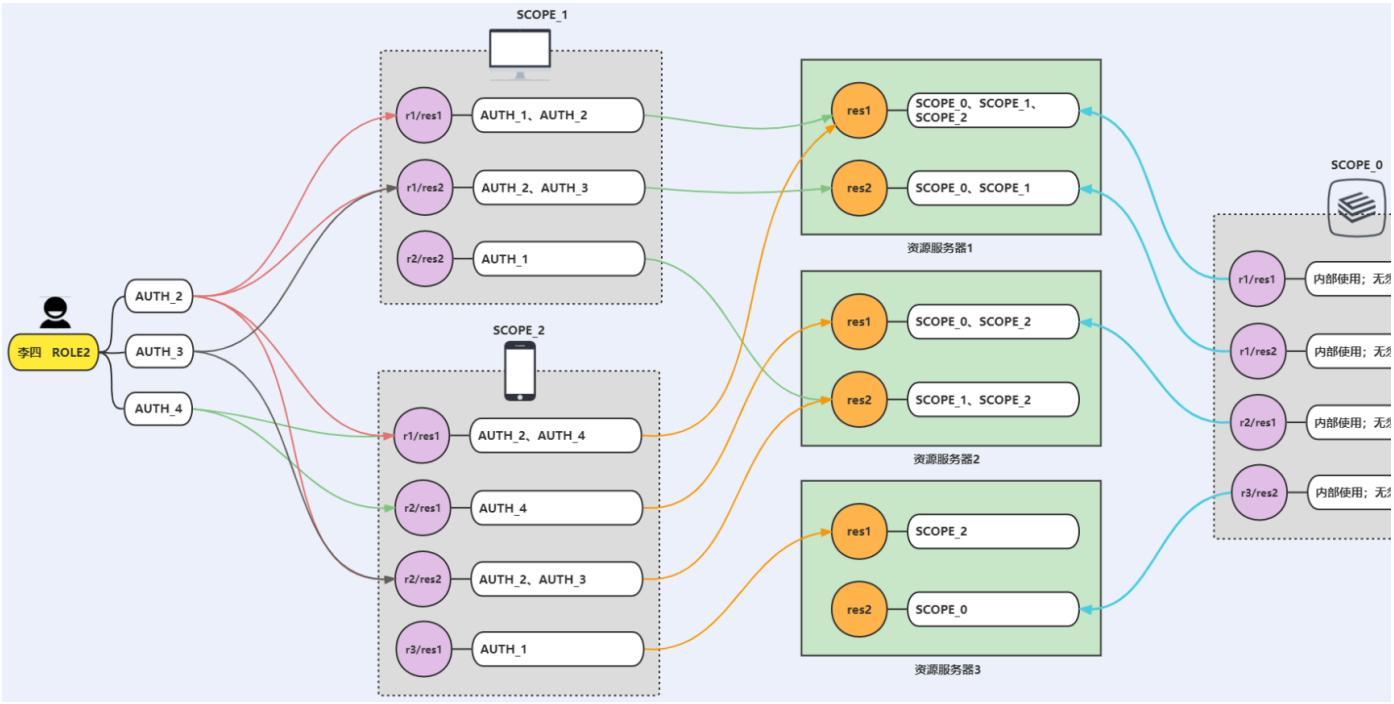
- 角色与权限

权限	角色
AUTH_1	ROLE_1
AUTH_2	ROLE_2
AUTH_3	ROLE_1、ROLE_2
AUTH_4	ROLE_2

ROLE_1：包含 AUTH_1、AUTH_3

ROLE_2：包含 AUTH_2、AUTH_3、AUTH_4





- 客户端与资源的访问绑定关系是一一对应的，应该相应稳定。客户端能访问某个资源就提供一个接口。不能随时修改。
- 用户通过角色访问客户端中的服务API，这个关系比较灵活， 相对松散。客户端中只需指定某个接口可以被哪些AUTH访问即可。
- 角色ROLE与权限AUTH的关系相对稳定，但可以比客户端和资源的关系灵活，可以修改编辑。
- 在项目设计阶段，应该首先确定**客户端**的种类，再基本确认项目中所涉及的**角色**。根据资源API功能，决定需要哪些权限，应该把权限赋于哪种角

6 条评论  開心呱呱 热评 动态权限怎么切入呢?

SSO、OAuth2、JWT、CAS、OpenID、LDAP、RBAC_cas oauth2
OAuth2的实现到底是不是单点登录呢,这个问题答案是比较模糊的,简单地说是或者不是回答都不够准确。但目前我们一般讲到OAuth2,都会比较普遍的认为他是属于单点登

Spring Boot、 Spring Cloud 、OAuth2 的RBAC 权限管理系统分享_oauth...
本书涵盖Spring 5的所有内容,如果想要充分利用这一领先的企业级 Java应用程序开发框架的强大功能,本书是最全面的Spring参考和实用指南。 本书第5版涵盖核心的Sprir

基于RBAC 的权限实现Demo
基于角色的权限访问控制（Role-Based Access Control）作为传统访问控制（自主访问，强制访问）的有前景的代替受到广泛的关注。在RBAC中，权限与角色相关联，用

基于角色的权限控制（RBAC）介绍 老徐的博客只有
什么是RBAC RBAC（Role-Based Access Control）基于角色的权限控制。其基本思想是，对系统操作的各种权限不是直接授予具体的用户，而是在用户集合与权限集合之

Spring Security OAuth2.0认证授权 --- 基础篇_spring oauth2.0-CS...
根据上边的例子发现,当需要修改角色的权限时就需要修改授权的相关代码,系统可扩展性差。 1.5.2、基于资源的访问控制 RBAC基于资源的访问控制(Resource-Based Acc

Spring Gateway+Security+OAuth2+RBAC 实现SSO统一认证平台_spring ga...
上面写的比较简单,因为中间的坑实在是太多了,全网没有一个全面的可用的Spring gateway+Spring Security+OAuth2 的例子,实在是很遗憾,在找了N多文档之后,才完成了上

OAuth 2.0 角色 **最新发布** qq_33240556的
在 OAuth 2.0 中，有四个主要角色，它们在整个授权过程中各自扮演不同的角色。了解这些角色对于理解 OAuth 2.0 的工作机制至关重要。

springCloud-分享版-基于Spring Cloud、OAuth2.0的前后端分离的系统。 通用RBAC权限设计及其数据权限和分库分表 支持服务限流、动
springCloud-分享版: 基于Spring Cloud、OAuth2.0的前后端分离的系统。 通用RBAC权限设计及其数据权限和分库分表 支持服务限流、动态路由、灰度发布、 支持常见登

【SpringSecurity】十七、OAuth2授权服务器 + 资源服务器Demo_spring...
com/spring-projects/spring-security-oauth/blob/master/spring-security-oauth2/src/test/resources/schema.sql 1 基于RBAC,简化下,只要角色,不要权限表,表结构为: 1)用户表

Spring Boot整合 Spring Security、 OAuth2_springboot security oauth...
Spring Boot整合 Spring Security、 OAuth2 Spring Security 1、Spring Security功能 身份认证,授权,防御常见攻击如csrf 2、RBAC权限模型 RBAC模型(Role-Based Access

OAuth 2.0系列教程（三） 角色 Gavin0808的
OAuth 2.0系列教程（三） 角色 原文地址：http://tutorials.jenkov.com/oauth2/roles.html 作者：Jakob Jenkov 译者：林浩 校对：郭蕾 OAuth 2.0为用户和应用定义了如

spring security oauth2 RBAC 基于角色的访问控制 qq_44758028的
RBAC 基于角色的访问控制 概述 RBAC（Role-Based Access Control，基于角色的访问控制），就是用户通过角色与权限进行关联。简单地说，一个用户拥有若干角色，

Spring Cloud 服务和网关整合OAuth2实现权限控制实战全流程_spring clou...
官方迁移指南中说到OAuth 2.0客户端和资源服务器从Spring Security OAuth 2.x迁移到Spring Security 5.2.x。Spring Security不提供Authorization Server支持。更多详细的

基于Oauth2授权码模式的SSO单点登录+基于RBAC权限模型的动态路由的前...
实现oauth2协议的授权码模式 实现基于oauth2的SSO单点登录 实现基于RBAC权限模型的动态路由 3.技术选型 前端: vue+Element plus组件库 后端: springboot+mybatisp

基于角色的权限设计 菜鸟零号 kungfu~
基于角色的权限设计 问题提出 做互联网相关产品时，比较常见的就会遇到数据请求身份校验问题，需要判断当前用户有没有身份，该身份是否可以进行进一步的操作 虽说

基于Spring Boot、Spring Cloud & Alibaba、OAuth2、Flowable的分布式微服务架构
【标题】"基于Spring Boot、Spring Cloud & Alibaba、OAuth2、Flowable的分布式微服务架构"揭示了这个项目的核心技术栈，它是一个先进的企业级应用框架，旨在构建

开发笔记 | 认证授权+Spring Security+OAuth2快速学习笔记_customcode...
本文仅为个人学习笔记,通过简单的框架搭建来初步学习Spring Security及OAuth2,文中部分方法注解已过时,但仅为学习使用 基础概念 认证:认证用户的合法性,如登录 授权

Spring Boot、Spring Security、Oauth2实现的权限控制和认证服务_spr...
基于Spring Boot、Spring Security、Oauth2等实现的权限控制和认证服务、支持第三方oauth授权和获取资源信息功能等 一.简介 该项目是基于Spring Boot搭建而成,通过S

SpringSecurity+OAuth2+JWT分布式权限控制.zip
本项目“SpringSecurity+OAuth2+JWT分布式权限控制”旨在提供一个完整的解决方案，帮助开发者构建安全的、基于微服务的分布式应用程序。 Spring Security 是一个强

基于Spring Boot、Spring Cloud & Alibaba的分布式微服务架构权限管理系统
在构建基于Spring Boot、Spring Cloud & Alibaba的分布式微服务架构权限管理系统时，开发者通常会采用一系列先进的技术来实现高效、可扩展且灵活的系统。这个系统

java进阶开发，高级版web项目。基于dubbo实现分布式微服务架构，基于spring boot.zip
Java进阶开发与高级Web项目的实践，特别是基于Dubbo构建分布式微服务架构，以及结合Spring Boot的应用，是现代企业级开发的重要技术栈。这个压缩包文件"java03

基于Java的OAuth2和Shiro整合设计源码
本OAuth2和Shiro整合项目基于Java开发，包含191个文件，其中包括137个Java源文件、21个HTML文件、10个TXT文件、6个XML文件、5个Properties文件、2个Gitignc

微服务[学成在线] day18：基于oauth2实现RBAC认证授权、微服务间认证实现 通往体面生活的
???? 知识点概览 为了方便后续回顾该项目时能够清晰的知道本章节讲了哪些内容，并且能够从该章节的笔记中

基于角色的权限管理 Shatty的
权限管理里边涉及到权限如何高效方便的管理，RBAC（Role based access control）是现在比较流行的一种权限管理方案。最近公司也在做这方面的开发。这里对R

RBAC权限设计--基于角色权限控制 weixin_46084100的
角色分配权限

RBAC(基于角色的权限访问控制) wyy的
第一节.RBAC简介 英文全称(Role-Based Access Control) 中文全称:基于角色的权限访问控制 rbac: 一种数据库设计思想,根据设计数据库设计方案,完成项目的权限控制. 经

权限管理——基于角色的访问 信息技术提高班第
对于我们这块东西，一开始的时候没有想明白，于是重点就放错了。讨论了两天的权限角色，通过讨论的这两天对权限，角色也有了一些认识。 一、认识 权限，角

java分布式微服务架构搭建
Java分布式微服务架构是一种将大型应用程序拆分成多个小型、独立的服务的架构模式。每个服务都可以独立开发、部署和扩展，通过网络进行通信。下面是Java分布式

土味儿~
码龄3年 暂无认证

229	6729	38万+	36万+	
原创	周排名	总排名	访问	等级
2909	1万+	435	194	2109
积分	粉丝	获赞	评论	收藏

私信 已关注

搜博主文章

热门文章

分类专栏

最新评论

【图文详解】搭建 Spring Authorization …
笔墨画卷: 你这个是springcloud项目吗？前后端分离下的实现方案吗？不太理解这个 …

【图文详解】搭建 Spring Authorization …
emapjan: 感谢作者，我在一个节点卡主了，登录成功获取用户信息的时候，Aut …

【图文详解】搭建 Spring Authorization …
锦瑟-华年: 访问资源的时候只带了token并没有refreshtoken，确定可以自动刷新to …

Java NIO网络编程深入剖析【尚硅谷】
南风Candy: FileChannelDemo1 这个demo中 为什么没有写入的操作 还要进行读写 …

图解模块间7种耦合关系
m0_70295566: 大佬讲的太清晰了，能再讲讲内聚吗

最新文章

SpringBoot中java操作excel【EasyExcel】

Nacos的安装与使用（SpringCloud）

限流与令牌桶

2023年 19篇 2022年 126篇
2021年 84篇

目录