目录

[一、 项目结构 1](#_Toc508308628)

[1. Eclipse中 1](#_Toc508308629)

[2. Idea中 2](#_Toc508308630)

[二、 模块和配置 2](#_Toc508308631)

[1. 父pom.xml 3](#_Toc508308632)

[2. 注册中心 4](#_Toc508308633)

[1) bootstrap.yml 4](#_Toc508308634)

[2) 启动注册中心 5](#_Toc508308635)

[3) 访问http://localhost:8761 5](#_Toc508308636)

[3. 配置中心 7](#_Toc508308637)

[1) bootstrap.yml 7](#_Toc508308638)

[2) {profile} 8](#_Toc508308639)

[3) 端口 9](#_Toc508308640)

[4) 注册到注册中心 9](#_Toc508308641)

[5) 注册中心里的显示 9](#_Toc508308642)

[4. 用户中心 10](#_Toc508308643)

[1) 数据库脚本 10](#_Toc508308644)

[2) bootstrap.yml 11](#_Toc508308645)

[3) user-center.yml 11](#_Toc508308646)

[a) 日志级别和文件配置 12](#_Toc508308647)

[b) 数据源配置 12](#_Toc508308648)

[c) Rabbitmq配置 12](#_Toc508308649)

[d) Mybatis配置 12](#_Toc508308650)

[4) 配置类 13](#_Toc508308651)

[a) 全局异常处理 13](#_Toc508308652)

[b) Rabbitmq的exchange声明 14](#_Toc508308653)

[c) 资源权限配置 14](#_Toc508308654)

[d) 密码加密处理器 15](#_Toc508308655)

[5. 认证中心 15](#_Toc508308656)

[1) 数据库脚本 16](#_Toc508308657)

[2) bootstrap.yml 16](#_Toc508308658)

[3) oauth-center.yml 16](#_Toc508308659)

[a) redis配置 16](#_Toc508308660)

[b) 数据库配置 17](#_Toc508308661)

[c) token是否用jwt 17](#_Toc508308662)

[4) 配置类 18](#_Toc508308663)

[a) 授权服务器配置 18](#_Toc508308664)

[b) 资源服务器 18](#_Toc508308665)

[c) Session共享配置 19](#_Toc508308666)

[6. 文件中心 20](#_Toc508308667)

[1) file-center.yml 21](#_Toc508308668)

[a) 数据库和mq 21](#_Toc508308669)

[b) 上传文件大小限制 21](#_Toc508308670)

[c) 自定义配置 21](#_Toc508308671)

[2) 配置类 22](#_Toc508308672)

[a) 加载jar包外文件 22](#_Toc508308673)

[b) 资源服务器 23](#_Toc508308674)

[7. 网关 23](#_Toc508308675)

[1) gateway-zuul.yml 24](#_Toc508308676)

[a) 路由规则 24](#_Toc508308677)

[b) 自定义参数 25](#_Toc508308678)

[2) 配置类 25](#_Toc508308679)

[a) 跨域配置 26](#_Toc508308680)

[b) 异常处理 26](#_Toc508308681)

[8. 日志组件log-starter 27](#_Toc508308682)

[1) spring.factories 27](#_Toc508308683)

[2) 使用该组件 28](#_Toc508308684)

[9. 日志中心 29](#_Toc508308685)

[1) log-center.yml 30](#_Toc508308686)

[2) 配置类 30](#_Toc508308687)

[a) 开启异步线程池 31](#_Toc508308688)

[b) 声明队列 31](#_Toc508308689)

[3) 处理日志消息 32](#_Toc508308690)

[10. 后台管理系统 33](#_Toc508308691)

[1) manage-backend.yml 34](#_Toc508308692)

[a) 数据库和mq 34](#_Toc508308693)

[b) 邮件配置 34](#_Toc508308694)

[2) 消息处理 35](#_Toc508308695)

[3) 静态资源 36](#_Toc508308696)

[11. 监控中心 37](#_Toc508308697)

[12. Swagger配置 37](#_Toc508308698)

[13. user-info-uri 38](#_Toc508308699)

[14. commons模块 39](#_Toc508308700)

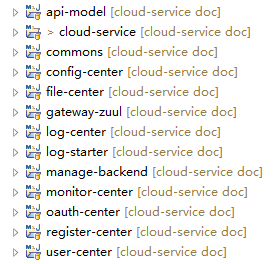
[1) PermitAllUrl 39](#_Toc508308701)

[2) AppUserUtil 40](#_Toc508308702)

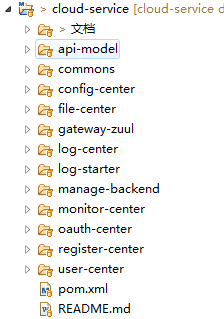
# 项目结构

导入项目后，在eclipse和idea下展示如下：

## Eclipse中

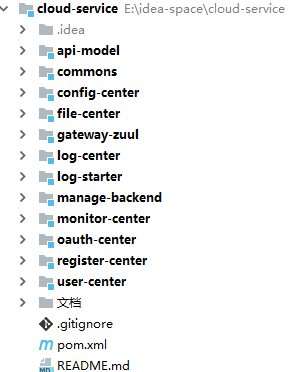


父pom在cloud-service中，文档也在cloud-service中，展开如下



## Idea中

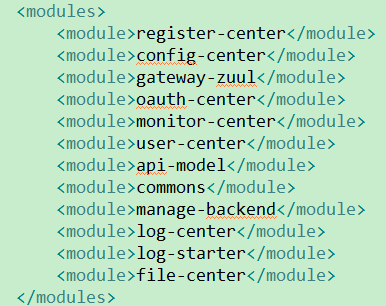
父pom已全部展开



# 模块和配置

|  |  |
| --- | --- |
| **工程名** | **说明** |
| api-model | 数据传输对象、常量等 |
| commons | 工具类、公共常量等 |
| config-center | 配置中心 |
| file-center | 文件中心 |
| gateway-zuul | 网关 |
| log-center | 日志中心 |
| log-starter | 日志组件，别的项目直接引入该依赖即可 |
| manage-backend | 后台管理 |
| monitor-center | 监控中心 |
| oauth-center | 认证中心 |
| register-center | 注册中心 |
| user-center | 用户中心、用户、角色、权限 |

## 父pom.xml

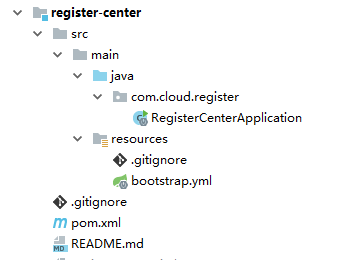


这里配置的各个模块目录

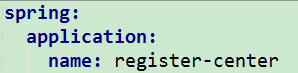


这里定义了一些jar包的版本号

## 注册中心



### bootstrap.yml

意思是给该应用起个名字

应用启动后的端口号

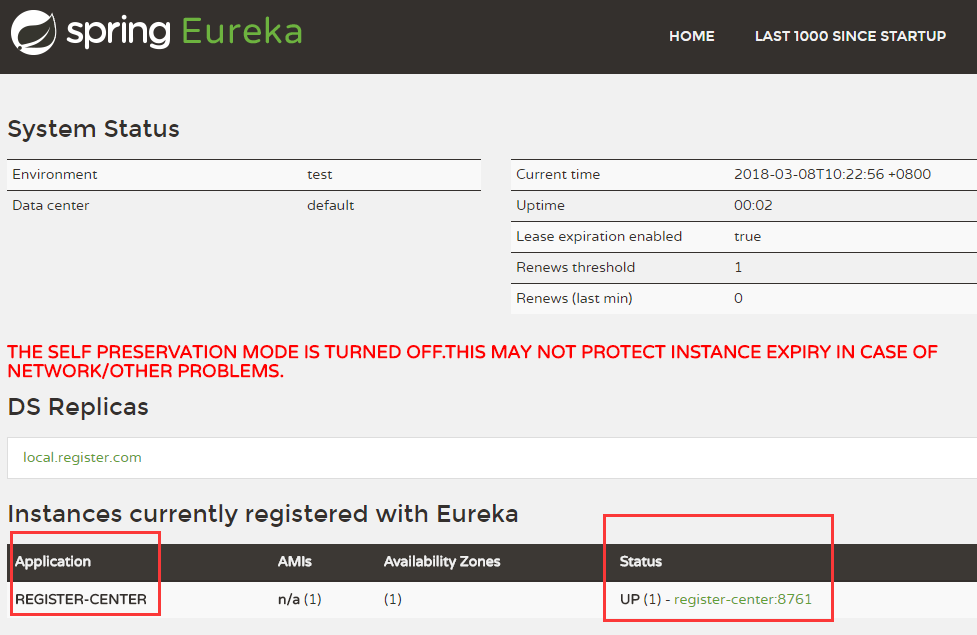
### 启动注册中心

直接运行下面类的main方法

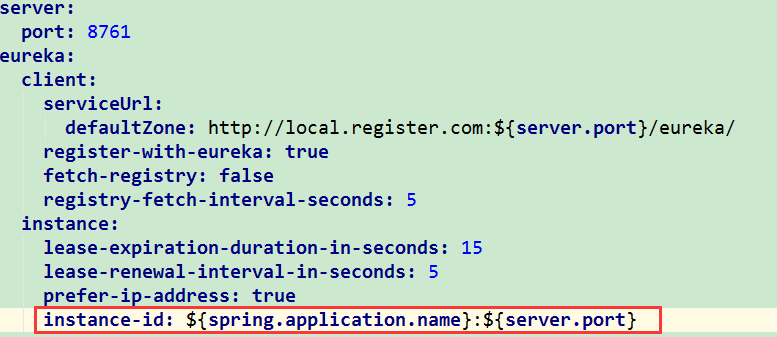
[com](eclipse-javadoc:%E2%98%82=register-center/src%5C/main%5C/java%3Ccom).[cloud](eclipse-javadoc:%E2%98%82=register-center/src%5C/main%5C/java%3Ccom.cloud).[register](eclipse-javadoc:%E2%98%82=register-center/src%5C/main%5C/java%3Ccom.cloud.register).RegisterCenterApplication

### 访问http://localhost:8761

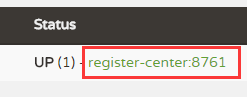
8761是bootstrap.yml里配置的系统端口号



Application列下的，REGISTER-CENTER就是我们在bootstrap.yml里指定的应用名，注册中心给我们大写处理了。

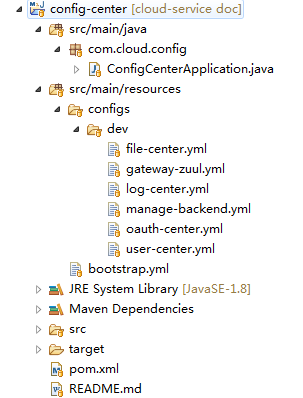
Status下显示的是我们在bootstrap.yml里的instance-id，如下图，我们这里是应用名加端口号

该配置为true的话，是用ip注册，否则是主机名注册，强烈建议配置为true

点击这里会跳转到类似如下的地址http://192.168.0.111:8761/info

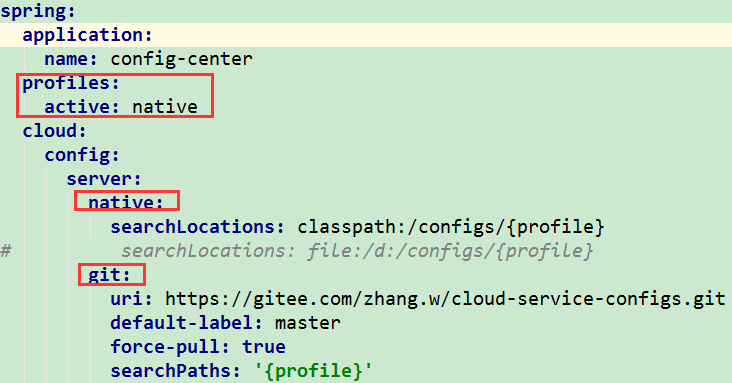
用这种方式，我们可以看到服务的具体ip地址和端口

## 配置中心



### bootstrap.yml

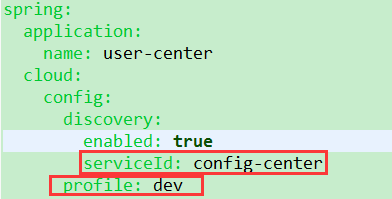
配置在本地或者git



通过spring.profiles.active这里可以指定配置文件在本地，还是classpath下还是在远程git上面，这里默认是放在了本地的classpath下，这里可根据实际项目需求和运维条件进行合理的选择配置方式。

### {profile}

上图配置中的{profile}，是由别的微服务指定的，如用户中心指定配置，用户中心里会有如下配置，



这里的profile: dev就会注入到

配置目录就成了classpath:/configs/dev

用户中心启动的时候，就会从配置中心拉取配置，目录就是

classpath:/configs/dev/user-center.yml

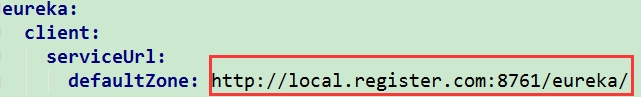
因此用户中心通过配置test或者production等等自定义的字符串，启动时会去找相应的配置，来达到分环境配置的目的，如



### 端口

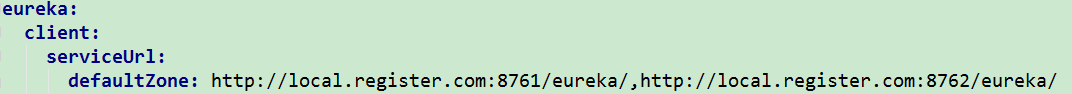
这里配置成0，启动时，项目会随机一个端口号

### 注册到注册中心

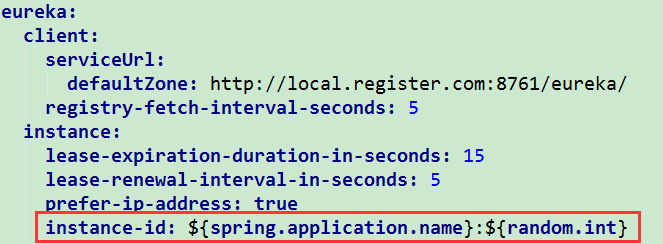


这里需要注意，地址后面有个/eureka/

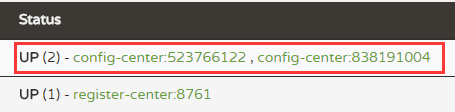
如果是多注册中心，那么通过逗号分隔



### 注册中心里的显示



因为我们是随机端口号，我们这里用了随机数字来显示

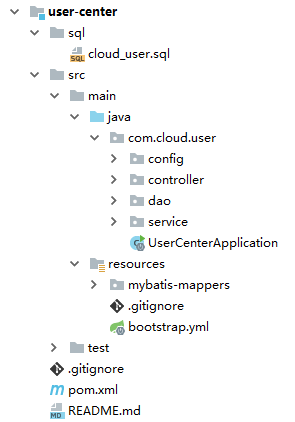


注意那个随机数字，并不是真正的端口号，点击跳转之后

http://192.168.0.104:53484/info

这里我们才能看到真正的端口号

## 用户中心



### 数据库脚本

在user-center模块下的sql文件夹下cloud\_user.sql里是用户中心的数据脚本，包含建表语句和初始化数据。

### bootstrap.yml

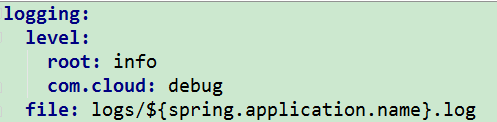


这里主要配置，配置中心的serviceId就是配置中心的spring.application.name，还有自己的profile，还有注册中心的url

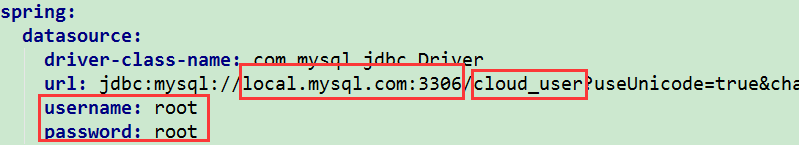
### user-center.yml

这里配置了用户系统具体的一些配置，比如数据库、mq、mybatis、日志级别等。

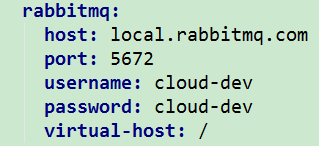
#### 日志级别和文件配置



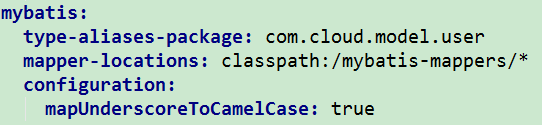
#### 数据源配置



#### Rabbitmq配置



#### Mybatis配置



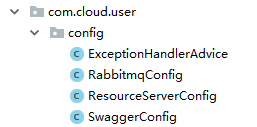
别名包有多个的话，逗号隔开，如下图



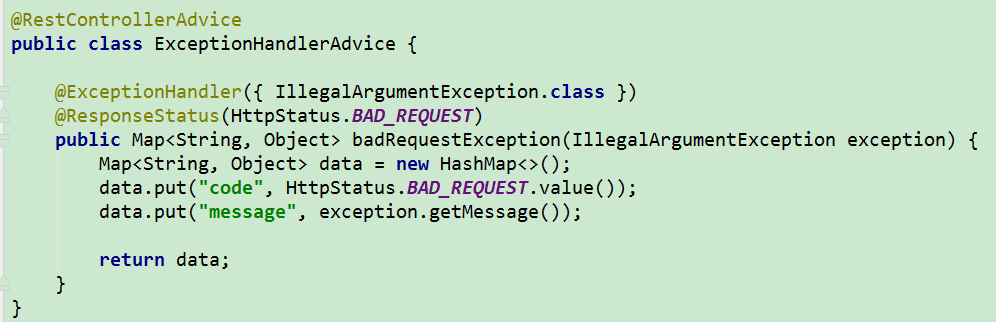
复杂sql写在mapper.xml文件里，在下图路径下



### 配置类

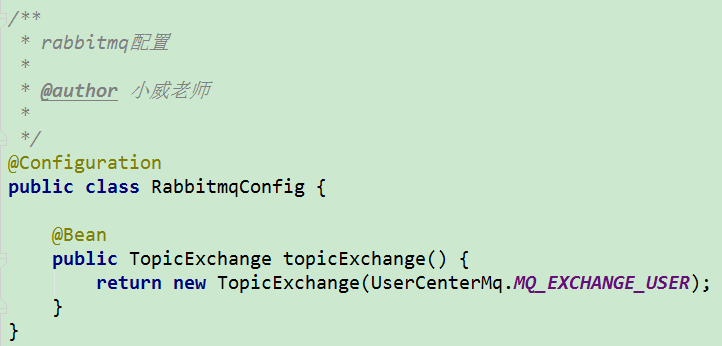


#### 全局异常处理



抛出[java](eclipse-javadoc:%E2%98%82=user-center/D:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_112%5C/jre%5C/lib%5C/rt.jar%3Cjava).[lang](eclipse-javadoc:%E2%98%82=user-center/D:%5C/Program%20Files%5C/Java%5C/jdk1.8.0_112%5C/jre%5C/lib%5C/rt.jar%3Cjava.lang).IllegalArgumentException异常的接口将返回http状态码400

#### Rabbitmq的exchange声明



这里声明一个topic类型的exchange，发消息时用。

#### 资源权限配置



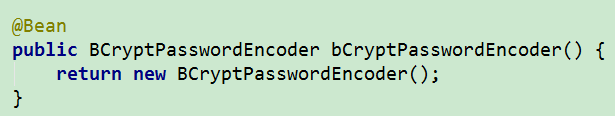
将我们的项目作为资源服务器

是启动权限注解支持



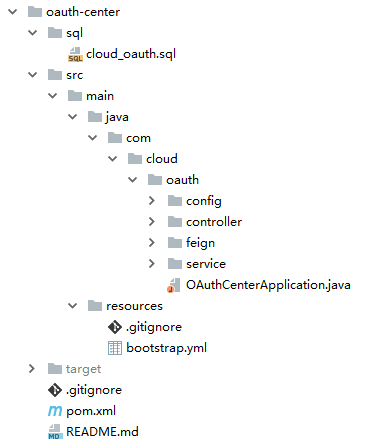
这里符合规则的url将不做权限拦截。

#### 密码加密处理器

上图的

声明一个密码加密和校验处理器Bean，该bean是spring security自带的。

## 认证中心



### 数据库脚本

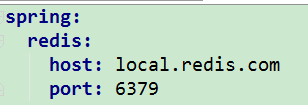
在oauth-center模块下的sql文件夹下cloud\_oauth.sql里是认证中心的数据脚本，包含建表语句和初始化数据。

### bootstrap.yml

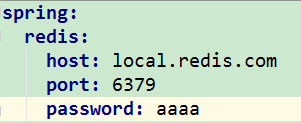
除spring.application.name是oauth-center外，其他如用户中心的bootstrap.yml相同

### oauth-center.yml

#### redis配置

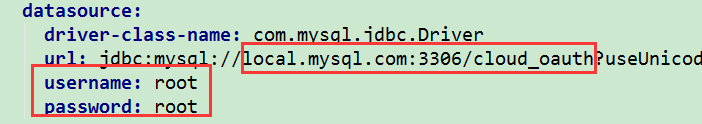


如redis有密码，与host同层级加节点password，如



注意password冒号后加一个空格

#### 数据库配置



#### token是否用jwt

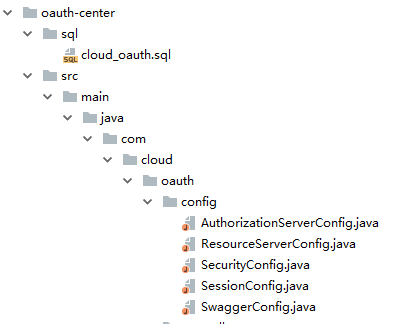


false的话token是默认的uuid，true的话token将采用jwt

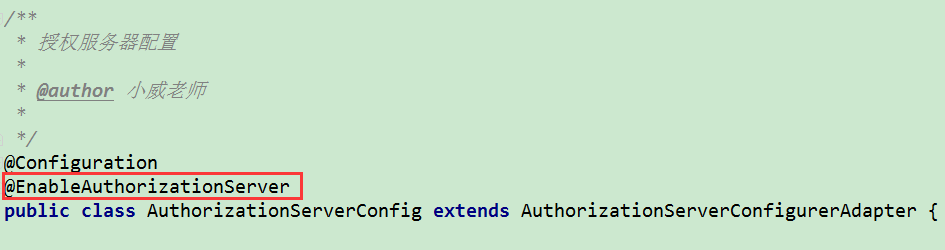
[com](eclipse-javadoc:%E2%98%82=oauth-center/src%5C/main%5C/java%3Ccom).[cloud](eclipse-javadoc:%E2%98%82=oauth-center/src%5C/main%5C/java%3Ccom.cloud).[oauth](eclipse-javadoc:%E2%98%82=oauth-center/src%5C/main%5C/java%3Ccom.cloud.oauth).[config](eclipse-javadoc:%E2%98%82=oauth-center/src%5C/main%5C/java%3Ccom.cloud.oauth.config).AuthorizationServerConfig里面



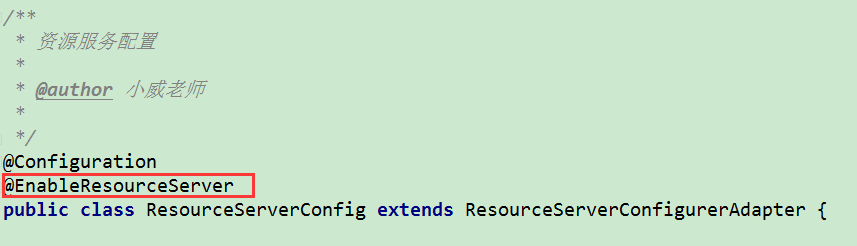
### 配置类



#### 授权服务器配置



#### 资源服务器



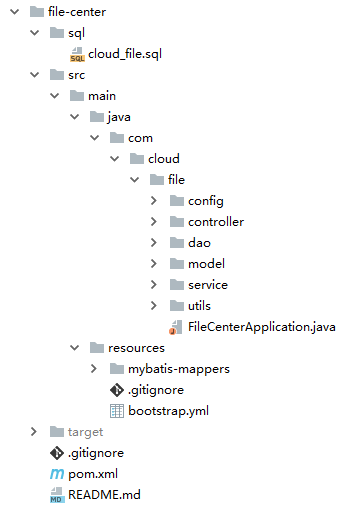
#### Session共享配置



用redis做session共享，在授权码模式下，可能会涉及参数code和state和redirect\_url的传递，多台服务器下需要共享session。

目前该项目没用授权码模式，此处不设置也没问题。

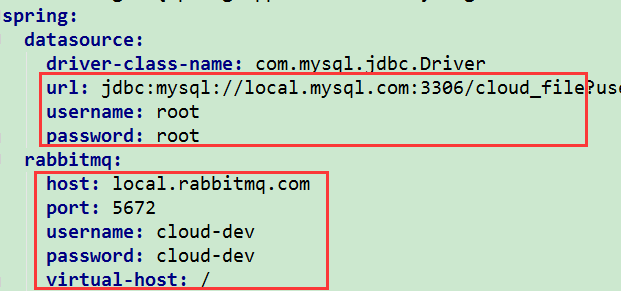
## 文件中心



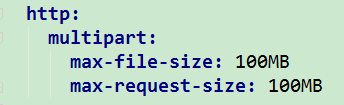
bootstrap.yml里spring.application.name为file-center其余跟用户中心的一样

### file-center.yml

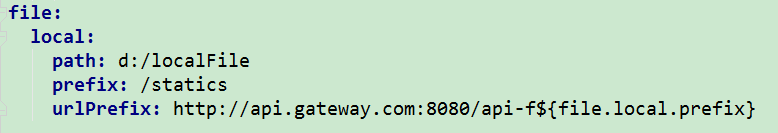
#### 数据库和mq



#### 上传文件大小限制



#### 自定义配置



上图path是上传文件存储根路径

上图prefix是前缀

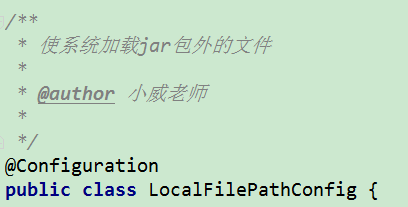
上图urlPrefix是域名加前缀

如d:/localFile/aaa.png用url访问就是

http://api.gateway.com:8080/api-f/statics/aaa.png

### 配置类

#### 加载jar包外文件

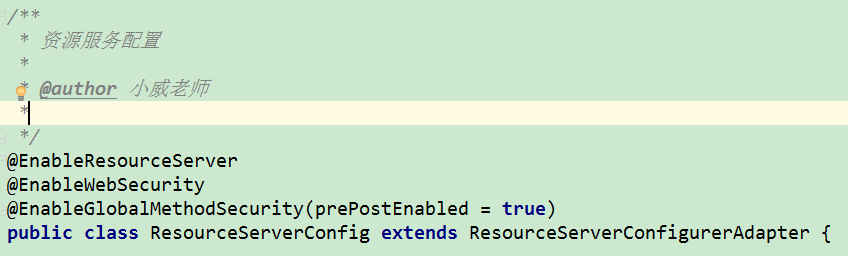


上传文件存储路径肯定是在jar包外部的，这里不像传统war包是解压成文件夹的，因此这里要做个静态资源的映射处理。

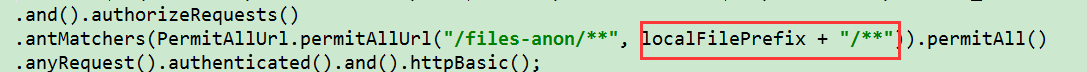


这里将url前缀和存储路径做了个映射

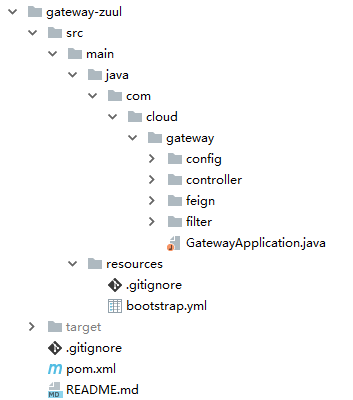
#### 资源服务器



这里要将静态资源下的路径放开权限



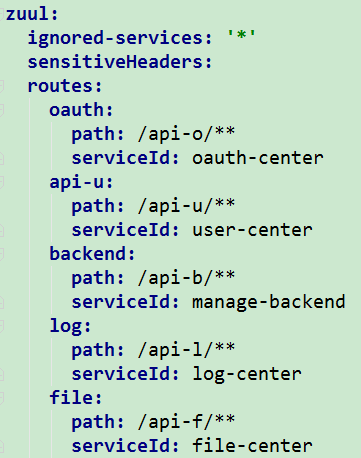
## 网关



bootstrap.yml里spring.application.name为gateway-zuul其余跟用户中心的一样

### gateway-zuul.yml

#### 路由规则



**sensitiveHeaders**过滤客户端附带的headers，如：

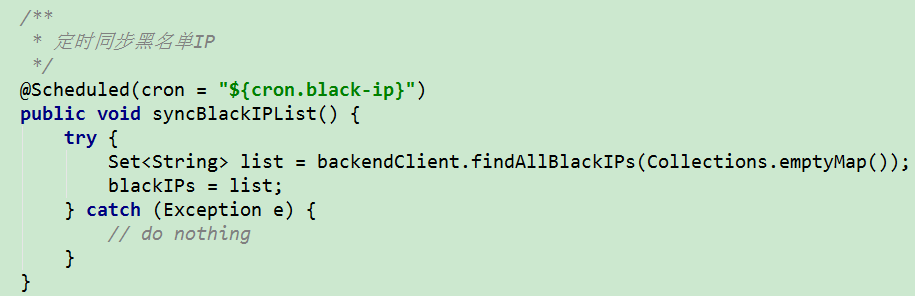
sensitiveHeaders: X-ABC  
如果在发请求时带了X-ABC，那么X-ABC不会往下游服务传递。

#### 自定义参数

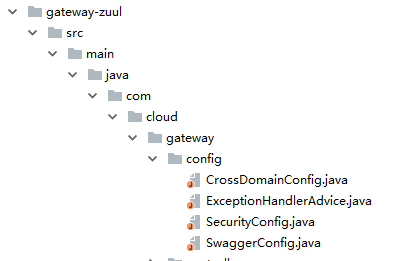


这里有个cron定时任务表达式，每5分钟执行一次，

[com](eclipse-javadoc:%E2%98%82=gateway-zuul/src%5C/main%5C/java%3Ccom).[cloud](eclipse-javadoc:%E2%98%82=gateway-zuul/src%5C/main%5C/java%3Ccom.cloud).[gateway](eclipse-javadoc:%E2%98%82=gateway-zuul/src%5C/main%5C/java%3Ccom.cloud.gateway).[filter](eclipse-javadoc:%E2%98%82=gateway-zuul/src%5C/main%5C/java%3Ccom.cloud.gateway.filter).BlackIPAccessFilter



### 配置类

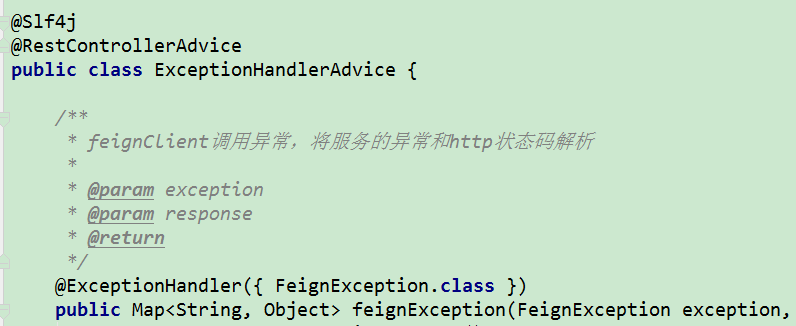


#### 跨域配置



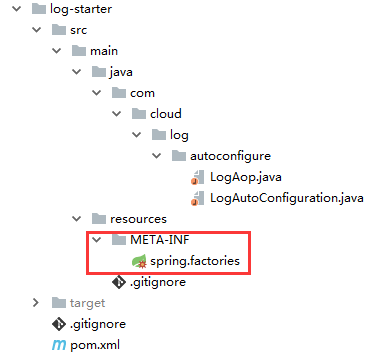
只需要在网关层配置，别的微服务不需要配置跨域

#### 异常处理

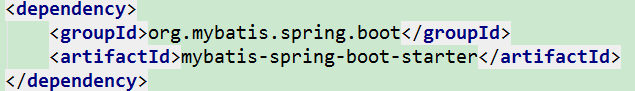


这里主要处理FeignException，这个是feignclient调用时的异常，不处理的话将会抛出500服务端异常，这里只是将下游服务的原始http状态码还原。

## 日志组件log-starter

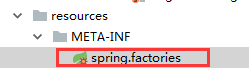


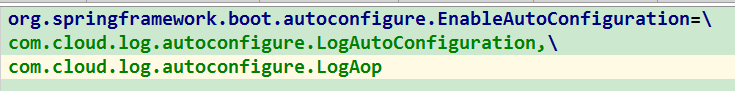
这里是模仿spring boot自动配置写的一个组件，就像spring boot里的各种starter，如



你只需要引入mybatis的starter，和数据源的配置，就可以用mybatis了。

### spring.factories



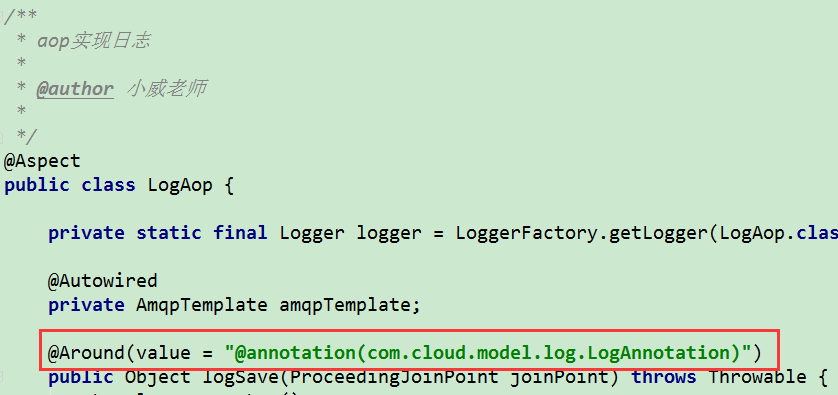
这里配置自动配置的类

### 使用该组件

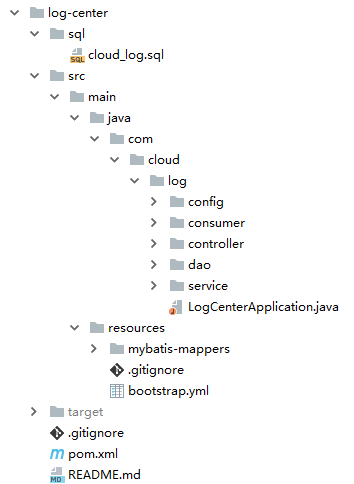
我们这里的log-starter是依赖rabbitmq的，只需要引入



再配置上mq信息，下图的aop类即可生效，就实现了aop日志拦截，将log信息发送到mq队列。



## 日志中心



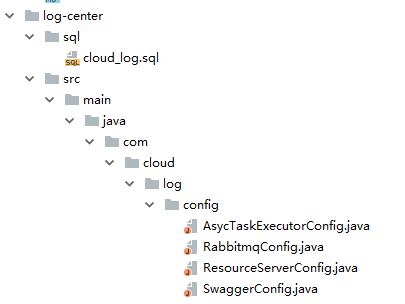
bootstrap.yml里spring.application.name为log-center其余跟用户中心的一样

### log-center.yml

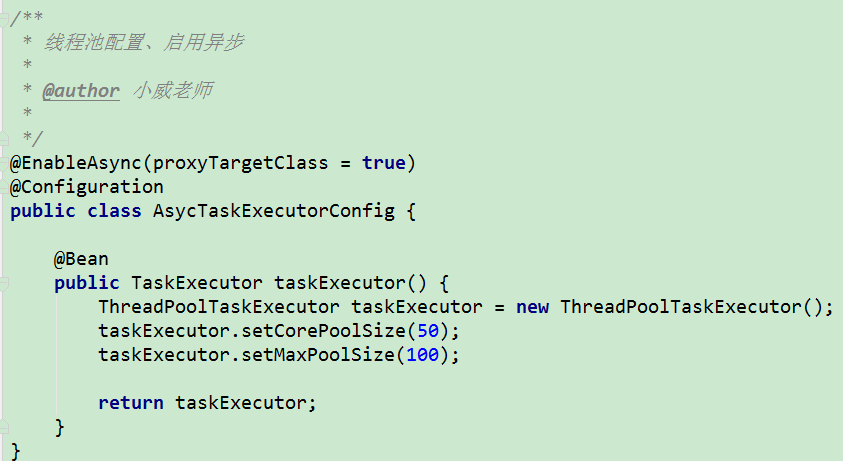


主要是数据库、mq、mybatis的配置

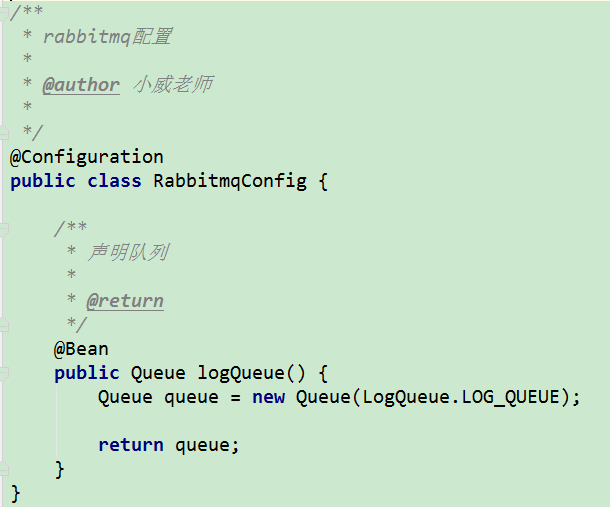
### 配置类



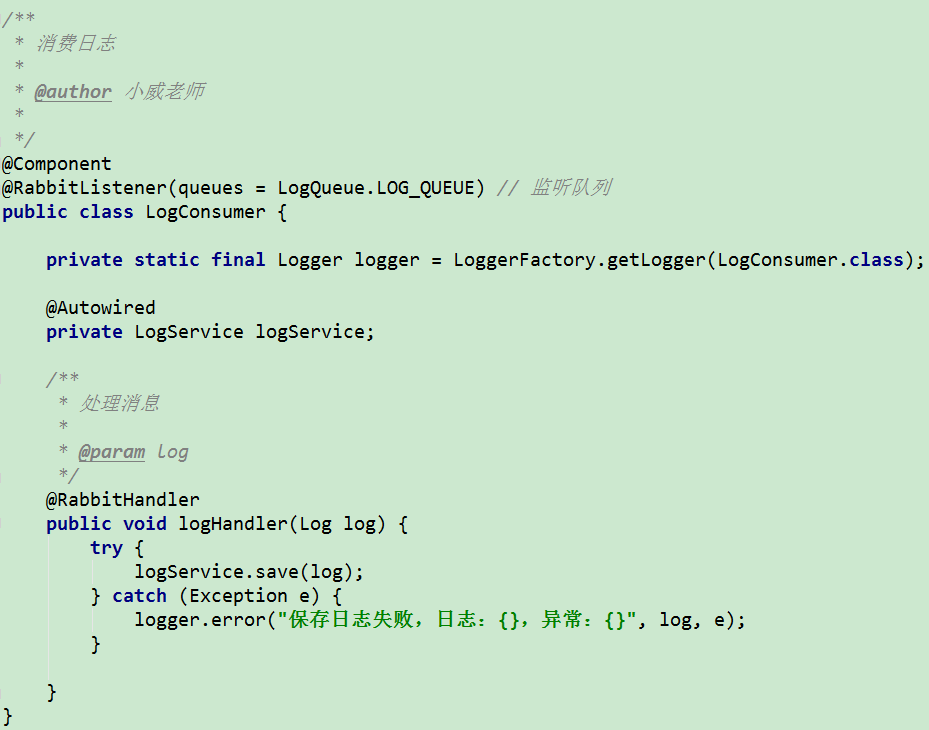
#### 开启异步线程池



#### 声明队列

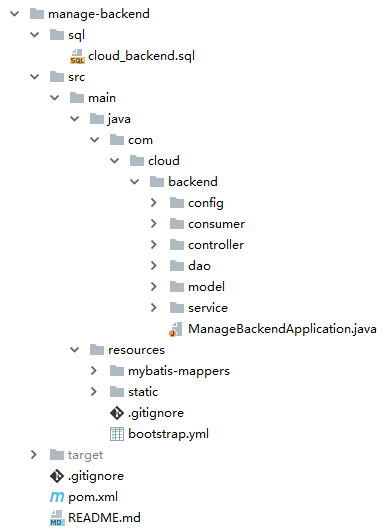


### 处理日志消息



从队列中处理消息，将日志存入数据库

## 后台管理系统



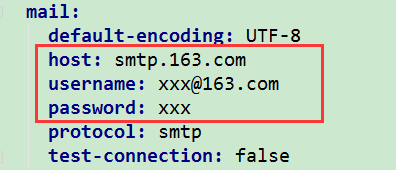
bootstrap.yml里spring.application.name为manage-backend其余跟用户中心的一样

### manage-backend.yml

#### 数据库和mq



#### 邮件配置



此处配置你自己的邮箱服务，否则发不出去邮件。

163邮箱如何开启POP3/SMTP/IMAP服务？

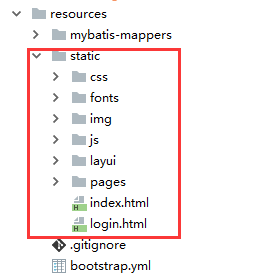
<http://help.163.com/10/0312/13/61J0LI3200752CLQ.html>

### 消息处理

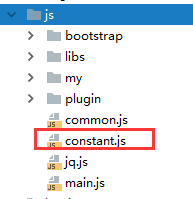


用户系统删除角色时，会抛消息，后台系统将接收该消息，删除菜单与角色的关系

### 静态资源



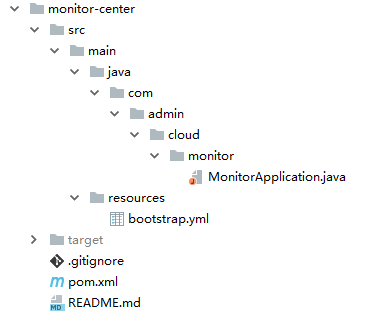
该目录的静态文件和页面是后台管理服务一部分，可单独拿出来部署，



这里定义了一个常量



## 监控中心



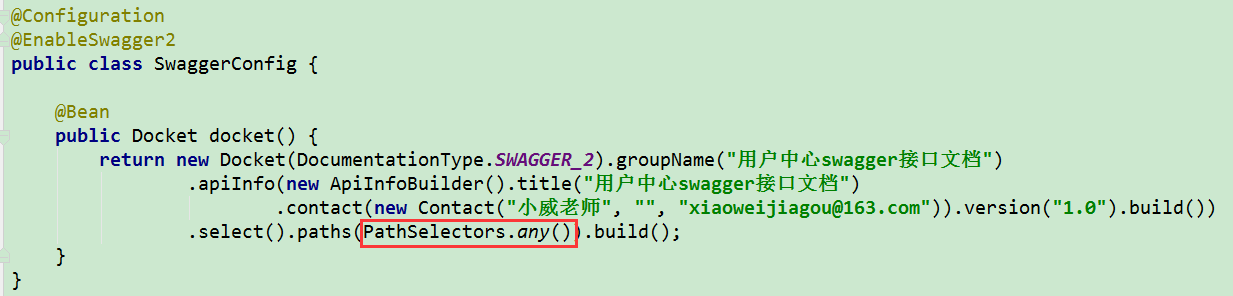
监控中心我们在bootstrap.yml指定了固定的端口号9001

访问http://localhost:9001即可查看监控界面

## Swagger配置

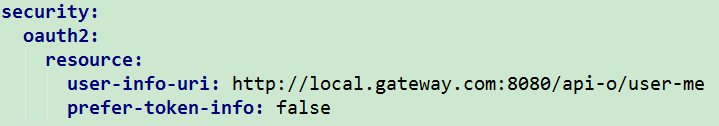
除注册中心、配置中心、监控中心不提供对外接口，别的项目都有

SwaggerConfig这个类，如下图



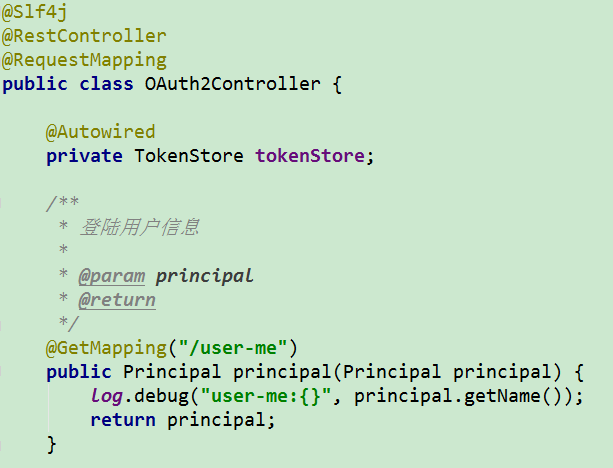
## user-info-uri

在配置中心许多微服务的配置里都有

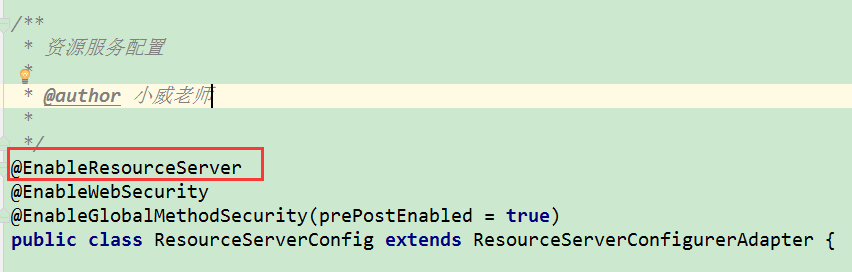


这里是从认证中心获取用户信息，

对应oauth-center里的接口，下图



因为加了注解@EnableResourceServer的各微服务都是资源服务器



是需要校验用户权限的，通过user-info-uri可以从认证中心获取到用户的信息和权限，感兴趣的可以看下

[org](eclipse-javadoc:%E2%98%82=file-center/C:%5C/Users%5C/wei.zhang1%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/oauth%5C/spring-security-oauth2%5C/2.0.14.RELEASE%5C/spring-security-oauth2-2.0.14.RELEASE.jar%3Corg).[springframework](eclipse-javadoc:%E2%98%82=file-center/C:%5C/Users%5C/wei.zhang1%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/oauth%5C/spring-security-oauth2%5C/2.0.14.RELEASE%5C/spring-security-oauth2-2.0.14.RELEASE.jar%3Corg.springframework).[security](eclipse-javadoc:%E2%98%82=file-center/C:%5C/Users%5C/wei.zhang1%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/oauth%5C/spring-security-oauth2%5C/2.0.14.RELEASE%5C/spring-security-oauth2-2.0.14.RELEASE.jar%3Corg.springframework.security).[oauth2](eclipse-javadoc:%E2%98%82=file-center/C:%5C/Users%5C/wei.zhang1%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/oauth%5C/spring-security-oauth2%5C/2.0.14.RELEASE%5C/spring-security-oauth2-2.0.14.RELEASE.jar%3Corg.springframework.security.oauth2).[provider](eclipse-javadoc:%E2%98%82=file-center/C:%5C/Users%5C/wei.zhang1%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/oauth%5C/spring-security-oauth2%5C/2.0.14.RELEASE%5C/spring-security-oauth2-2.0.14.RELEASE.jar%3Corg.springframework.security.oauth2.provider).[authentication](eclipse-javadoc:%E2%98%82=file-center/C:%5C/Users%5C/wei.zhang1%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/oauth%5C/spring-security-oauth2%5C/2.0.14.RELEASE%5C/spring-security-oauth2-2.0.14.RELEASE.jar%3Corg.springframework.security.oauth2.provider.authentication).OAuth2AuthenticationProcessingFilter

## commons模块

### PermitAllUrl

[com](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom).[cloud](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom.cloud).[common](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom.cloud.common).[constants](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom.cloud.common.constants).PermitAllUrl

该类主要定义了一些不需要权限拦截的url，



下图这几个是swagger文档需要放开的url，其余是监控中心需要放开的url



### AppUserUtil

[com](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom).[cloud](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom.cloud).[common](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom.cloud.common).[utils](eclipse-javadoc:%E2%98%82=commons/src%5C/main%5C/java%3Ccom.cloud.common.utils).AppUserUtil

该类是获取当前登录用户的工具类