# 用户权限模块概要说明文档

V1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本信息** | **版本说明** | **编写者** | **修订者** | **审核** |
| V1.0 | 初始模块设计 | 戚羿辰  董双伟 | 戚羿辰  董双伟 | 顾文伟  叶闻达  胡加坚 |
| V1.1 | 更新用户模块登录流程 | 戚羿辰 | 戚羿辰 | 顾文伟  叶闻达  胡加坚 |

[用户权限模块概要说明文档 1](#_Toc26723)

[1.引言 5](#_Toc26155)

[1.1 编写目的 5](#_Toc18091)

[1.2 定义 5](#_Toc8533)

[2.需求分析 5](#_Toc16324)

[2.1 功能需求分析 5](#_Toc12387)

[2.2 业务需求分析 6](#_Toc13618)

[3.存储设计 6](#_Toc13167)

[3.1 数据存储设计 6](#_Toc8361)

[3.2 数据缓存设计 7](#_Toc14173)

[4.外部引用接口 8](#_Toc21903)

[5.运行环境说明 8](#_Toc7102)

[5.1 测试开发环境说明 8](#_Toc20432)

[5.2 部署配置说明 8](#_Toc9451)

[6.总体设计说明 9](#_Toc533)

[6.1 调用链说明 9](#_Toc979)

[6.2 权限模块结构 9](#_Toc26326)

[7.详细模块设计说明 10](#_Toc10571)

[8.问题说明 10](#_Toc29411)

[9版本更新说明 11](#_Toc10473)

[9.1 V1版本功能说明 11](#_Toc15704)

[9.2 V1版本升级说明 11](#_Toc20403)

[10. 接口调用说明 11](#_Toc4126)

[10.1 过滤功能 11](#_Toc29979)

[10.1.1 查询过滤列表 11](#_Toc27993)

[10.1.2 新增过滤 12](#_Toc301)

[10.1.3 修改过滤 12](#_Toc14717)

[10.1.4 删除过滤 13](#_Toc14549)

[10.1.5 查询过滤 13](#_Toc30435)

[10.2 获取令牌 13](#_Toc10661)

[10.3角色功能 14](#_Toc9995)

[10.3.1 查询角色列表 14](#_Toc6281)

[10.3.2 删除角色对应的权限 15](#_Toc18161)

[10.3.3 查询角色对应的权限 15](#_Toc13120)

[10.3.4 新增角色对应的权限 16](#_Toc18216)

[10.3.5 查询角色 16](#_Toc675)

[10.4 令牌操作相关 17](#_Toc9613)

[10.4.1 新增令牌类型 17](#_Toc19440)

[10.4.2 修改令牌类型 17](#_Toc15825)

[10.4.3 删除令牌类型 18](#_Toc16830)

[10.4.4 查询令牌类型 18](#_Toc20278)

[10.4.5 查询所有令牌 18](#_Toc10566)

[10.5 权限相关 19](#_Toc26914)

[10.5.1 查询权限列表 19](#_Toc32536)

[10.5.2 新增权限 20](#_Toc15748)

[10.5.3 修改权限 20](#_Toc16007)

[10.5.4 删除权限 20](#_Toc3865)

[10.5.5 查询权限 21](#_Toc18755)

[10.5.6 删除权限对应的资源 21](#_Toc25359)

[10.5.7 查询权限对应的资源 21](#_Toc6401)

[10.5.8 新增权限对应的资源 22](#_Toc6301)

[10.6 用户相关 23](#_Toc29470)

[10.6.1 查询用户列表 23](#_Toc24204)

[10.6.2 查询用户 24](#_Toc22175)

[10.6.3 用户登录 24](#_Toc32191)

[10.6.4 用户退出 25](#_Toc24324)

[10.7 资源相关 25](#_Toc26937)

[10.7.1 查询资源列表 25](#_Toc27324)

[10.7.2 查询资源 26](#_Toc18239)

[10.7.3 新增资源 27](#_Toc2627)

[10.7.4 删除资源 27](#_Toc1148)

[10.7.5 修改资源 27](#_Toc9525)

# 1.引言

## 1.1 编写目的

此文档使开发和测试人员了解用户权限模块整个业务流程.

# 1.2 定义

**用户**:包含app用户以及控制台的用户,数据完全从既有系统中获取.如app租户账号

**角色**:用户所对应的角色,用于区分用户的不同身份.一个用户可以同时含有多个角色,如1888888888[工人,材料供应商]

**权限**:单个角色所具有的权限.一个角色可以具有多个权限,同时单个权限可以分为增删改查的具体方法,使用请求方法进行区分.增删改查对应的方法为[post,delete,put,get].为方便权限控制和判断,建议开发过程严格按照该标准进行接口定义.

**资源**:各个系统中存在的可以访问的实际地址.后续版本将从注册中心拉取.目前由于注册中心暂缺,因而改为通过接口自行创建.

平台类型:用于区分用户和角色的来源,如控制台(console),普通登录用户(user)

**Filter**:权限系统中需要的忽略的请求,已后缀名进行判断,如.html

**AccessToken**:主要用于app用户的验证,包含时效性以及使用次数的属性.对于已经颁发的token无法修改其内容.该属性可以和资源绑定达到对指定接口的一次性校验

**未实现的部分**:本文档中已红色显示的部分为目前版本中未实现的部分

# 2.需求分析

## 2.1 功能需求分析

本模块为独立模块,api网关作为其上层入口,将符合规则的请求发至本模块,之后本模块对请求进行权限校验,如果校验未通过则返回错误,校验通过则将地址通过映射转发至真实地址.本模块的完整功能点为:

1. 实现对权限的细粒度拦截以及设置

2.对登录凭证token进行发放,设置与销毁

3.对部分请求进行一次性校验

4.集成oauth2授权方式（通过第三方微信，qq进行认证登录，或者是在认 证手机号上登录的劳务，商城app认证后允许登录）

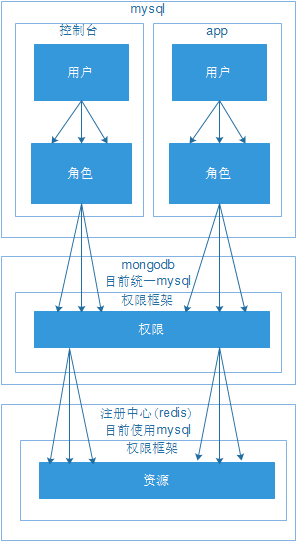
## 2.2 业务需求分析

该模块需要满足对当前即存系统的无缝插入.各个服务除了对于特定请求(如订单的一次性token申请)等,对于本系统不需额外发送请求.同时本系统将在既有系统之上存在独立的权限控制,如web控制台的额外过滤.

# 3.存储设计

## 3.1 数据存储设计

本系统的用户和角色以及其对应关系由即存模块中获取,资源由注册中心获取(目前由本系统录入).权限和资源以及角色的对应关系也由本系统创建.由于需要录入保存资源,为方便目前的修改(直接通过数据库),目前将数据存储在mysql中.注册中心模型建立后,除了既有用户和角色外,其余内容将存储于mongodb中.具体关系如下图所示:



## 3.2 数据缓存设计

为方便后续集群部署的支持,目前所有权限相关的信息在初次读取后都通过缓存redis的方法来进行后续的处理.

# shiro介绍

## 3.1 什么是shiro

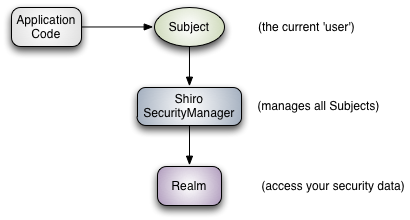
Apache Shiro是Java的一个安全框架。目前，使用Apache Shiro的人越来越多，因为它相当简单，对比Spring Security，可能没有Spring Security做的功能强大，但是在实际工作时可能并不需要那么复杂的东西，所以使用小而简单的Shiro就足够了。

Shiro可以非常容易的开发出足够好的应用，其不仅可以用在JavaSE环境，也可以用在JavaEE环境。Shiro可以帮助我们完成：认证、授权、加密、会话管理、与Web集成、缓存等。这不就是我们想要的嘛，而且Shiro的API也是非常简单

## 3.2 shiro相关定义

概览

在概念层，shiro架构包含三个主要的理念：Subject,SecurityManager和Realm。下面的图展示了这些组件如何相互作用，我们将在下面依次对其进行描述。



**subject**:就像我们在上一章示例中提到的那样，subject本质上是当前运行用户特定的'view'，而单词“user”经常暗指一个人，subject可以是一个人，但也可以是第三方服务、守护进程帐户、时钟守护任务或者其它--当前和软件交互的任何事件。

subject实例都和（也需要）一个SecurityManager绑定，当你和一个subject进行交互，这些交互动作被转换成SecurityManager下subject特定的交互动作。

**SecurityManager**:SecurityManager是Shiro架构的核心，配合内部安全组件共同组成安全伞。然而，一旦一个程序配置好了SecurityManager和它的内部对象，SecurityManager通常独自留下来，程序开发人员几乎花费的所有时间都集中在Subjet API上。

我们将在以后详细讨论SecurityManager，但当你和一个Subject互动时了解它是很重要的。任何Subject的安全操作中SecurityManager是幕后真正的举重者，这在上面的图表中可以反映出来。

**Realms**：Reamls是Shiro和你的程序安全数据之间的“桥”或者“连接”，它用来实际和安全相关的数据如用户执行身份认证（登录）的帐号和授权（访问控制）进行交互，shiro从一个或多个程序配置的Realm中查找这些东西。

**Realm**本质上是一个特定的安全DAO：它封装与数据源连接的细节，得到shiro所需的相关的数据。在配置shiro的时候，你必须指定至少一个Realm来实现认证（authentication）和/或授权（authorization）。SecurityManager可以配置多个复杂的Realm，但是至少有一个是需要的。

Shiro提供out-of-the-box Realms来连接安全数据源（或叫地址）如LDAP、JDBC、文件配置如INI和属性文件等，如果已有的Realm不能满足你的需求你也可以开发自己的Realm实现。

和其它内部组件一样，ShiroSecurityManager管理如何使用Realms获取Subject实例所代表的安全和身份信息。

## 3.3 shiro结构

下面的图表展示了Shiro的核心架构思想，下面有简单的解释。



**Subject** (org.apache.shiro.subject.Subject)

正在与软件交互的一个特定的实体“view”（用户、第三方服务、时钟守护任务等）。

**SecurityManager**(org.apache.shiro.mgt.SecurityManager)

如同上面提到的，SecurityManager 是Shiro的核心，它基本上就是一把“伞”用来协调它管理的组件使之平稳地一起工作，它也管理着Shiro中每一个程序用户的视图，所以它知道每个用户如何执行安全操作。

**Authenticator**(org.apache.shiro.authc.Authenticator)

Authenticator是一个组件，负责执行和反馈用户的认证（登录），如果一个用户尝试登录，Authenticator就开始执行。Authenticator知道如何协调一个或多个保存有相关用户/帐号信息的Realm，从这些Realm中获取这些数据来验证用户的身份以确保用户确实是其表述的那个人。

**Authentication Strategy**(org.apache.shiro.authc.pam.AuthenticationStrategy)

如果配置了多个Realm，AuthenticationStrategy将会协调Realm确定在一个身份验证成功或失败的条件（例如，如果在一个方面验证成功了但其他失败了，这次尝试是成功的吗？是不是需要所有方面的验证都成功？还是只需要第一个？）

**Authorizer**(org.apache.shiro.authz.Authorizer)

Authorizer是负责程序中用户访问控制的组件，它是最终判断一个用户是否允许做某件事的途径，像Authenticator一样，Authorizer也知道如何通过协调多种后台数据源来访问角色和权限信息，Authorizer利用这些信息来准确判断一个用户是否可以执行给定的动作。

**SessionManager**(org.apache.shiro.session.mgt.SessionManager)

SessionManager知道如何创建并管理用户Session生命周期而在所有环境中为用户提供一个强有力的Session体验。这在安全框架领域是独一无二--Shiro具备管理在任何环境下管理用户Session的能力，即使没有Web/Servlet或者EJB容器。默认情况下，Shiro将使用现有的session（如Servlet Container），但如果环境中没有，比如在一个独立的程序或非web环境中，它将使用它自己建立的session提供相同的作用，sessionDAO用来使用任何数据源使session持久化。

**SessionDAO**(org.apache.shiro.session.mgt.eis.SessionDAO)

SessionDAO代表SessionManager执行Session持久（CRUD）动作，它允许任何存储的数据挂接到session管理基础上。

**CacheManager**(org.apache.shiro.cache.CacheManager)

CacheManager创建并管理其它shiro组件的catch实例生命周期，因为shiro要访问许多后端数据源来实现认证、授权和session管理，caching已经成为提升性能的一流的框架特征，任何一个现在开源的和/或企业级的caching产品都可以插入到shiro中实现一个快速而有效的用户体验。

**Cryptography** (org.apache.shiro.crypto.\*)

Cryptography在安全框架中是一个自然的附加产物，shiro的crypto包包含了易用且易懂的加密方式，Hashes（亦即digests）和不同的编码实现。该包里所有的类都亦于理解和使用，曾经用过Java自身的加密支持的人都知道那是一个具有挑战性的工作，而shiro的加密API简化了java复杂的工作方式，将加密变得易用。

**Realms** (org.apache.shiro.realm.Realm)

如同上面提到的，Realm是shiro和你的应用程序安全数据之间的“桥”或“连接”，当实际要与安全相关的数据进行交互如用户执行身份认证（登录）和授权验证（访问控制）时，shiro从程序配置的一个或多个Realm中查找这些数据，你需要配置多少个Realm便可配置多少个Realm（通常一个数据源一个），shiro将会在认证和授权中协调它们。

**SecurityManager**

因为shiro API鼓励以Subject为中心的开发方式，大部分开发人员将很少会和SecurityManager直接交互（尽管框架开发人员也许发现它非常有用），尽管如此，知道SecurityManager如何工作，特别是当在一个程序中进行配置的时候，是非常重要的。

# 5.运行环境说明

## 5.1 测试开发环境说明

为方便后续的统一测试和部署，本项目将在以下环境中运行。

服务器地址：192.168.59.3

自动化部署：Jenkins

负载均衡：nginx

内网测试地址: http://192.168.59.3:8091/Authority/swagger-ui.html

## 5.2 部署配置说明

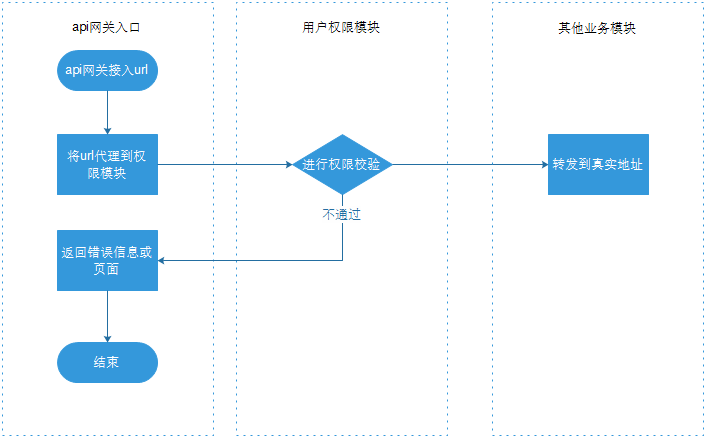
本模块需要修改的配置统一统一放在resource/application.yml中.

其中可以对数据源,端口,以及服务名称进行配置

# 6.总体设计说明

## 6.1 调用链说明

本模块由上层api网关作为入口,到本模块后进行权限校验等操作,并根据权限通过情况进行转发.具体如下图所示

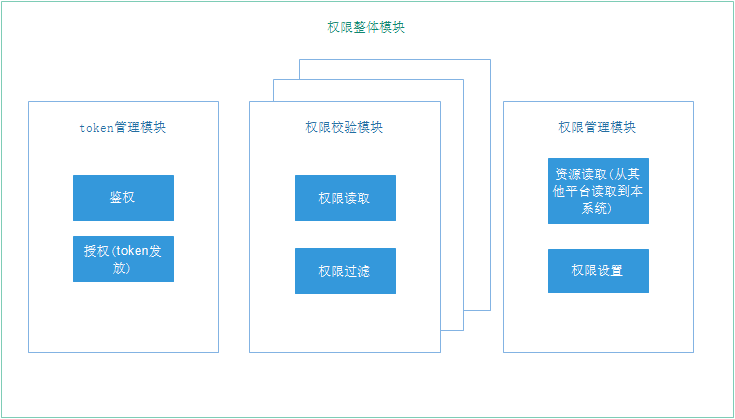


## 6.2 权限模块结构

权限模块整体可分为

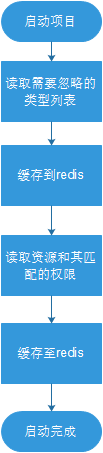
1. **token证书发放模块**:该模块后续可为oauth模块提供token,使用频率中
2. **权限设置模块**:设置模块,设置完成后即使服务宕机也不影响权限校验,使用频率低
3. **权限校验模块**:权限过滤的主要功能,提供权限校验过滤功能,使用频率高

目前为方便测试部署,暂为单一服务.具体如下图所示



### 6.3 项目整体运行流程

项目启动后,将从数据库读取需要忽略的类型列表,以及资源和其需要的权限列表,并进行设置.其内容均存在于redis中.之后则在请求时交友权限校验部分进行处理.项目启动流程如下图所示:



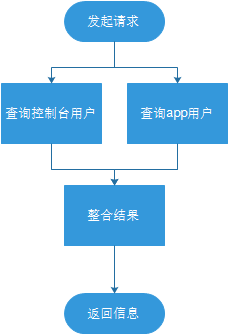
# 详细模块设计说明

**数据库结构:**

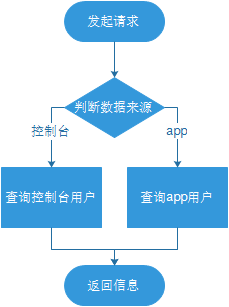
## 7.1 用户模块设计说明

本系统的用户目前分为控制台用户和app用户.原有系统中已存在用户以及其对应的角色关系,本系统仅负责读取其资源,而不提供设置的功能.其流程如下:

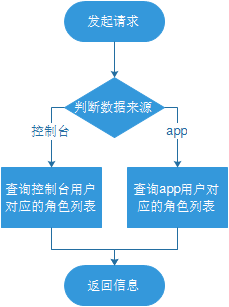
### 7.1.1查询用户列表:



### 7.1.2查询单个用户:



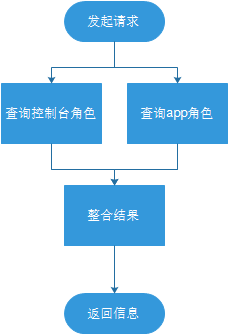
**7.1.3查询用户对应的角色:**



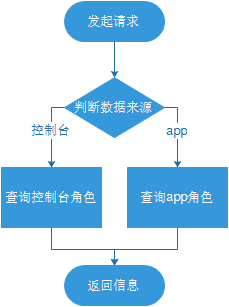
## 7.2 角色模块设计说明

本系统中所有角色均由从现有数据中生成,对于角色本系统不提供编辑功能.本系统仅维护角色所对应的权限.其流程如下:

### 7.2.1查询角色列表:

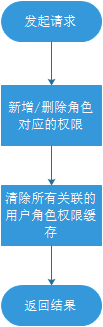


### 7.2.2查询单个角色:



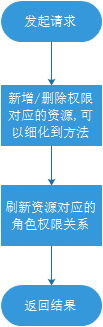
### 7.2.3 绑定角色和权限关系

在绑定或删除角色的权限关系后,需要对所有存在关系的在线用户对应的角色进行清除,用来刷新权限.权限清除后,在下次发起访问时将重新读取权限.



## 7.3 权限模块设计说明

权限信息由于既有系统中不存在,因此单独在本系统中进行维护.本模块包含了基本的权限信息的增删改查,以及提供权限对资源的绑定功能.另外,如果需要对同一个权限进行细致的增删改查权限区分,可以通过指定[read,create,update,delete]进行细致处理.最基本的增删改查这里不再赘述.下面就绑定权限资源的流程进行说明

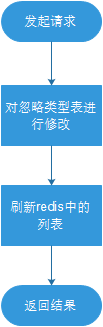


## 7.4 资源模块设计说明

资源模块目前由本系统进行编辑和维护.操作仅为最基本的单表增删改查,这里不再赘述.对应表结构详见本章最初部分.

## 7.5 过滤类型模块设计说明

该模块提供忽略类型的增删改查,用于对特定类型的请求进行忽略.在该模块进行修改后,需要对redis中的列表进行同步刷新,其流程如下:



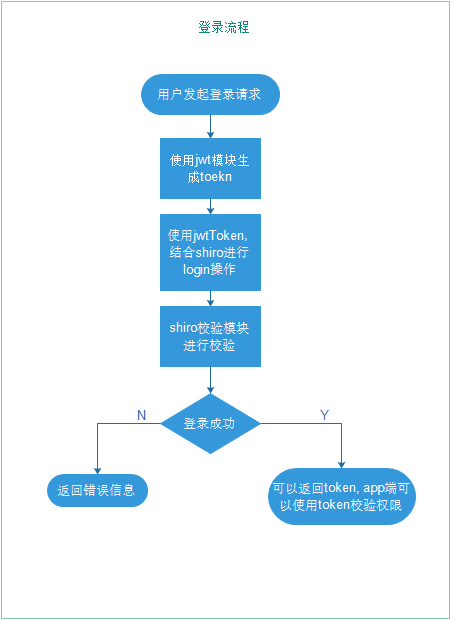
## 7.6 jwt模块设计说明

该模块提供token的生成和校验功能.token使用jwt生成,输入客户端ID,用户名等信息并返回字符串.

需要校验时,直接对该token进行解析即可完成校验.

## 7.7 登录模块设计说明

由于需要调用shiro框架的登录方法,因此先前的登录操作必须同时调用本系统的登录接口.后续可以替换为自动登录,即需要对各模块的登录接口做转发时,如果转发后获取的结果为成功,则自动获取其参数进行登录.登录的流程如下:

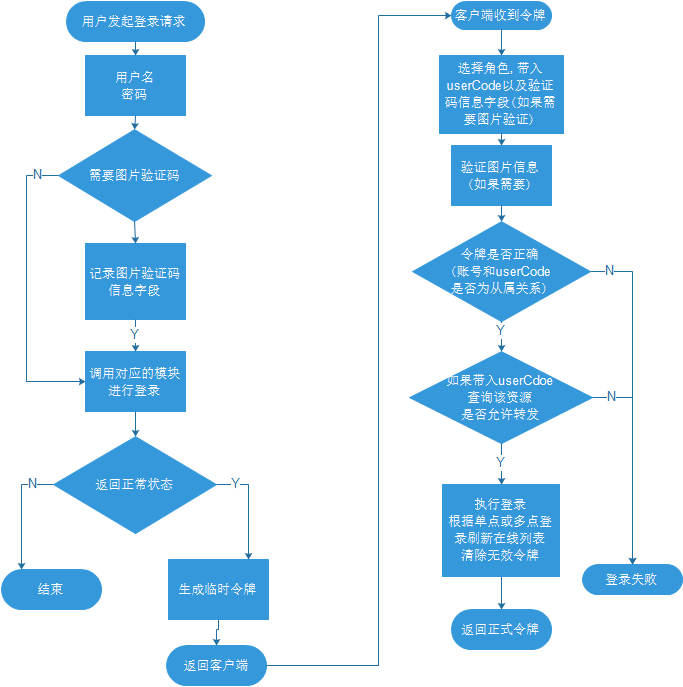


### 7.7.1 登录模块版本变动

权限模块的登录功能需求发生了变动.变动点如下:

1. 需要在选择角色后才正式登录,因此登录流程修改为2步,密码和账号校验通过后,会颁发一个临时的令牌.客户使用该令牌配合userCode来正式登录.
2. 登录时增加图片验证码校验功能
3. 登录时增加单点或多点登录设置(条件:用户,应用,角色,登录平台,默认单点)
4. 用户登录后添加用户在线状态以及登录的客户端列表
5. 登录时根据参数来判断userCode的允许通过规则.

其流程图更新为如下:

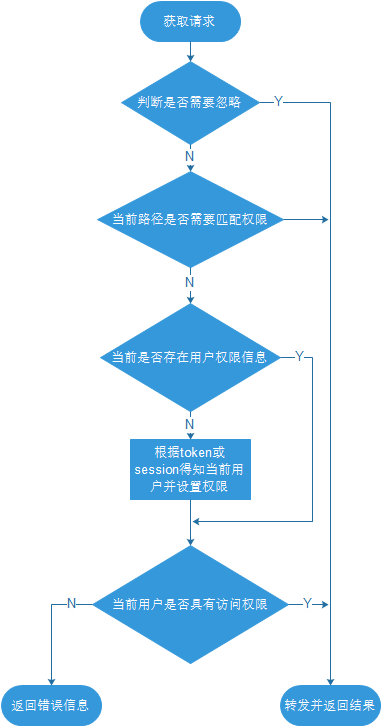


## 7.8 权限校验模块设计说明

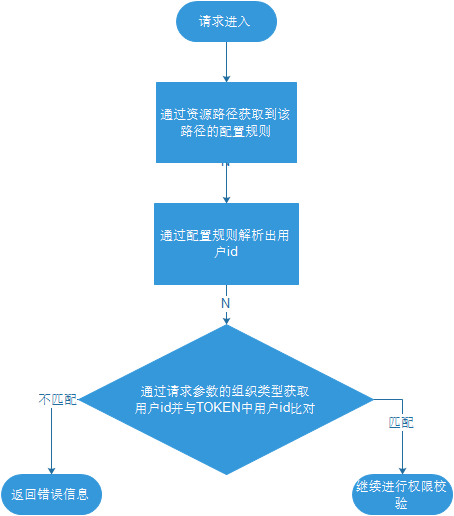
### 7.8.1 用户资源权限校验

权限校验不是单独的接口,而是贯穿整个系统的功能.所有接收的请求都会进行权限校验.

校验时,首先会对请求类型进行过滤,忽略的请求直接进行转发,需要过滤的请求则会根据来源进行校验.用户登录后初次请求时,会根据session或token得知当前的用户,并对其权限进行设置并缓存.之后进行权限校验,查询用户是否具有访问该路径的权限.其具体流程如下:



### 7.8.2用户身份校验

来自客户端的用户请求参数，在收集和传输过程当中如果被篡改，那么对整个公司的系统安全性和用户的信息安全性都将造成严重的后果，所以本模块开发的意义就是在请求到达后端真实服务之前，对用户的请求URL进行解析和验证，判断出URL是否按照固定的配置进行组织，判断URL当中的参数是否跟当前登录用户TOKEN当中存储的用户信息一致，如果一致，则可以进行下一步权限验证，如果不一致则拒绝访问。具体流程包括如下几个:

# 8.问题说明

1. 目前web端基于session,常规操作即使token过期也不需要登录,app端则需要重新申请.该模式是否合理

2. 为严格控制增删改查权限,所有请求尽量符合restful规范

3. 目前被调用方(转发时), put等类型如果不使用@requestbody注解,则可能存在swagger可以直接访问,但是转发时无法解析的情况

# 9版本更新说明

## 9.1 V1版本功能说明

|  |  |
| --- | --- |
| **接口功能** | **说明** |
| [过滤功能](#_10.1 过滤功能) | 对后缀类型实现增删改查,用于对指定内容忽略权限 |
| [获取令牌](#_10.2 获取令牌) | 获取jwt模块生成的令牌 |
| [角色功能](#_10.3角色功能) | 对其他模块的角色进行提取,同时提供角色对权限的增删查功能 |
| [用户功能](#_10.6 用户相关) | 对其他模块的用户进行提取,同时提供用户的登录登出接口 |
| [令牌类型功能](#_10.4 令牌操作相关) | 提供令牌类型的增删改查功能,令牌主要包含类型,时效,次数属性 |
| [资源功能](#_10.7 资源相关) | 对资源进行管理,以及资源对应的token类型的绑定功能 |
| [权限功能](#_10.5 权限相关) | 对权限进行增删改查,同时提供权限对资源的绑定功能 |

## 9.2 后续版本升级说明

|  |  |
| --- | --- |
| **接口功能** | **说明** |
| Oatuh2模块 | 添加oauth2模块的支持 |
| 资源动态拉取 | 将资源的获取方式由手动输入修改为从注册中心(redis)动态拉取 |
| 存储方式修改 | 资源不需要手动录入后,将存储方式修改为mongo |
| 支持动态配置 | 项目注册到注册中心实现负载均衡,配置动态化 |

# 接口调用说明

内网测试地址

[http://192.168.59.3:8091/Authority/swagger-ui.html](http://localhost:8091/Authority/swagger-ui.html)

Host

http://192.168.59.3:8091

## 10.1 过滤功能

### 10.1.1 查询过滤列表

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resourceFilters | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": [  {  "id": 1,  "filterType": ".jpg"  }  ]  } | | |

### 10.1.2 新增过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resourceFilters | | |
| **请求方式** | Post | **协议** | http |
| **Header** |  | | |
| **Body** | filterType:string // 需要忽略的类型 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": 16  } | | |

### 10.1.3 修改过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resourceFilters | | |
| **请求方式** | Put | **协议** | http |
| **Header** |  | | |
| **Body** | {  "filterType": ".string", // 需要忽略的类型  "id": 16 // 记录id  } | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.1.4 删除过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resourceFilters/{id} | | |
| **请求方式** | Delete | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.1.5 查询过滤

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resourceFilters/{id} | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "id": 1,  "filterType": ".jpg"  }  } | | |

## 10.2 获取令牌

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/oauth/token | | |
| **请求方式** | Post | **协议** | http |
| **Header** |  | | |
| **Body** | {  "password": "string", // 密码  "platformId": "string", // 平台类型  "redirectUri": "string", // url  "userName": "string" // 用户名  } | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": "{access\_token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiMTU3NTcxMjk0NzEjdXNlciIsInBhc3N3b3JkIjoiYmU1NmUwNTQ5YmE1OWFmMjBmODgzZTEwYWRjMyIsImNoZWNrVGltZXMiOjAsImV4cCI6MTQ4ODI2NzQzMywibmJmIjoxNDg4MDk0NjMzfQ.qlMjqQzYjwQRI6GzU4rHZ76M1GSrqnJ0l4\_3NQMl6Gw, token\_type=bearer, expires\_in=172800}"  } | | |

## 10.3角色功能

### 10.3.1 查询角色列表

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/role | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** | roleName:string // 角色名称  platType:string // 平台类型  pageSize:int // 每页大小  pageNo:int // 页码 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "list": [  {  "id": 42,  "roleName": "超级管理员",  "platType": "console",  "permissionList": null,  "stringPermissions": null  }  ],  "orderBy": null,  "pageNumber": 1,  "pageSize": 10,  "totalPage": 3,  "totalRow": 26,  "firstPage": true,  "lastPage": false  }  } | | |

### 10.3.2 删除角色对应的权限

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/role/permission | | |
| **请求方式** | delete | **协议** | http |
| **Header** |  | | |
| **Body** | roleName:string // 角色名称  platType:string // 平台类型  permissionId:string // 角色id  method:string // 权限类型 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.3.3 查询角色对应的权限

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/role/permission | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** | roleId:int // 角色id  platType:string // 平台类型 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": [  {  "id": 3,  "permission": "user",  "permissionCode": "aaaaa",  "method": null  },  {  "id": 3,  "permission": "user",  "permissionCode": "aaaaa",  "method": null  }  ]  } | | |

### 10.3.4 新增角色对应的权限

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/role/permission | | |
| **请求方式** | post | **协议** | http |
| **Header** |  | | |
| **Body** | roleName:string // 角色名称  platType:string // 平台类型  permissionId:string // 角色id  method:string // 权限类型 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.3.5 查询角色

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/role/{id}/{playType} | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "id": 1,  "roleName": "工人",  "platType": "user",  "permissionList": [],  "stringPermissions": null  }  } | | |

## 10.4 令牌操作相关

### 10.4.1 新增令牌类型

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/token/type | | |
| **请求方式** | post | **协议** | http |
| **Header** |  | | |
| **Body** | {  checkTimes (integer, optional): token验证次数 0表示不限次数 ,  describe (string, optional): 描述 ,  expire (integer, optional): 超时时间默认2天 单位是毫秒 ,  id (integer, optional): 令牌类型id ,  name (string, optional): 令牌名称  } | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": 11  } | | |

### 10.4.2 修改令牌类型

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/token/type | | |
| **请求方式** | put | **协议** | http |
| **Header** |  | | |
| **Body** | {  checkTimes (integer, optional): token验证次数 0表示不限次数 ,  describe (string, optional): 描述 ,  expire (integer, optional): 超时时间默认2天 单位是毫秒 ,  id (integer, optional): 令牌类型id ,  name (string, optional): 令牌名称  } | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.4.3 删除令牌类型

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/token/type/{id} | | |
| **请求方式** | delete | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.4.4 查询令牌类型

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/token/type/{1} | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "id": 1,  "checkTimes": 0,  "expire": 172800000,  "describe": "用来测试的token",  "name": "测试token"  }  } | | |

### 10.4.5 查询所有令牌

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/token/type | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": [  {  "id": 1,  "checkTimes": 0,  "expire": 172800000,  "describe": "用来测试的token",  "name": "测试token"  }  ]  } | | |

## 10.5 权限相关

### 10.5.1 查询权限列表

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** | permission: string // 权限名  permissionCode:string // 权限编码  pageSize:int // 行数  pageNo:int // 页数 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "list": [  {  "id": 2,  "permission": "ccc",  "permissionCode": "ccc",  "method": null  }  ],  "orderBy": null,  "pageNumber": 1,  "pageSize": 10,  "totalPage": 1,  "totalRow": 4,  "firstPage": true,  "lastPage": true  }  } | | |

### 10.5.2 新增权限

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission | | |
| **请求方式** | post | **协议** | http |
| **Header** | permission: string // 权限名  permissionCode:string // 权限编码 | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": 16  } | | |

### 10.5.3 修改权限

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission | | |
| **请求方式** | put | **协议** | http |
| **Header** |  | | |
| **Body** | permission: string // 权限名  permissionCode:string // 权限编码  Id:int // 记录id | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.5.4 删除权限

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission/{id} | | |
| **请求方式** | delete | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.5.5 查询权限

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission/{id} | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "id": 2,  "permission": "ccc",  "permissionCode": "ccc",  "method": null  }  } | | |

### 10.5.6 删除权限对应的资源

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission/resource | | |
| **请求方式** | delete | **协议** | http |
| **Header** |  | | |
| **Body** | resource:string // 资源地址  permissionId:int // 权限id | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.5.7 查询权限对应的资源

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission/resource | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** | resource:string // 资源地址  permissionId:int // 权限id  pageSize:int // 行数  pageNo:int // 页数 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "list": [  {  "id": 3, “rule”:{type:”1”,”position”:3,”paramname”:”userid”,”attributename”:”userid”},  "resourcePath": "/user/aaaaa",  "tokenTypeId": 2,  "tokenType": null,  "permStr": null,  "tails": {  "resource": "/user/aaaaa",  "permissionId": 3  }  },  ],  "orderBy": null,  "pageNumber": 1,  "pageSize": 10,  "totalPage": 1,  "totalRow": 3,  "firstPage": true,  "lastPage": true  }  } | | |
|  |  | | |

### 10.5.8 新增权限对应的资源

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/permission/resource | | |
| **请求方式** | post | **协议** | http |
| **Header** |  | | |
| **Body** | resource:string // 资源地址  rule:string //资源url配置规则  permissionId:int // 权限id  pageSize:int // 行数  pageNo:int // 页数 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

## 10.6 用户相关

### 10.6.1 查询用户列表

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/user | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** | username:string // 登录用户名  platType:stirng // 平台类型 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "list": [  {  "id": 1,  "password": "be56e0549ba59af20f883e10adc3",  "phone": null,  "username": "yhyssgys:001",  "salt": null,  "platType": "user",  "roles": null  }  ],  "orderBy": null,  "pageNumber": 1,  "pageSize": 10,  "totalPage": 8,  "totalRow": 77,  "firstPage": true,  "lastPage": false  }  } | | |

### 10.6.2 查询用户

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/user/{id}/{platType} | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** | Id:int // 用户id  platType:string // 平台类型 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "id": 10,  "password": "be56e0549ba59af20f883e10adc3",  "phone": "15757129471",  "username": "15757129471",  "salt": null,  "platType": "user",  "roles": [  {  "id": 3,  "roleName": "项目经理",  "platType": "user",  "permissionList": null,  "stringPermissions": null  }  ]  }  } | | |

### 10.6.3 用户登录

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/user/login | | |
| **请求方式** | Post | **协议** | http |
| **Header** |  | | |
| **Body** | username:string // 用户名  password:string // 密码  platType:string // 平台类型 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": "{access\_token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJuYW1lIjoiMTU3NTcxMjk0NzEjdXNlciIsInBhc3N3b3JkIjoiYmU1NmUwNTQ5YmE1OWFmMjBmODgzZTEwYWRjMyIsImNoZWNrVGltZXMiOjAsImV4cCI6MTQ4ODI2NzQzMywibmJmIjoxNDg4MDk0NjMzfQ.qlMjqQzYjwQRI6GzU4rHZ76M1GSrqnJ0l4\_3NQMl6Gw, token\_type=bearer, expires\_in=172800}"  } | | |

### 10.6.4 用户退出

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/user/logout | | |
| **请求方式** | get | **协议** | http |
| **Header** | accessToken | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

## 10.7 资源相关

### 10.7.1 查询资源列表

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resource | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** | resourcePath:string // 资源地址  tokenTypeId:int // token类型  pageSize:int // 行数  pageNo:int // 页数 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "list": [  {  "id": 1,  "resourcePath": "/user/login",  "tokenTypeId": 0,  "tokenType": null,  "permStr": null,  "tails": {}  }  ],  "orderBy": null,  "pageNumber": 1,  "pageSize": 10,  "totalPage": 1,  "totalRow": 6,  "firstPage": true,  "lastPage": true  }  } | | |

### 10.7.2 查询资源

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resource/{id} | | |
| **请求方式** | get | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": {  "id": 1,  "resourcePath": "/user/login",  "tokenTypeId": 0,  "tokenType": null,  "permStr": null,  "tails": {}  }  } | | |

### 10.7.3 新增资源

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resource | | |
| **请求方式** | Post | **协议** | http |
| **Header** |  | | |
| **Body** | resourcePath:string // 资源地址  tokenTypeId:int // token类型  allowResetHeader:int // 是否允许转发userCode  default\_permission:string // 默认权限 | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": 16  } | | |

### 10.7.4 删除资源

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resource/{id} | | |
| **请求方式** | Delete | **协议** | http |
| **Header** |  | | |
| **Body** |  | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |

### 10.7.5 修改资源

|  |  |  |  |
| --- | --- | --- | --- |
| **请求地址** | Host/Authority/resource | | |
| **请求方式** | Put | **协议** | http |
| **Header** |  | | |
| **Body** | resourcePath:string // 资源地址  tokenTypeId:int // token类型  Id:int // 记录id | | |
| **Response** | {  "status": 200,  "message": "SUCCESS",  "data": null  } | | |