1. Zookeeper注册中心服务启动

>zkServer.cmd

>zkCli.cmd

1. Redis-server服务启动
2. C:\project\public 放置public文件
3. Gamma-user

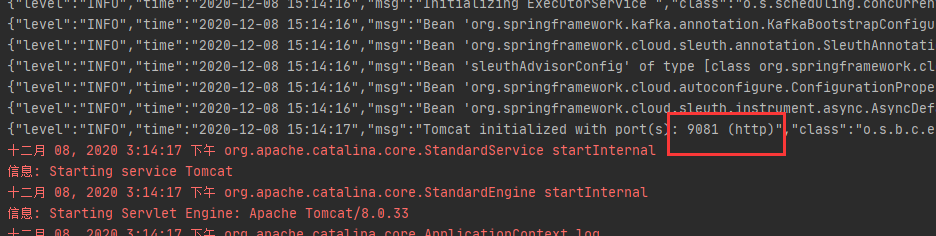
>UserCenterApplication启动用户中心服务 单点登录

1. 登录后：Eta-user 、

>先启动UaaWeb开启8883端口

>pubtransuserwebapplication启动后台

客户端口8883 后台端口9081

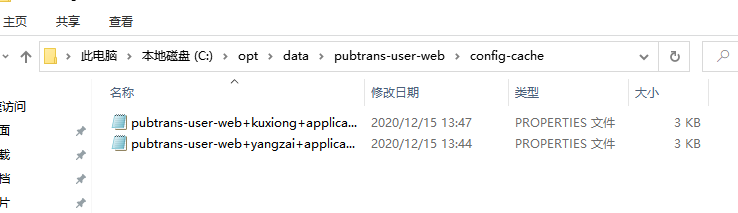


登录：ibus DtDream01

-Denv=dev -Dapollo.cluster=kuxiong -Dspring.profiles.active=test -Ddev\_meta=http://192.168.110.24:8080 -Dapollo.meta=http://192.168.110.24:8080

Gamma-user：用户中心服务层

Eta-user：用户中心WEB层



单点登录

1. 获取账户各个参数，用户名、密码等；
2. 创建账户记录对象，使用[shiro框架的UsernamePasswordToken](https://www.cnblogs.com/ihaokun/p/10073449.html)
3. 设置IP。进行登陆
4. 登录成功后，获取session，得到sid，
5. 创建CODE记录账户信息
6. 查询cookies
7. 传回UAA的sid，加入到cookie

RPC

Q1.当用户中心新建有集团管理员权限的A账号后，用该A账号新建B账号时，可以关联所有的机构，不合理，应该有且只有关联A账号下组织机构。同时给B账号关联角色的时候，也可以看到产品模块下的所有角色。

A:新增

EmployeeController

获取机构信息  
\*  
\* @param id 机构id或者机构下的子节点id  
\* @return 机构信息

findInsByHintNodeId

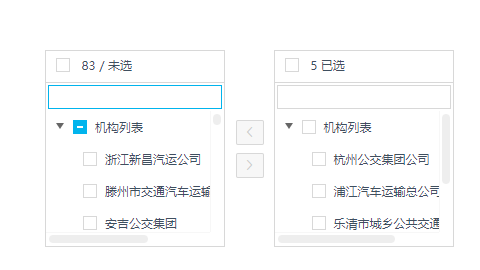
EmployeeWriteServiceImpl 职员写入

[bjtest@test.com](mailto:bjtest@test.com)

用户：bj3@test.com

地铁：[bj@ibuscloud.com](mailto:bj@ibuscloud.com)

获取机构列表时有getAssociatedInsList 和getUnAssociatedInsList



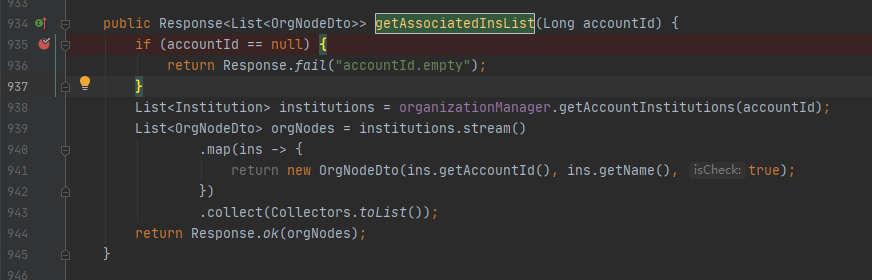
先北京用户

OrganizationReadService 获取机构列表接口

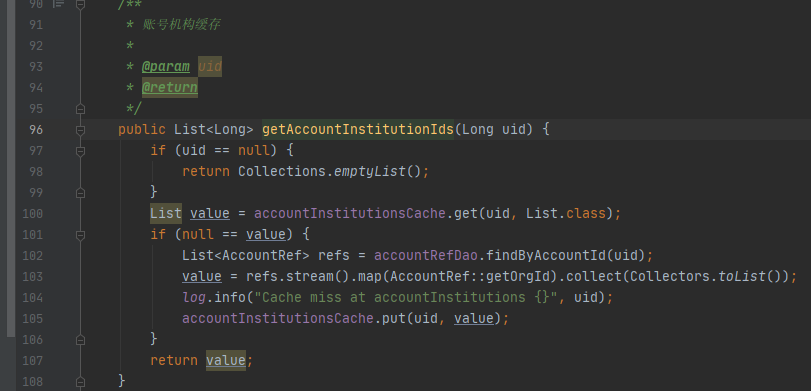
\* 根据AccountId获取所有关联的机构列表  
\* @param accountId 账号ID  
\* @return 机构列表

getAssociatedInsList 获取相关列表

实现

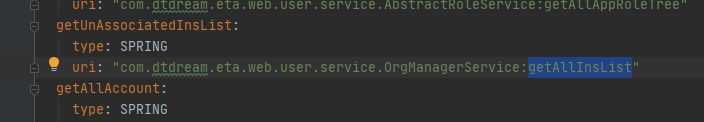


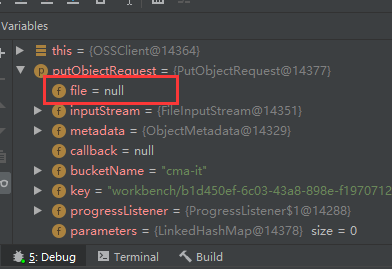
OrganizationManager中获取列表



List<AccountRef> refs = accountRefDao.findByAccountId(uid);

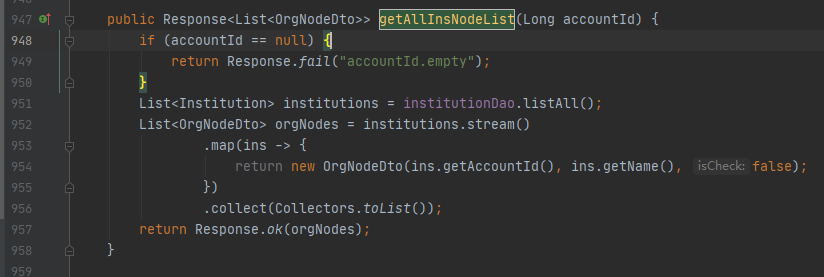
getUnAssociatedInsList:关联的是ALLins





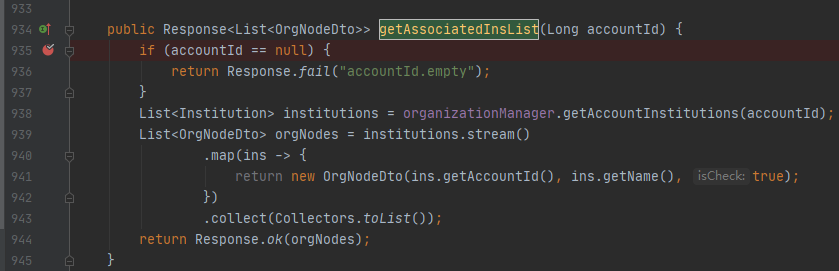


超管：



非超管执行：同样返回

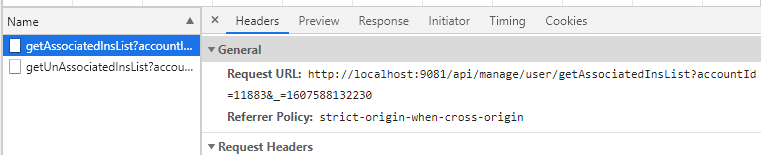
return this.getAssociatedInsList(accountId);

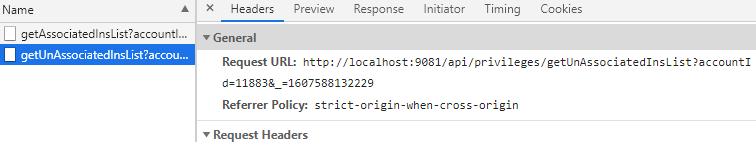


一次关联机构中，返回两次

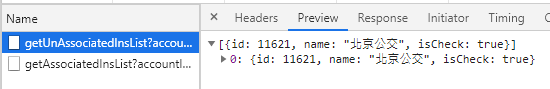
Account:11883

两此accountid均为11883





返回内容相同



getAllAppRoleTree

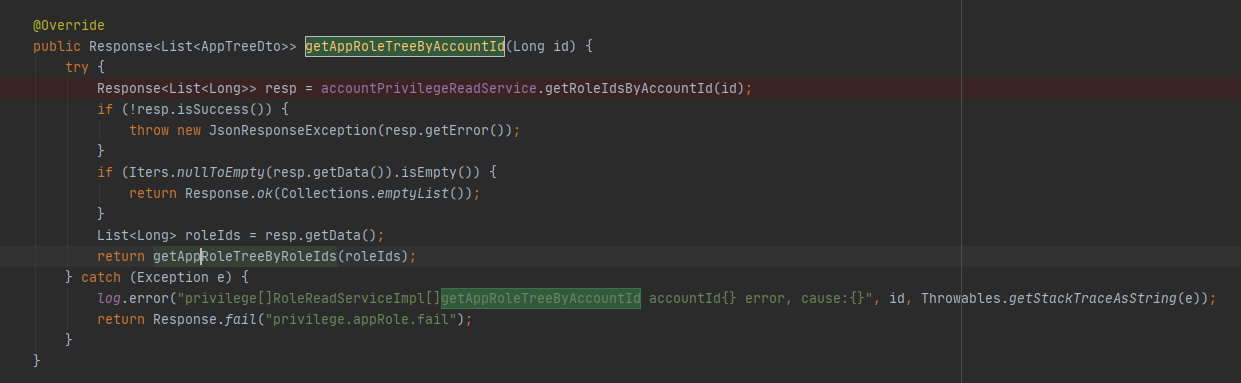
getAppRoleTreeByAccountId



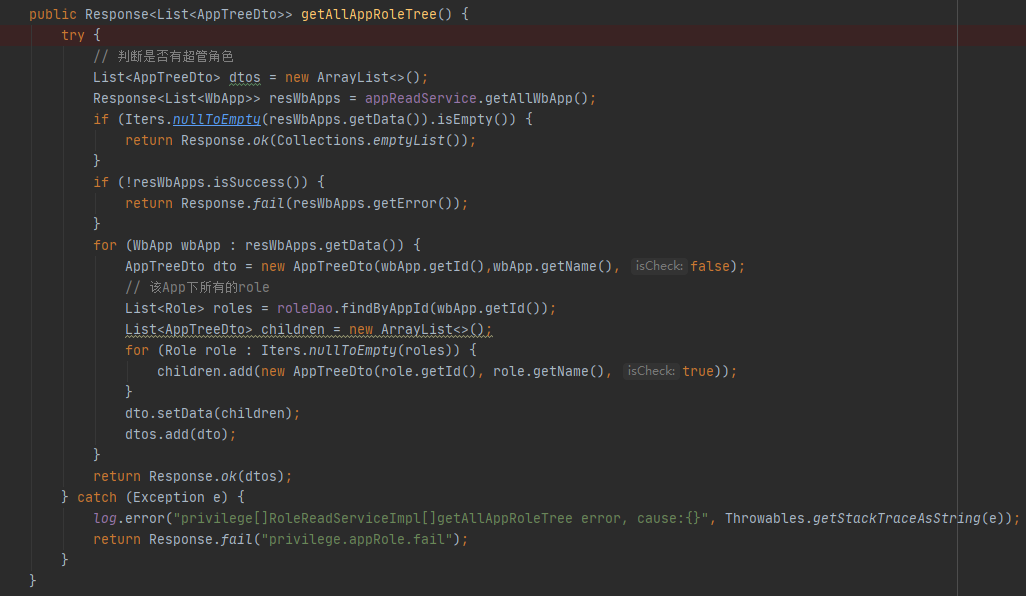


角色





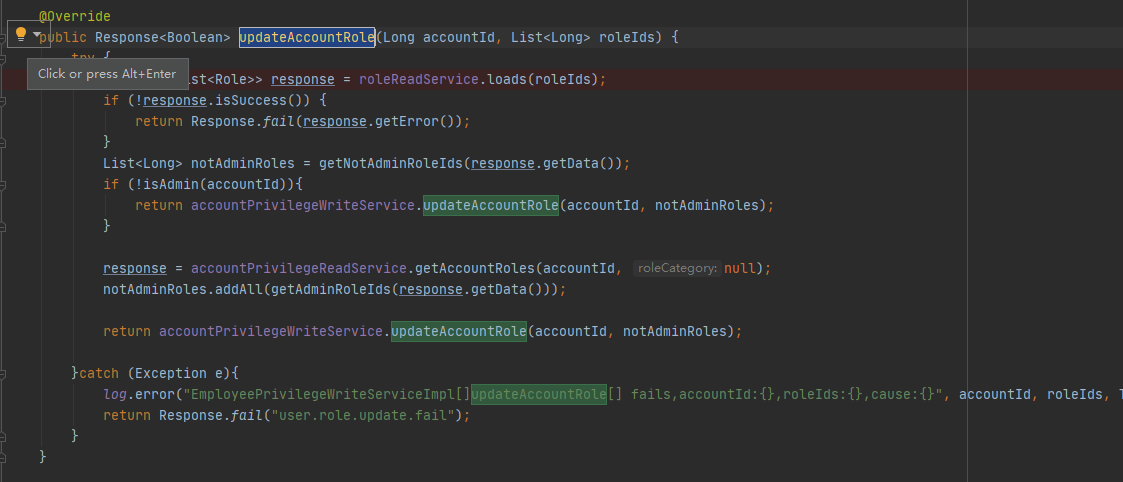




先查账户自己，再查客户

Response<Boolean> res = employeePrivilegeWriteService.updateAccountRole(param.getAccountId(), param.getRoleIds());

BUG



程序入口

角色应该修改



public void updateAccountRole(Long accountId, List<Long> roleIds) {

差客户时，先查的客户



public Response<Boolean> associateEmpAndOrgs(Long accountId, List<Long> newAccountIds) {

Controller： 地址到操作的映射

* @Controller标识是控制器
* @Slf4j 日志
* @RequestMapping("/api/user") 映射地址，写在类前则是一级地址，具体到方法时则在类基础上的二级地址。
* @Autowired 与IOC容器中对应的bean匹配，还有@Resource按照ByName自动注入
* @ResponseBody将java对象转为json格式的数据。将controller的方法返回的对象通过适当的转换器转换为指定的格式之后，写入到response对象的body区，通常用来返回JSON数据或者是XML数据。注意：在使用此注解之后不会再走视图处理器，而是直接将数据写入到输入流中，他的效果等同于通过response对象输出指定格式的数据。在使用 @RequestMapping后，返回值通常解析为跳转路径，但是加上 @ResponseBody 后返回结果不会被解析为跳转路径，而是直接写入 HTTP response body 中。 比如异步获取 json 数据，加上 @ResponseBody 后，会直接返回 json 数据。@RequestBody 将 HTTP 请求正文插入方法中，使用适合的 HttpMessageConverter 将请求体写入某个对象。
* @RestController 相当于@ResponseBody ＋ @Controller合在一起的作用。
* @ControllerAdvice：配合@ExceptionHandler、@InitBinder、@ModelAttribute 拦截异常并处理

Service:

Interface+implement

@Service：告诉[spring](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "https://www.cnblogs.com/guoziyi/p/_blank)，当Spring要创建UserServiceImpl的的实例时，bean的名字必须叫做"userService"，这样当Action需要使用UserServiceImpl的的实例时,就可以由Spring创建好的"userService"，然后注入给Action。

@Transactional：事务，应该只被应用到 public 方法上， Spring 建议不要在接口或者接口方法上使用该注解，因为这只有在使用基于接口的代理时它才会生效。

@GetMapping：@GetMapping用于处理请求方法的GET类型，@ PostMapping用于处理请求方法的POST类型等。如果我们想使用传统的@RequestMapping注释实现URL处理程序，那么它应该是这样的：@RequestMapping(value = "/get/{id}", method = RequestMethod.GET)

新方法可以简化为：@GetMapping("/get/{id}")。。。。同样还有@GetMapping。@PostMapping。@PutMapping。@DeleteMapping。@PatchMapping

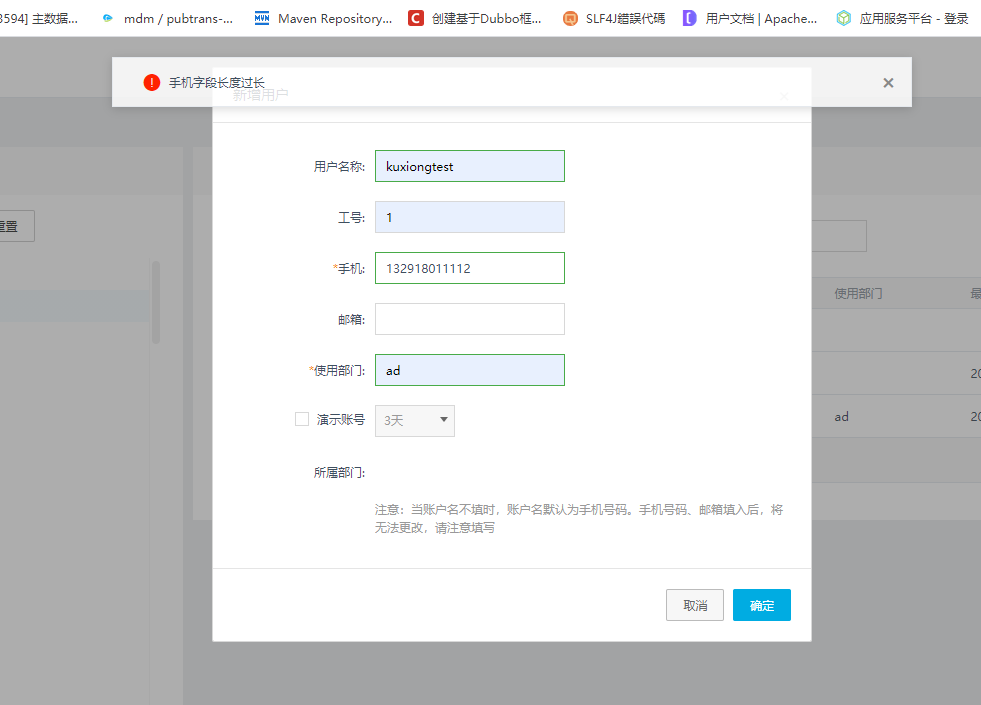
持久层

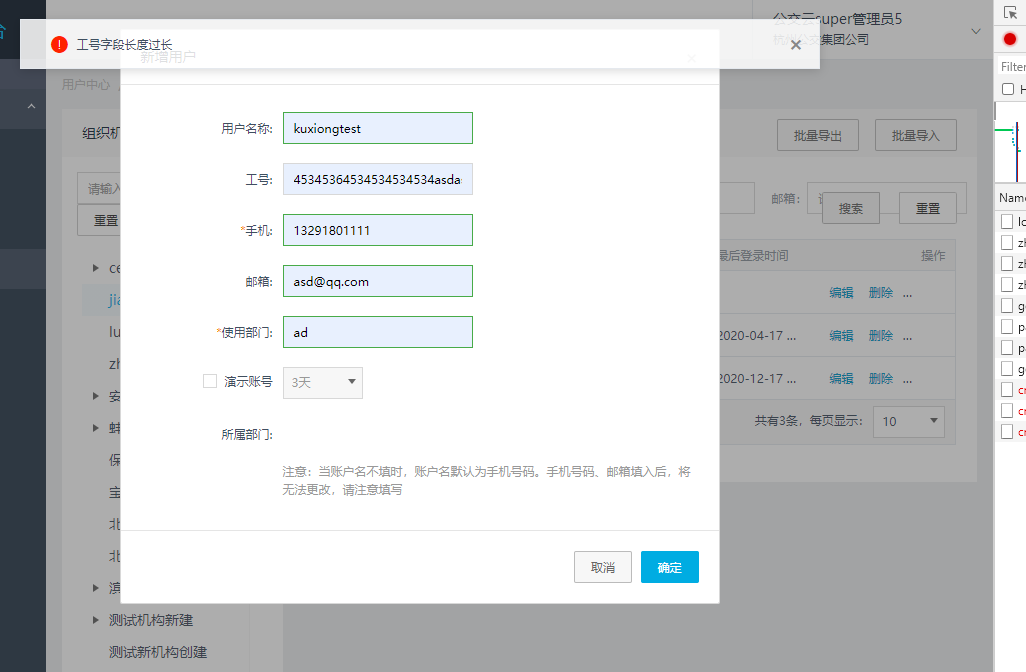
@Repository它用于将数据访问层 (DAO 层 ) 的类标识为 Spring Bean。具体只需将该注解标注在 DAO类上即可。



权限吗查询

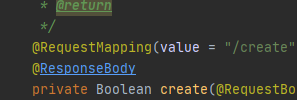
privilegeDao.findByCode(privilege.getCode()))







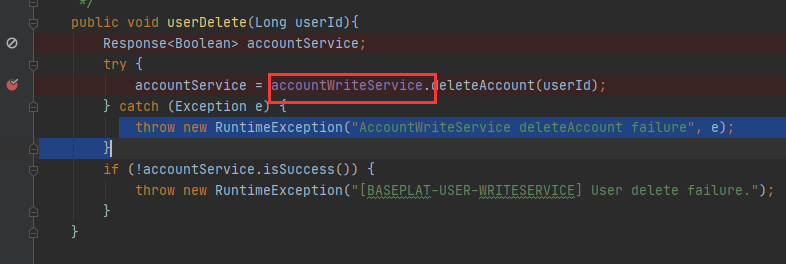
账号创建



删除



删除用户前端页面中调用accountwriteservice的deleteaccount方法



删除基础账户信息 不删除权限信息  
\* 如果account关联了权限，角色，请使用accountPrivilegeWriteService中的deleteAccount(Long  
\* id)进行删除

根据不同的机构ID查询可查询的用户

删除-只删除该机构下的用户-取消该机构和用户的关联-使用关联机构

当所有机构都删除：

白名单：添加账户IP

输入IP为String，

WhiteIpListUtils.isVerifiedWhiteIp验证

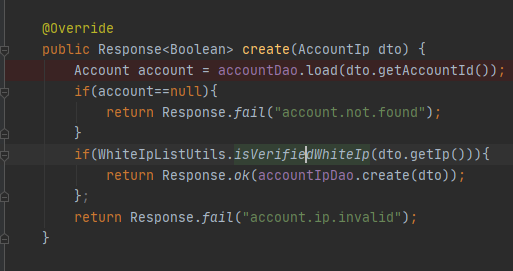
Response<Boolean> create(AccountIp var1);

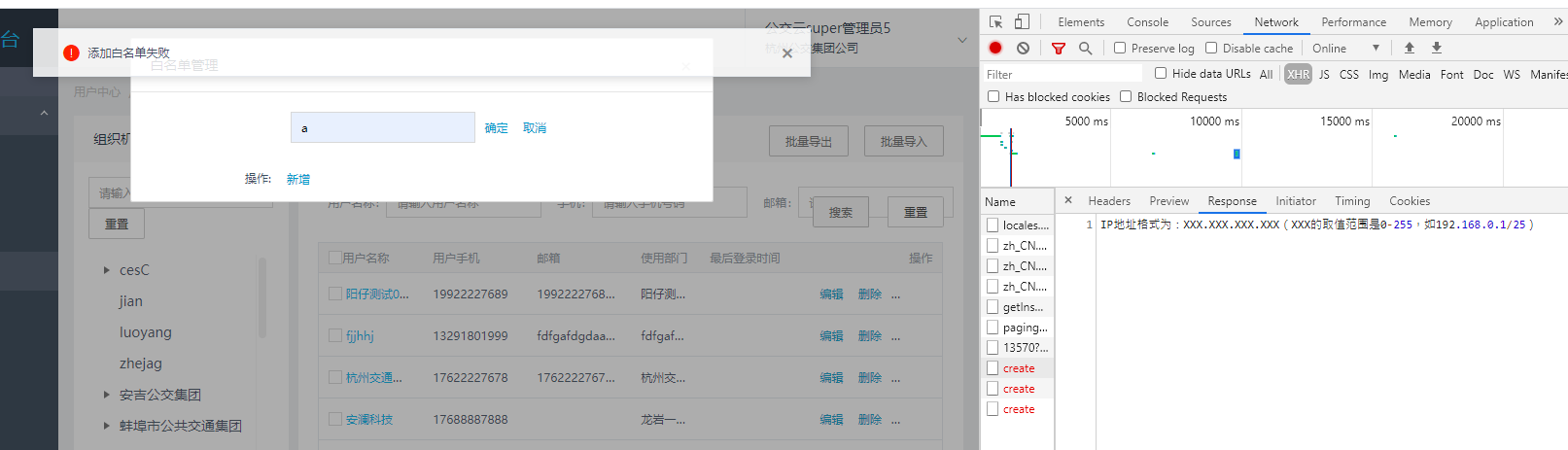
应当修改

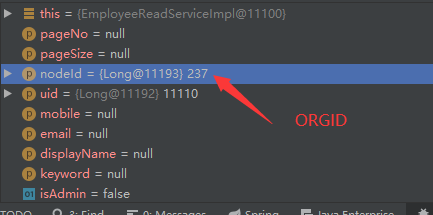


先使用 . 将字符串分割

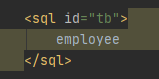
如果长度不为4，则返回F

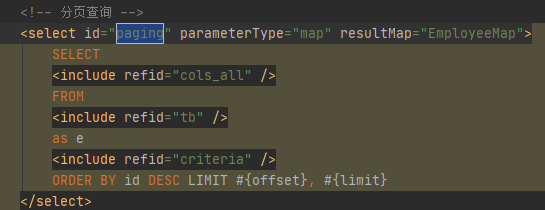






关联修改：

机构和账号查询从employee表中查询



分公司和机构映射关系，NodeID为ID号，对应机构ID department表



该表记录分级机构和账户关系

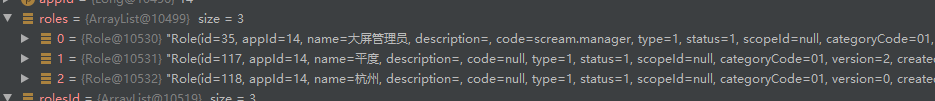
修改的表 org-user-ref

roleManage/detail 服务：getRoleDetail

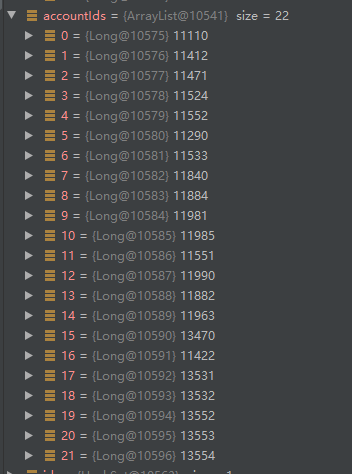
privilege/roleManage：getElements

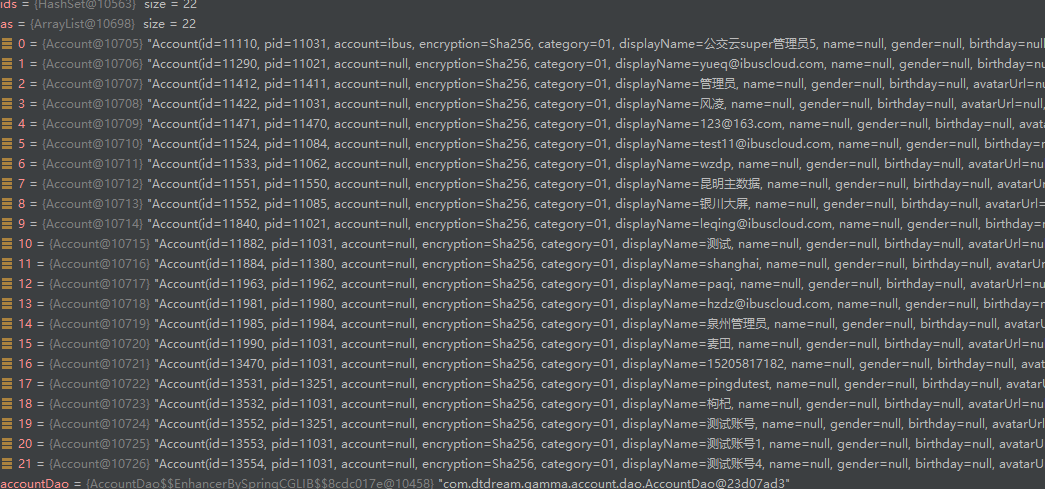
获取角色





获取账户





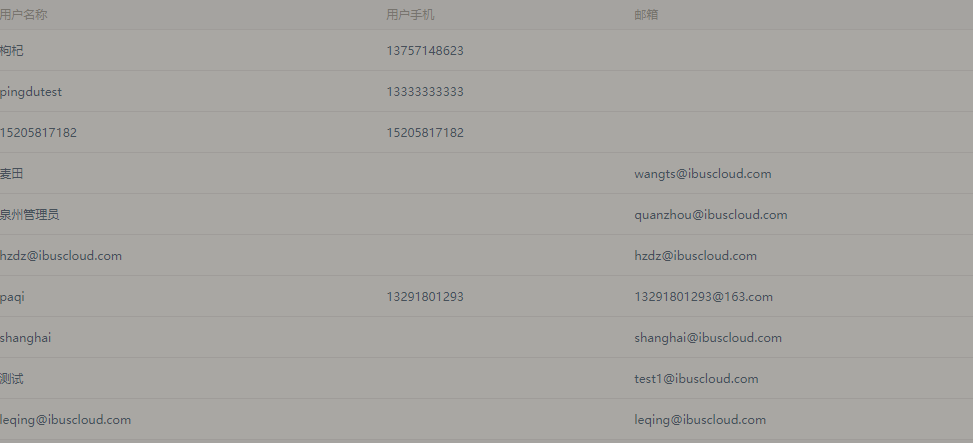
杭州



平度



大屏管理员





Response<List<Account>> getAccountsByAppId(Long appId)

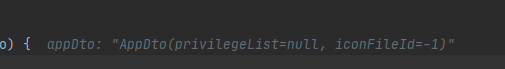
图标问题

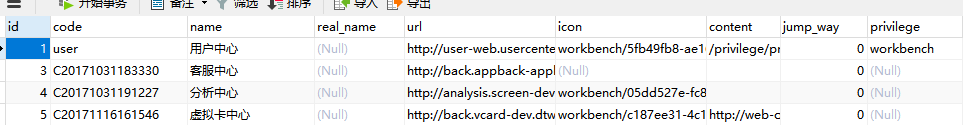
AppManagerController

uploadIcon

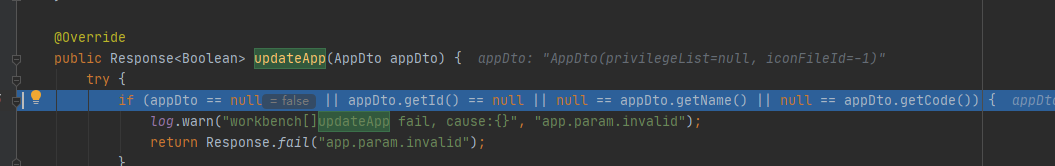
更新调用 APP写服务

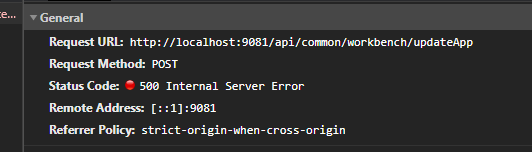
Response<Boolean> updateApp(AppDto appDto) {



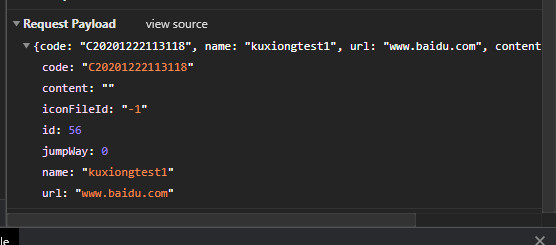


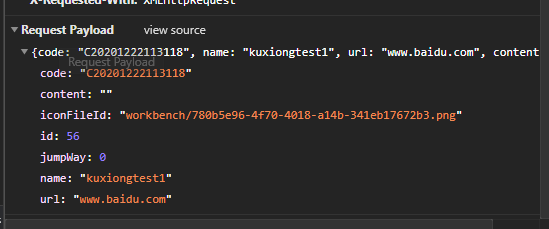
所有的传入的iconfileid都是-1



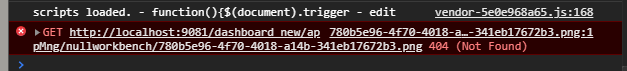


请求参数是payload，jason格式



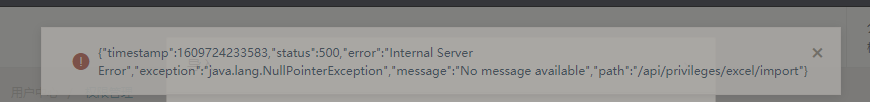


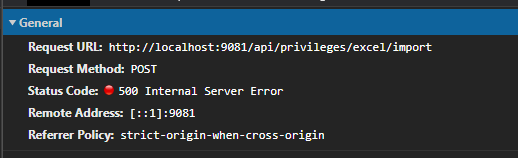
{"code":"C20201222113118","name":"kuxiongtest1","url":"www.baidu.com","content":"","iconFileId":"workbench/780b5e96-4f70-4018-a14b-341eb17672b3.png","jumpWay":0,"id":56}

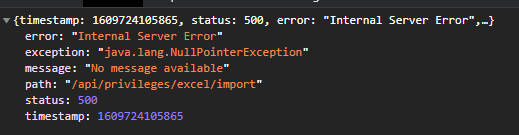


上传图像，

权限批量导入文件页面提示









PrivilegeManagerController.importExcel

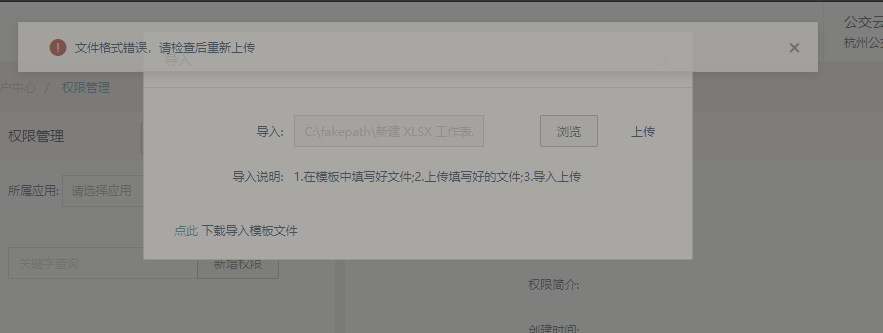
使用excelUtil工具根据文件流和文件名读取EXCEL表格

因为表为空，所以报了空指针异常

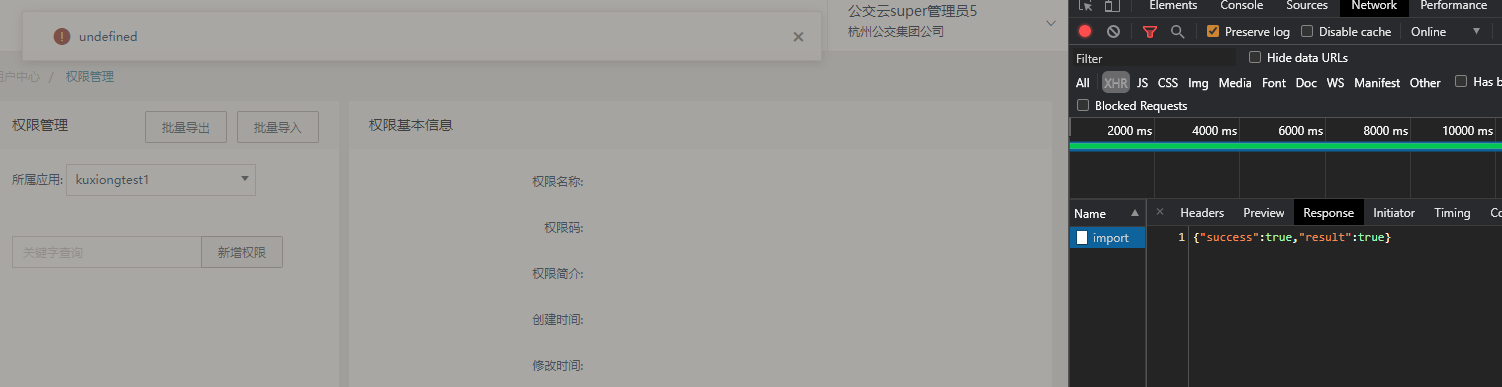
修改：通过try抓取再表格转换时发生的异常，抛出jason

模板错误

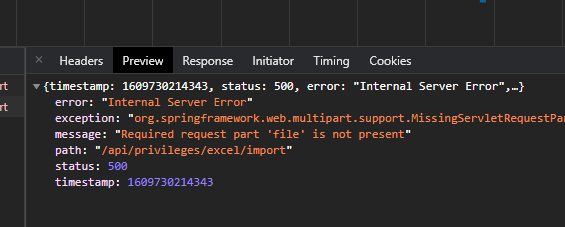
文件格式错误



正确文件



问题：



主数据和用户中心里有重名的角色如“管理员”，导入后用户有了所有应用的管理员角色

关键词：用户，角色

从表中读取到的List<String> roles;

在雇员写服务的importUsers方法中进行了账户创建导入和角色关联，

角色关联部分：

organization\service\EmployeeWriteServiceImpl.getRoleIds

角色需要2个限定。

应用限定和登录用户限定

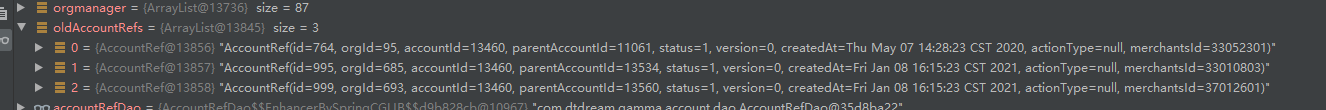
EmployeeWriteServiceImpl.importUsers

需要修改的

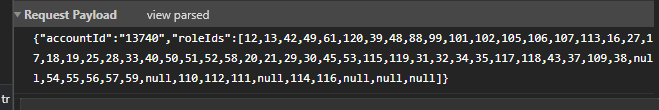
新增：roleReadService.findByNamesAndAppIds();

22222222222

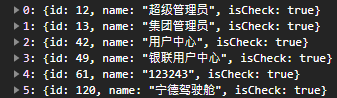
AppReadService.findByNames();

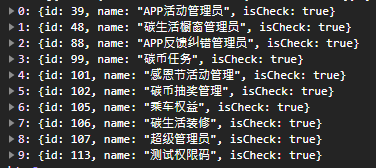


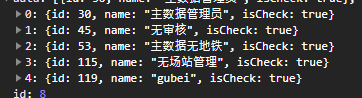
Post



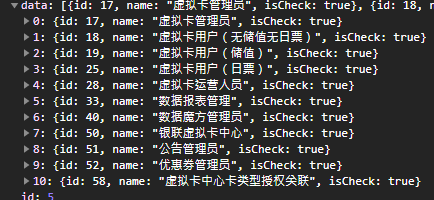




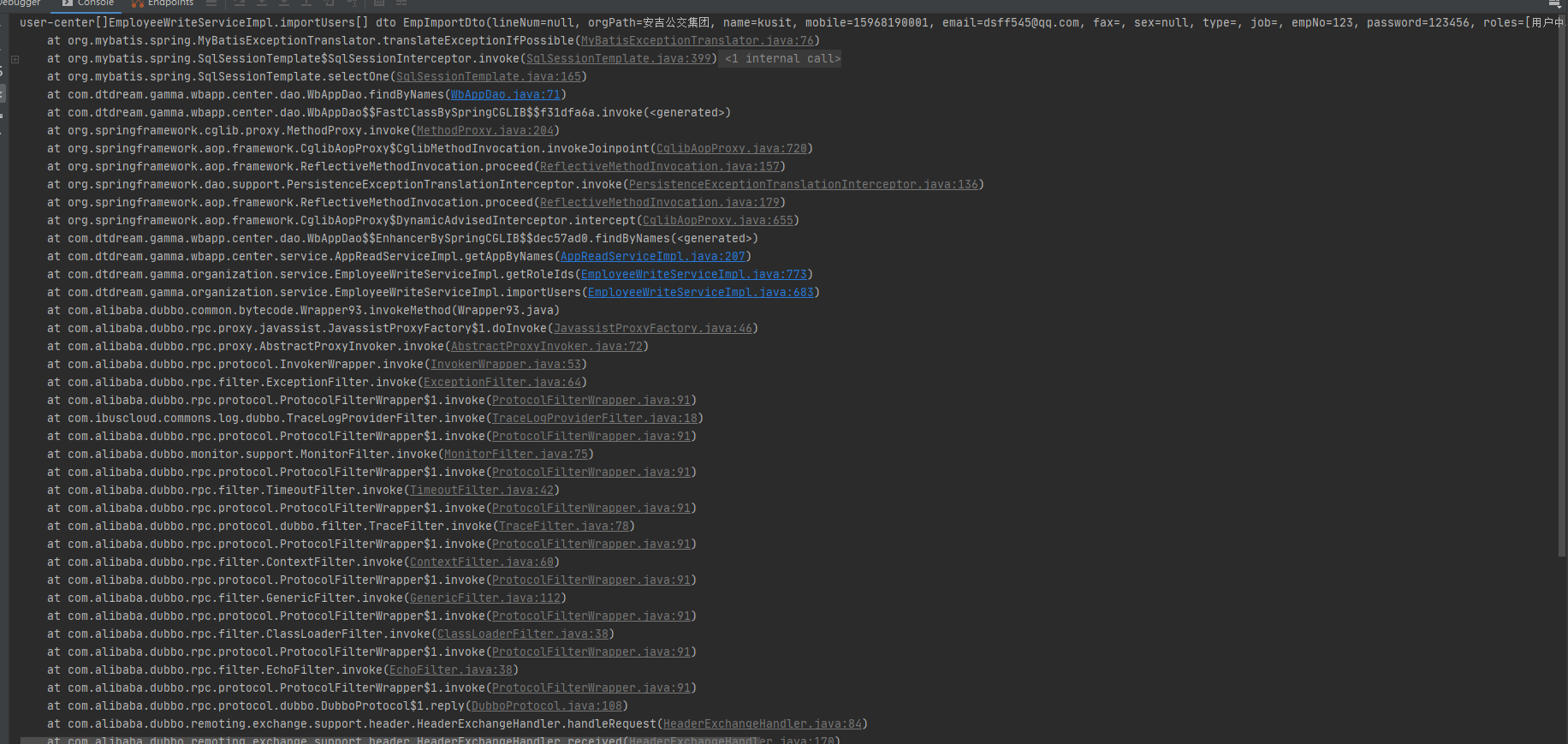


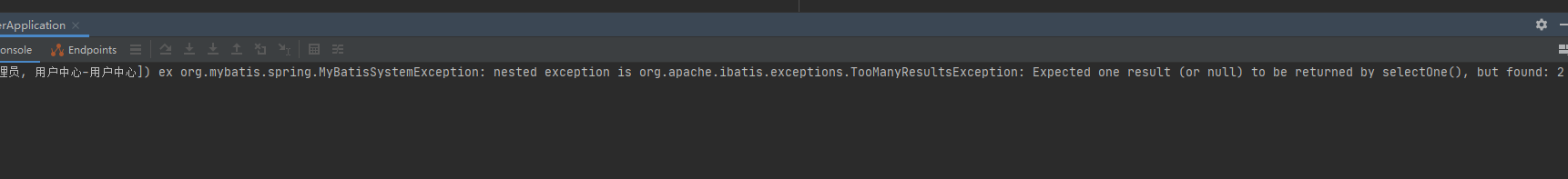




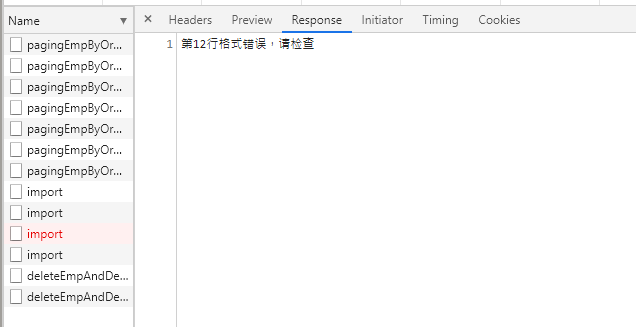


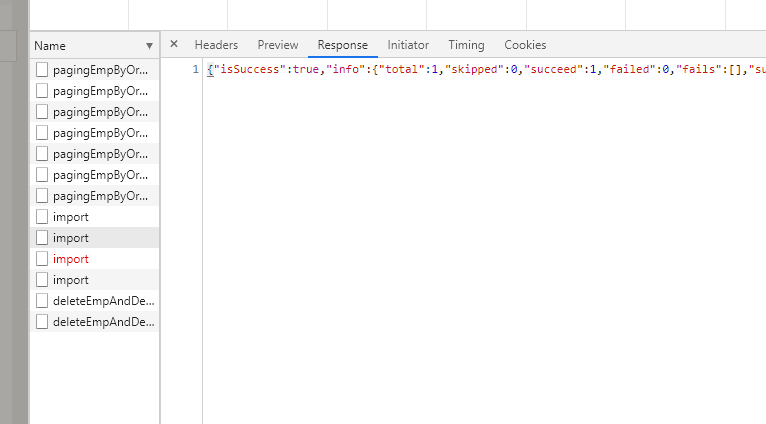










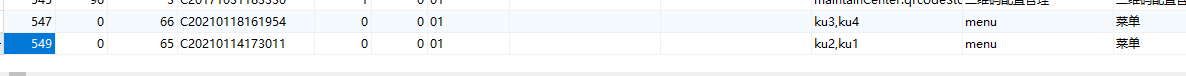


权限关联：

点击某个菜单

页面菜单读 ApplicationPrivilegeManageService.getMenuInfo

此时发现已经删除的权限码依然存在 此时是从menu数据表中读取的



编辑某个菜单

getPrivilegeTreeByAppId

从privilege中读取的权限 因为权限已经删掉所以 查询正确

正则表达式错误 code%

测试

正常删除权限后，菜单关联也正常取消

删除后 查询本应用下的权限 显示权限已经存在 菜单关联并未取消