**项目名称：**

——————通用后台权限管理系统

项目目标：

1. 掌握软件开发的基本流程
2. 掌握springboot框架
3. 掌握mybatis框架各知识点
4. 掌握shiro安全框架的认证和授权
5. 掌握html和javascript各知识点
6. 掌握mysql各知识点

项目大纲：

1. 背景知识
2. 需求分析
3. 功能分析
4. 架构设计
5. 代码开发

小组成员：辛挺，杨顺成，曾环凯，杨伟涛，黄星扬

1. 目录

[1、背景知识 4](#_Toc41822170)

[2、需求分析 5](#_Toc41822171)

[3、功能分析 6](#_Toc41822172)

[3.1、登录功能 6](#_Toc41822173)

[3.2、用户管理 7](#_Toc41822174)

[3.3、部门管理 9](#_Toc41822175)

[3.4、角色管理 11](#_Toc41822176)

[3.5、菜单管理 13](#_Toc41822177)

[4、原型设计 15](#_Toc41822178)

[5、架构设计 15](#_Toc41822179)

[5.1、整体架构设计 15](#_Toc41822180)

[5.7、数据模型设计 16](#_Toc41822181)

[I 简介 16](#_Toc41822182)

[I.1 描述 16](#_Toc41822183)

[I.2 模型 PhysicalDataModel\_2的卡片 16](#_Toc41822184)

[II 简单模型描述 17](#_Toc41822185)

[II.1 图的清单 17](#_Toc41822186)

[II.2 图PhysicalDiagram\_1 17](#_Toc41822187)

[II.3 表的清单 18](#_Toc41822188)

[II.4 引用的清单 18](#_Toc41822189)

[III 完全模型描述 19](#_Toc41822190)

[III.1 图的清单 19](#_Toc41822191)

[III.2 图PhysicalDiagram\_1 19](#_Toc41822192)

[III.3 表的清单 20](#_Toc41822193)

[III.3.1 表tbl\_dept 20](#_Toc41822194)

[III.3.2 表tbl\_loginfo 21](#_Toc41822195)

[III.3.3 表tbl\_notice 22](#_Toc41822196)

[III.3.4 表tbl\_permission 23](#_Toc41822197)

[III.3.5 表tbl\_role 24](#_Toc41822198)

[III.3.6 表tbl\_role\_permission 25](#_Toc41822199)

[III.3.7 表tbl\_role\_user 26](#_Toc41822200)

[III.3.8 表tbl\_user 27](#_Toc41822201)

[III.4 引用的清单 29](#_Toc41822202)

[III.4.1 引用tbl\_role\_permission\_ibfk\_1 29](#_Toc41822203)

[III.4.2 引用tbl\_role\_permission\_ibfk\_2 30](#_Toc41822204)

[III.4.3 引用tbl\_role\_user\_ibfk\_1 30](#_Toc41822205)

[III.4.4 引用tbl\_user\_ibfk\_1 31](#_Toc41822206)

[6、 代码开发 32](#_Toc41822207)

[一、 创建数据库 32](#_Toc41822208)

[二、 集成项目 32](#_Toc41822209)

[III.5 创建项目 32](#_Toc41822210)

[三、 用户登陆 33](#_Toc41822211)

[III.5.1 生成 33](#_Toc41822212)

[III.5.2 创建ActiverUser 33](#_Toc41822213)

[III.5.3 创建UserRealm 34](#_Toc41822214)

[III.5.4 创建ShiroAutoConfigruation 34](#_Toc41822215)

[III.5.5 创建ResultObj 36](#_Toc41822216)

[III.5.6 创建Constast 37](#_Toc41822217)

[III.5.7 创建SystemController 38](#_Toc41822218)

[III.5.8 创建LoginController 39](#_Toc41822219)

[四、 加载系统主页的左边的导航树 40](#_Toc41822220)

[III.6 生成Permission 40](#_Toc41822221)

[III.7 生成 41](#_Toc41822222)

[III.8 PermissionMapper,PermissionMapper.xml,PermissionService ,PermissionServiceImpl 41](#_Toc41822223)

[III.9 生成PermissionController 41](#_Toc41822224)

[III.10 创建DataGridView 44](#_Toc41822225)

[III.11 修改Constast 45](#_Toc41822226)

[III.12 创建TreeNode 45](#_Toc41822227)

[III.13 创建TreeNodeBuilder 46](#_Toc41822228)

[III.14 创建MenuController 46](#_Toc41822229)

[III.15 创建PermissionVo 47](#_Toc41822230)

[五、 工作台的管理 48](#_Toc41822231)

[III.16 创建deskManager.html 48](#_Toc41822232)

[III.17 修改SystemController 48](#_Toc41822233)

[六、 日志管理 50](#_Toc41822234)

[III.18 查询 50](#_Toc41822235)

[III.18.1 生成Loginfo 50](#_Toc41822236)

[III.18.2 生成LoginfoMapper， LoginfoMapper.xml， LoginfoService， LoginfoServiceImpl 50](#_Toc41822237)

[III.18.3 生成LoginfoController 51](#_Toc41822238)

[III.18.4 创建LoginfoVo 51](#_Toc41822239)

[III.18.5 创建system/loginfo/loginfoManager.html 52](#_Toc41822240)

[III.18.6 修改菜单地址 52](#_Toc41822241)

[III.19 删除和批量删除 53](#_Toc41822242)

[III.19.1 修改loginfoVo 53](#_Toc41822243)

[III.19.2 修改ResultObj 53](#_Toc41822244)

[III.19.3 修改LogInfoController 54](#_Toc41822245)

[III.20 记录登陆日志 55](#_Toc41822246)

[III.20.1 修改LoginController 55](#_Toc41822247)

[七、 公告管理 56](#_Toc41822248)

[III.21 查询 56](#_Toc41822249)

[III.21.1 生成Notice 56](#_Toc41822250)

[III.21.2 生成 57](#_Toc41822251)

[III.21.3 NoticeMapper,NoticeMapper.xml,NoticeService,NoticeServiceImpl 57](#_Toc41822252)

[III.21.4 创建NoticeVo 57](#_Toc41822253)

[III.21.5 生成NoticeController 58](#_Toc41822254)

[III.21.6 修改SystemController 58](#_Toc41822255)

[III.21.7 修改菜单地址 58](#_Toc41822256)

[III.21.8 创建system/notice/noticeManager.html 59](#_Toc41822257)

[III.22 添加修改 60](#_Toc41822258)

[III.22.1 修改NoticeController 60](#_Toc41822259)

[III.23 查看 61](#_Toc41822260)

[八、 部门管理 62](#_Toc41822261)

[III.24 查询 62](#_Toc41822262)

[III.24.1 生成Dept 62](#_Toc41822263)

[III.24.2 生成DeptMapper DeptMapper.xml DeptService,DeptService,DeptServiceImpl 63](#_Toc41822264)

[III.24.3 修改TreeNode 64](#_Toc41822265)

[III.24.4 创建DeptController 65](#_Toc41822266)

[III.24.5 引入dtree 65](#_Toc41822267)

[III.24.6 修改SystemController 66](#_Toc41822268)

[III.24.7 修改菜单地址 67](#_Toc41822269)

[III.25 添加修改 67](#_Toc41822270)

[III.25.1 修改DeptController 68](#_Toc41822271)

[III.26 删除 68](#_Toc41822272)

[III.26.1 修改DeptController 69](#_Toc41822273)

[III.27 缓存处理 69](#_Toc41822274)

[III.27.1 创建CacheAspect 69](#_Toc41822275)

[九、 菜单管理CRUD 71](#_Toc41822276)

[III.28 修改MenuController 71](#_Toc41822277)

[III.29 解决删除的问题 73](#_Toc41822278)

[III.30 解决删除最后一行数据的分页问题 73](#_Toc41822279)

[十、 权限管理CRUD 74](#_Toc41822280)

[III.31 修改PermissionController 74](#_Toc41822281)

[十一、 角色管理 77](#_Toc41822282)

[III.32 角色的CRUD 77](#_Toc41822283)

[III.32.1 生成Role 77](#_Toc41822284)

[III.32.2 生成RoleMapper,RoleMapper.xml,RoleService, RoleServiceImpl 77](#_Toc41822285)

[III.32.3 创建RoleVo 78](#_Toc41822286)

[III.32.4 生成RoleController 78](#_Toc41822287)

[III.32.5 创建role/roleManager.html 80](#_Toc41822288)

[III.32.6 修改菜单地址 80](#_Toc41822289)

[III.33 角色分配权限 81](#_Toc41822290)

[III.33.1 修改role/roleManager.html 81](#_Toc41822291)

[III.33.2 修改TreeNode 82](#_Toc41822292)

[III.33.3 修改RoleController 83](#_Toc41822293)

[十二、 用户管理 84](#_Toc41822294)

[III.34 查询 84](#_Toc41822295)

[III.34.1 修改User 84](#_Toc41822296)

[III.34.2 创建UserVo 85](#_Toc41822297)

[III.34.3 修改UserServiceImpl 85](#_Toc41822298)

[III.34.4 修改DeptServiceImpl 86](#_Toc41822299)

[III.34.5 创建UserController 86](#_Toc41822300)

[III.34.6 创建SystemController 88](#_Toc41822301)

[III.34.7 创建user/userManager.html 88](#_Toc41822302)

[III.35 添加用户 89](#_Toc41822303)

[III.35.1 修改userControlller 89](#_Toc41822304)

[III.35.2 修改pom.xml 90](#_Toc41822305)

[III.35.3 创建PinyinUtils 90](#_Toc41822306)

[III.36 修改用户 90](#_Toc41822307)

[III.36.1 修改userController 91](#_Toc41822308)

[III.37 删除用户 91](#_Toc41822309)

[III.37.1 修改userController 92](#_Toc41822310)

[III.37.2 修改RoleMapper 92](#_Toc41822311)

[III.37.3 修改RoleMapper.xml 92](#_Toc41822312)

[III.37.4 修改UserController 93](#_Toc41822313)

[III.38 分配角色 93](#_Toc41822314)

[III.38.1 修改userController 93](#_Toc41822315)

[III.38.2 修改RoleMapper 94](#_Toc41822316)

[III.38.3 修改RoleMapper.xml 94](#_Toc41822317)

[十三、 登陆关联菜单和权限 95](#_Toc41822318)

[III.39 关联菜单 95](#_Toc41822319)

[III.40 关联权限 95](#_Toc41822320)

[III.40.1 修改UserRealm 95](#_Toc41822321)

## 1、背景知识

在当前风起云涌的电商领域和广大企业信息化管理领域，软件系统的后台管理都是不可或缺的组成部分，而后台管理中的权限管理模块是衡量一个后台管理是否安全高效的重要指标。

## 2、需求分析

由于不同的后台管理系统需求多样化，本次开发的是通用型，对于大多数的后台管理系统逻辑都已足够使用，主要应用于WEB应用程序，如：网站管理后台、CMS、CRM、OA、商城后台等等。

目前常见的权限控制分为四大类：

1. **用户管理**

对于后台产品而言，是需要公司内部人员去创建账号的。而这个账号就是一把钥匙，我们通过控制账号所具备的权限，进而控制这个员工的所操作范围。

1. **角色管理**

角色往往是基于业务管理需求而预先在系统中设定好的固定标签，每个角色对应明确的系统权限，他是一个集合的概念，是众多最小权限颗粒的组成。我们通过把权限给这个角色，再把角色给账号，从而实现账号的权限，因此它承担了一个桥梁的作用。引入角色这个概念，可以帮助我们灵活的扩展，使一个账号可以具备多种角色。

1. **功能权限控制**

功能权限定义：为可见、可以操作的功能范围。在目录、菜单上加权限控制，有权限的就可以访问对应模块，没有的连菜单名都看不到。在业务模块的功能按钮上加权限控制，最小粒度的控制用户行为。

1. **数据权限控制**

数据权限控制主要控制某条数据记录对用户是否可见，结合功能权限可以更灵活的配置业务过程中每一位员工的功能操作权限及数据可见范围，全面保障企业数据的安全性。

## 3、功能分析

### 3.1、登录功能

#### 3.1.1、登录页面



图3-1-1

#### 3.1.2、异常登录提示错误信息



图3-1-2

### 3.2、用户管理

#### 3.2.1、查询用户



图3-2-1

#### 3.2.2、添加用户，设置角色



图3-2-2

#### 3.2.3、修改用户，禁用用户



图3-2-3

#### 3.2.4、删除用户

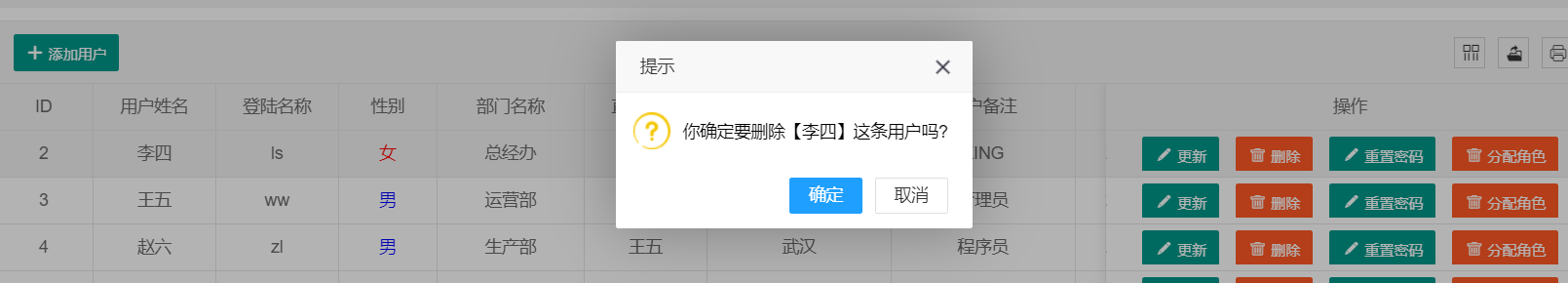


图3-2-4

### 3.3、部门管理

#### 3.3.1、查询部门

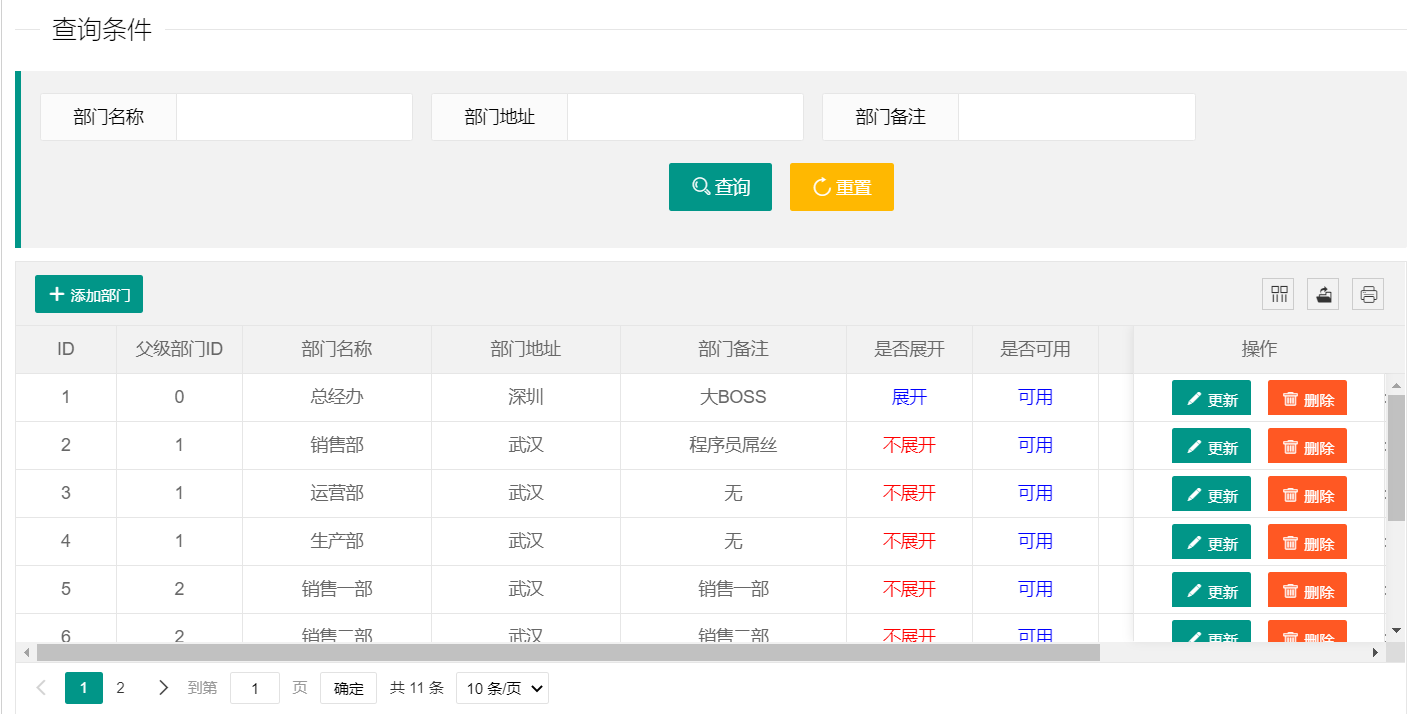


图3-3-1

#### 3.3.2、添加部门

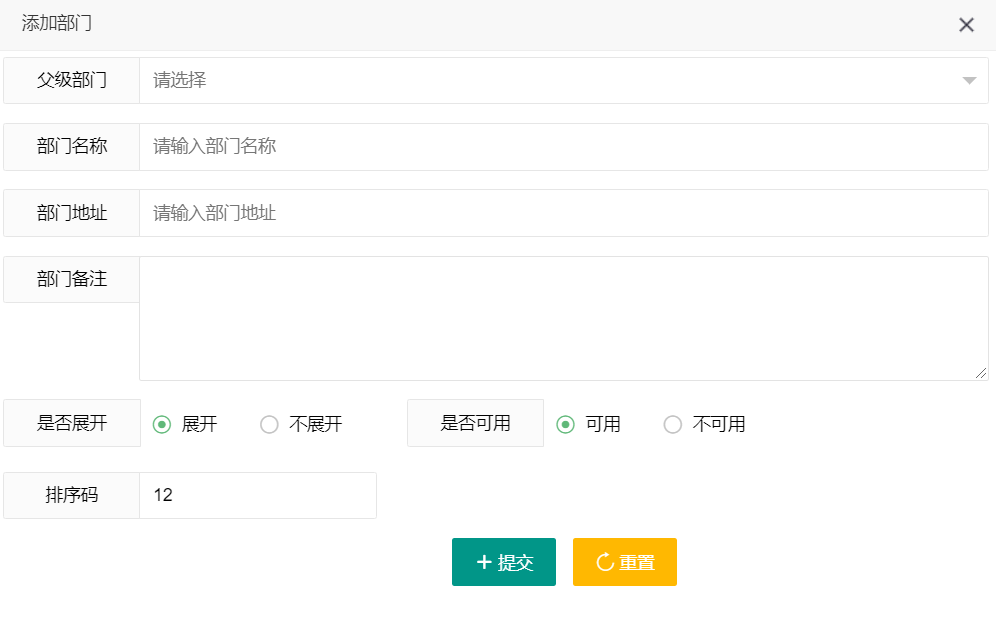


图3-3-2

#### 3.3.3、修改部门，禁用部门

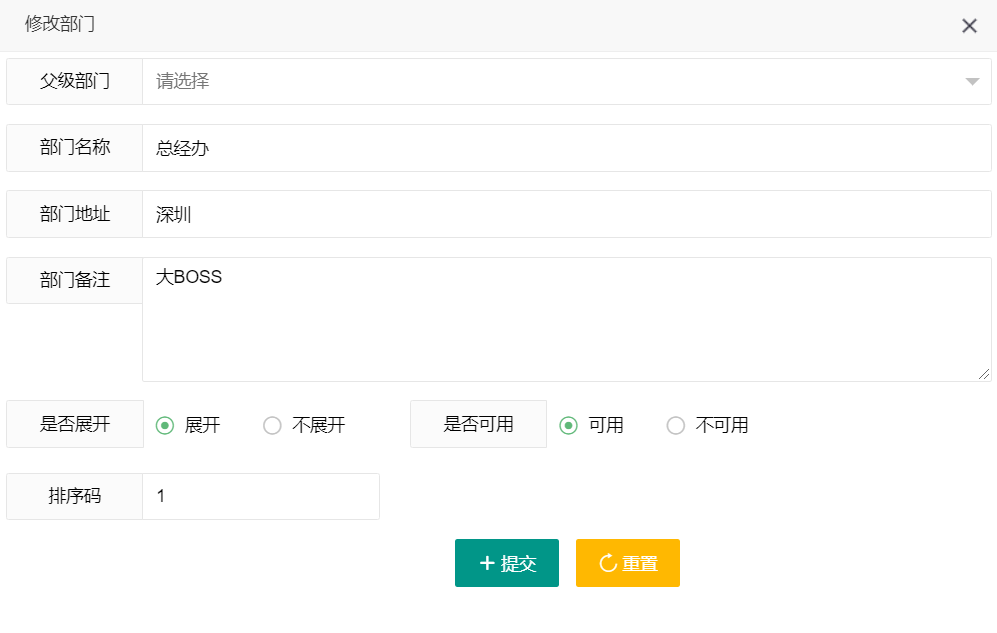


图3-3-3

#### 3.3.4、删除部门

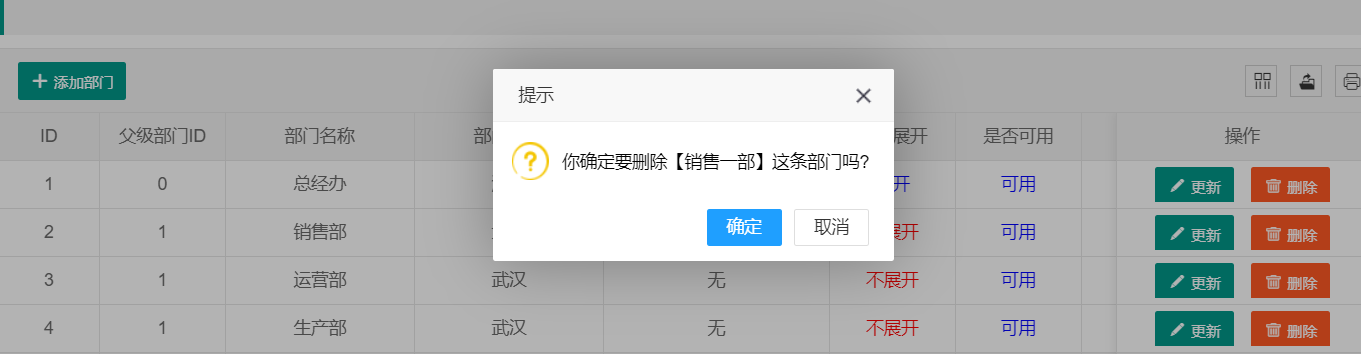


图3-3-4

### 3.4、角色管理

#### 3.4.1、查询角色

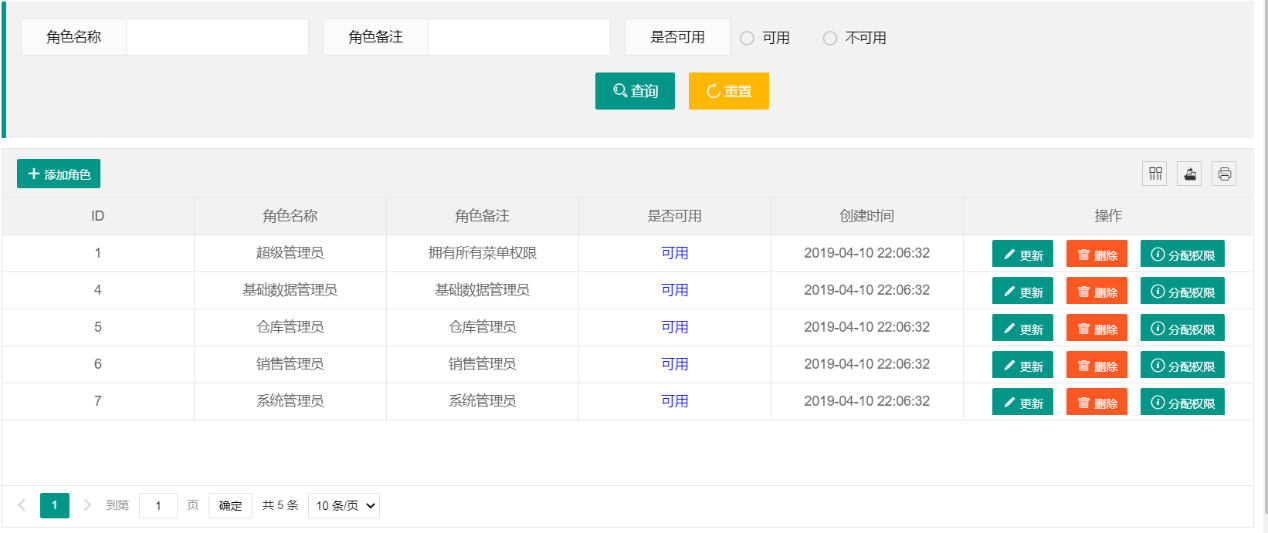


图3-4-1

#### 3.4.2、添加角色



图3-4-2

#### 3.4.3、修改角色



图3-4-3-1

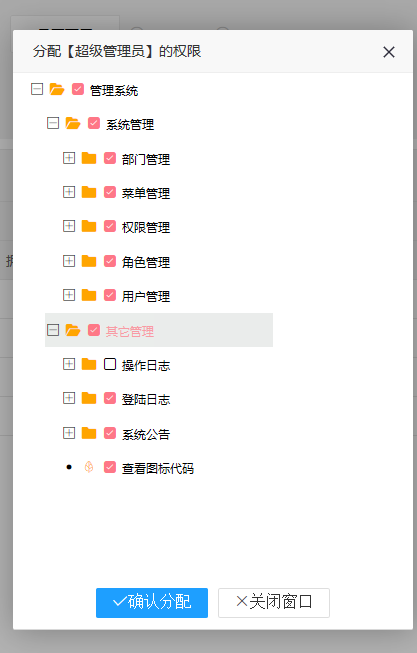


图3-4-3-2

#### 3.4.4、删除角色

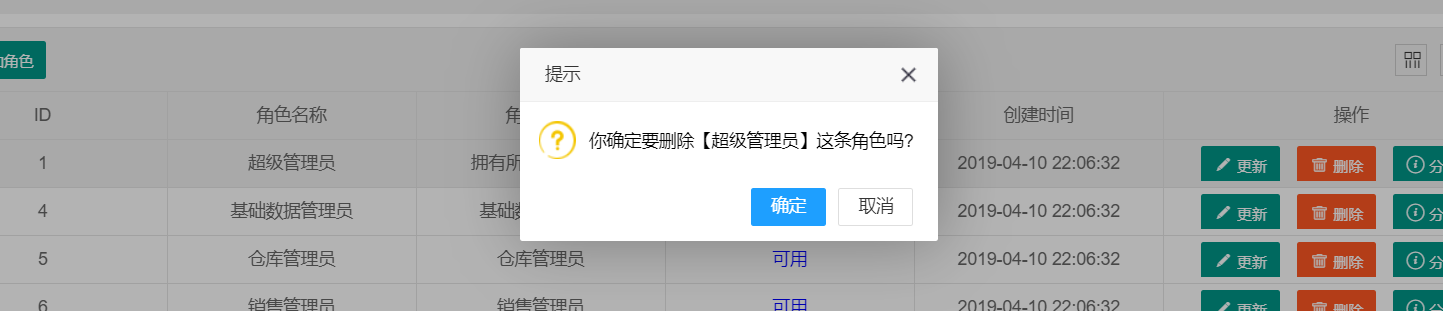


图3-4-4

### 3.5、菜单管理

#### 3.5.1、查询菜单

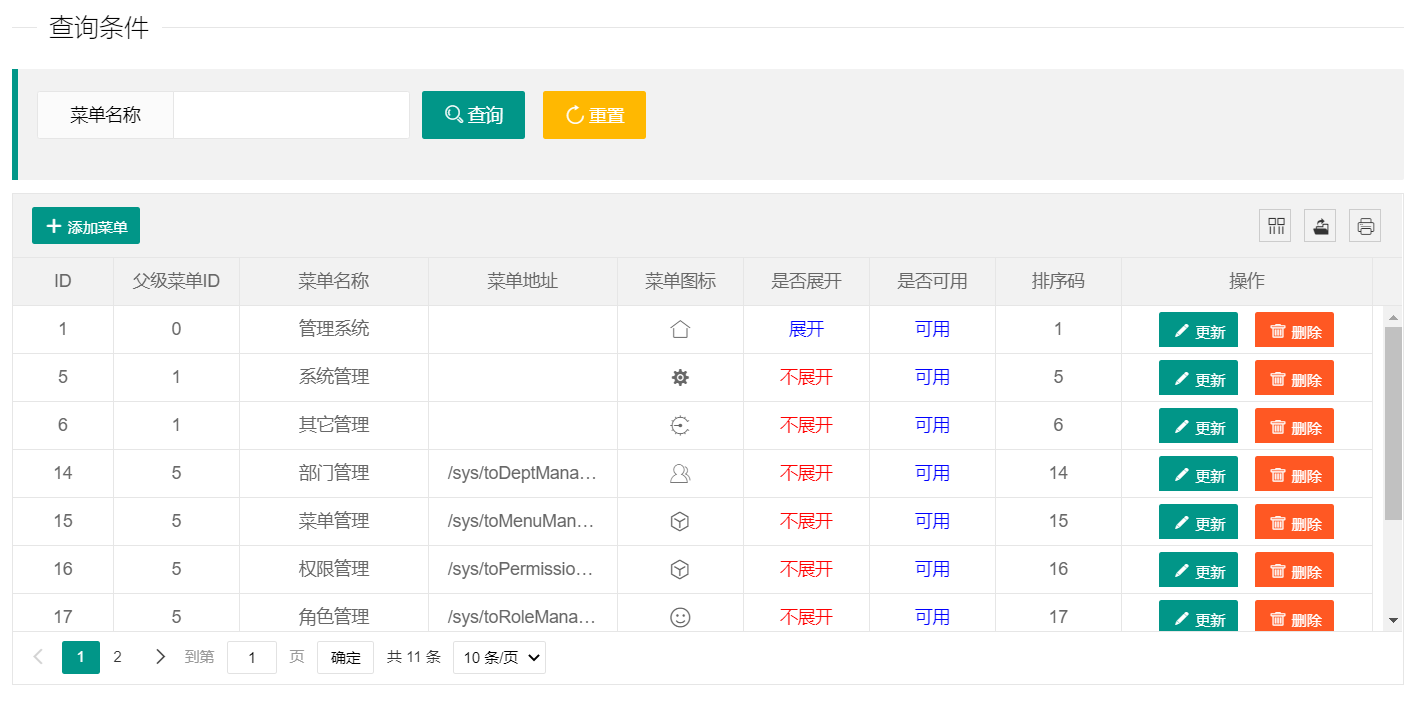


图3-5-1

#### 3.5.2、添加菜单

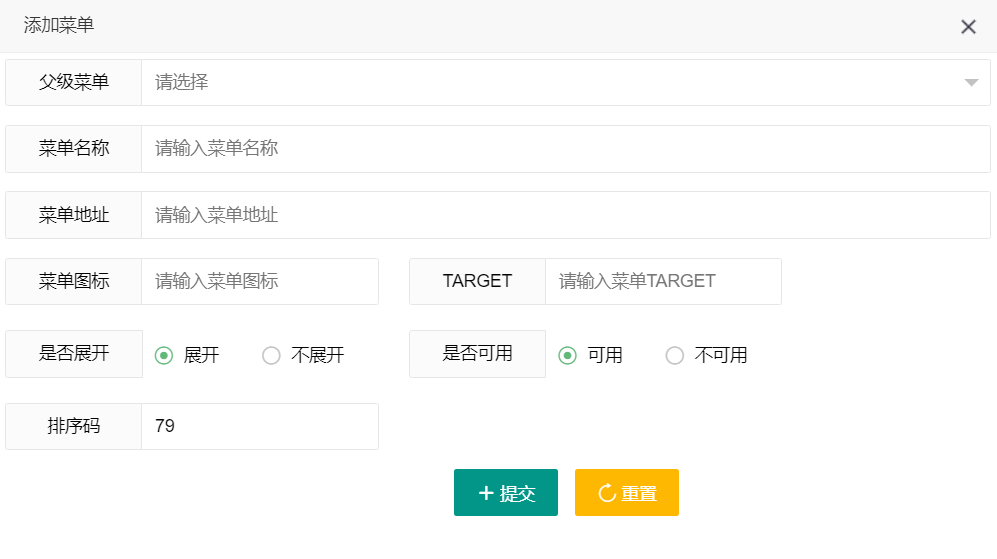


图3-5-2

#### 3.5.3、修改菜单

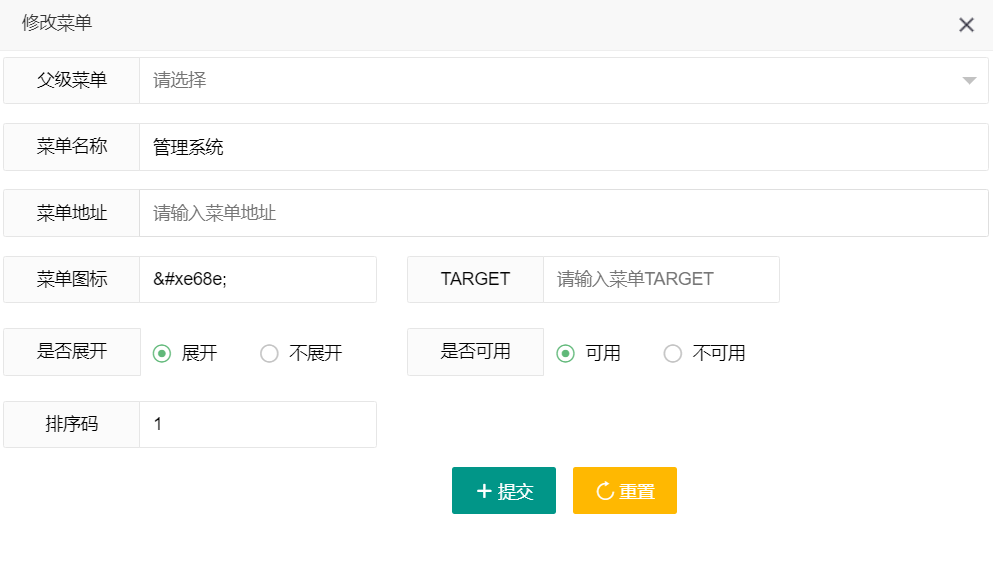


图3-5-3

#### 3.5.4、删除菜单

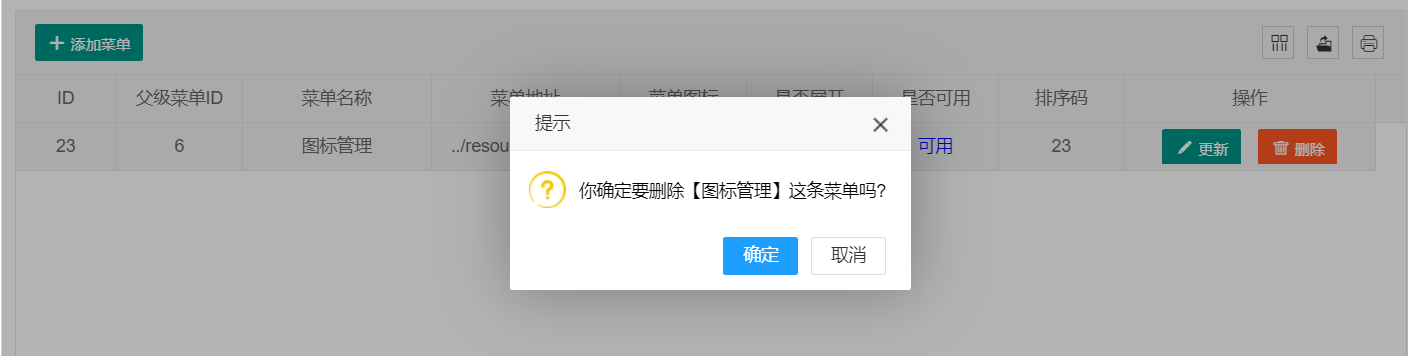


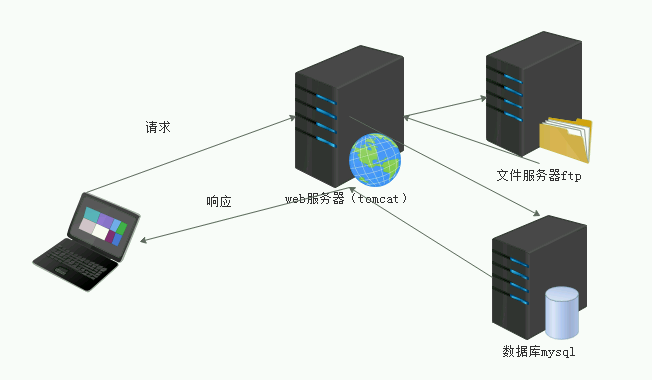
图3-5-4

## 4、原型设计

产品经理设计产品原形

## 5、架构设计

### 5.1、整体架构设计



### 5.7、数据模型设计

# 简介

## 描述

## 模型 PhysicalDataModel\_2的卡片

|  |  |
| --- | --- |
| 名称 | PhysicalDataModel\_2 |
| 代码 | PhysicalDataModel\_2 |
| DBMS | MySQL 5.0 |
| 注释 |  |
| 数据库 | aaa\_06\_trainingproject |

# 简单模型描述

## 图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

## 图PhysicalDiagram\_1



## 表的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| tbl\_dept | tbl\_dept |
| tbl\_loginfo | tbl\_loginfo |
| tbl\_notice | tbl\_notice |
| tbl\_permission | tbl\_permission |
| tbl\_role | tbl\_role |
| tbl\_role\_permission | tbl\_role\_permission |
| tbl\_role\_user | tbl\_role\_user |
| tbl\_user | tbl\_user |

## 引用的清单

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 父表格 | 子表格 | 外键列 | 父角色 | 子角色 |
| tbl\_role\_permission\_ibfk\_1 | tbl\_role\_permission\_ibfk\_1 | tbl\_permission | tbl\_role\_permission | pid |  |  |
| tbl\_role\_permission\_ibfk\_2 | tbl\_role\_permission\_ibfk\_2 | tbl\_role | tbl\_role\_permission | rid |  |  |
| tbl\_role\_user\_ibfk\_1 | tbl\_role\_user\_ibfk\_1 | tbl\_role | tbl\_role\_user | rid |  |  |
| tbl\_user\_ibfk\_1 | tbl\_user\_ibfk\_1 | tbl\_dept | tbl\_user | deptid |  |  |

# 完全模型描述

## 图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

## 图PhysicalDiagram\_1



## 表的清单

|  |  |
| --- | --- |
| *名称* | *代码* |
| tbl\_dept | tbl\_dept |
| tbl\_loginfo | tbl\_loginfo |
| tbl\_notice | tbl\_notice |
| tbl\_permission | tbl\_permission |
| tbl\_role | tbl\_role |
| tbl\_role\_permission | tbl\_role\_permission |
| tbl\_role\_user | tbl\_role\_user |
| tbl\_user | tbl\_user |

### 表tbl\_dept

#### 表tbl\_dept的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_dept |
| 代码 | tbl\_dept |

#### 表的服务器验证规则tbl\_dept

%RULES%

#### 表的选项tbl\_dept

ENGINE=InnoDB AUTO\_INCREMENT=19 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_dept

TK\_tbl\_dept

#### 表tbl\_dept的引入的引用的清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 子表格 | 外键列 | 父角色 | 子角色 |
| tbl\_user\_ibfk\_1 | tbl\_user\_ibfk\_1 | tbl\_user | deptid |  |  |

#### 包含表tbl\_dept的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_dept的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| id | id |
| pid | pid |
| title | title |
| open | open |
| remark | remark |
| address | address |
| available | available |
| ordernum | ordernum |
| createtime | createtime |

#### 表tbl\_dept的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |

### 表tbl\_loginfo

#### 表tbl\_loginfo的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_loginfo |
| 代码 | tbl\_loginfo |

#### 表的服务器验证规则tbl\_loginfo

%RULES%

#### 表的选项tbl\_loginfo

ENGINE=InnoDB AUTO\_INCREMENT=182 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_loginfo

TK\_tbl\_loginfo

#### 包含表tbl\_loginfo的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_loginfo的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| id | id |
| loginname | loginname |
| loginip | loginip |
| logintime | logintime |

#### 表tbl\_loginfo的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |

### 表tbl\_notice

#### 表tbl\_notice的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_notice |
| 代码 | tbl\_notice |

#### 表的服务器验证规则tbl\_notice

%RULES%

#### 表的选项tbl\_notice

ENGINE=InnoDB AUTO\_INCREMENT=12 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_notice

TK\_tbl\_notice

#### 包含表tbl\_notice的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_notice的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| id | id |
| title | title |
| content | content |
| createtime | createtime |
| opername | opername |

#### 表tbl\_notice的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |

### 表tbl\_permission

#### 表tbl\_permission的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_permission |
| 代码 | tbl\_permission |

#### 表的服务器验证规则tbl\_permission

%RULES%

#### 表的选项tbl\_permission

ENGINE=InnoDB AUTO\_INCREMENT=94 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_permission

TK\_tbl\_permission

#### 表tbl\_permission的引入的引用的清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 子表格 | 外键列 | 父角色 | 子角色 |
| tbl\_role\_permission\_ibfk\_1 | tbl\_role\_permission\_ibfk\_1 | tbl\_role\_permission | pid |  |  |

#### 包含表tbl\_permission的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_permission的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| id | id |
| pid | pid |
| type | type |
| title | title |
| percode | percode |
| icon | icon |
| href | href |
| target | target |
| open | open |
| ordernum | ordernum |
| available | available |

#### 表tbl\_permission的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |

### 表tbl\_role

#### 表tbl\_role的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_role |
| 代码 | tbl\_role |

#### 表的服务器验证规则tbl\_role

%RULES%

#### 表的选项tbl\_role

ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_role

TK\_tbl\_role

#### 表tbl\_role的引入的引用的清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 子表格 | 外键列 | 父角色 | 子角色 |
| tbl\_role\_permission\_ibfk\_2 | tbl\_role\_permission\_ibfk\_2 | tbl\_role\_permission | rid |  |  |
| tbl\_role\_user\_ibfk\_1 | tbl\_role\_user\_ibfk\_1 | tbl\_role\_user | rid |  |  |

#### 包含表tbl\_role的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_role的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| id | id |
| name | name |
| remark | remark |
| available | available |
| createtime | createtime |

#### 表tbl\_role的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |

### 表tbl\_role\_permission

#### 表tbl\_role\_permission的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_role\_permission |
| 代码 | tbl\_role\_permission |

#### 表的服务器验证规则tbl\_role\_permission

%RULES%

#### 表的选项tbl\_role\_permission

ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_role\_permission

TK\_tbl\_role\_permission

#### 表tbl\_role\_permission的外出的引用的清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 父表格 | 外键列 | 父角色 | 子角色 |
| tbl\_role\_permission\_ibfk\_1 | tbl\_role\_permission\_ibfk\_1 | tbl\_permission | pid |  |  |
| tbl\_role\_permission\_ibfk\_2 | tbl\_role\_permission\_ibfk\_2 | tbl\_role | rid |  |  |

#### 包含表tbl\_role\_permission的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_role\_permission的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| rid | rid |
| pid | pid |

#### 表tbl\_role\_permission的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |
| FK\_tcsr9ucxypb3ce1q5qngsfk33 | FK\_tcsr9ucxypb3ce1q5qngsfk33 | FALSE |

### 表tbl\_role\_user

#### 表tbl\_role\_user的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_role\_user |
| 代码 | tbl\_role\_user |

#### 表的服务器验证规则tbl\_role\_user

%RULES%

#### 表的选项tbl\_role\_user

ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_role\_user

TK\_tbl\_role\_user

#### 表tbl\_role\_user的外出的引用的清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 父表格 | 外键列 | 父角色 | 子角色 |
| tbl\_role\_user\_ibfk\_1 | tbl\_role\_user\_ibfk\_1 | tbl\_role | rid |  |  |

#### 包含表tbl\_role\_user的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_role\_user的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| rid | rid |
| uid | uid |

#### 表tbl\_role\_user的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |
| FK\_203gdpkwgtow7nxlo2oap5jru | FK\_203gdpkwgtow7nxlo2oap5jru | FALSE |

### 表tbl\_user

#### 表tbl\_user的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_user |
| 代码 | tbl\_user |

#### 表的服务器验证规则tbl\_user

%RULES%

#### 表的选项tbl\_user

ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC

#### 表的检验约束名称tbl\_user

TK\_tbl\_user

#### 表tbl\_user的外出的引用的清单

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 代码 | 父表格 | 外键列 | 父角色 | 子角色 |
| tbl\_user\_ibfk\_1 | tbl\_user\_ibfk\_1 | tbl\_dept | deptid |  |  |

#### 包含表tbl\_user的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 表tbl\_user的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| id | id |
| name | name |
| loginname | loginname |
| address | address |
| sex | sex |
| remark | remark |
| pwd | pwd |
| deptid | deptid |
| hiredate | hiredate |
| mgr | mgr |
| available | available |
| ordernum | ordernum |
| type | type |
| imgpath | imgpath |
| salt | salt |

#### 表tbl\_user的键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 主要的 |
| Key\_1 | Key\_1 | TRUE |
| FK\_3rrcpvho2w1mx1sfiuuyir1h | FK\_3rrcpvho2w1mx1sfiuuyir1h | FALSE |

## 引用的清单

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *名称* | *代码* | *父表格* | *子表格* | *外键列* | *父角色* | *子角色* |
| tbl\_role\_permission\_ibfk\_1 | tbl\_role\_permission\_ibfk\_1 | tbl\_permission | tbl\_role\_permission | pid |  |  |
| tbl\_role\_permission\_ibfk\_2 | tbl\_role\_permission\_ibfk\_2 | tbl\_role | tbl\_role\_permission | rid |  |  |
| tbl\_role\_user\_ibfk\_1 | tbl\_role\_user\_ibfk\_1 | tbl\_role | tbl\_role\_user | rid |  |  |
| tbl\_user\_ibfk\_1 | tbl\_user\_ibfk\_1 | tbl\_dept | tbl\_user | deptid |  |  |

### 引用tbl\_role\_permission\_ibfk\_1

#### 引用tbl\_role\_permission\_ibfk\_1的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_role\_permission\_ibfk\_1 |
| 代码 | tbl\_role\_permission\_ibfk\_1 |
| 子表格 | tbl\_role\_permission |
| 父表格 | tbl\_permission |
| 外键列 | pid |
| 父角色 |  |
| 子角色 |  |

#### 包含引用tbl\_role\_permission\_ibfk\_1的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 引用tbl\_role\_permission\_ibfk\_1的引用接入的清单

|  |  |  |
| --- | --- | --- |
| 显示的名称 | 父表格列 | 子表格列 |
| id = pid | id | pid |

### 引用tbl\_role\_permission\_ibfk\_2

#### 引用tbl\_role\_permission\_ibfk\_2的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_role\_permission\_ibfk\_2 |
| 代码 | tbl\_role\_permission\_ibfk\_2 |
| 子表格 | tbl\_role\_permission |
| 父表格 | tbl\_role |
| 外键列 | rid |
| 父角色 |  |
| 子角色 |  |

#### 包含引用tbl\_role\_permission\_ibfk\_2的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 引用tbl\_role\_permission\_ibfk\_2的引用接入的清单

|  |  |  |
| --- | --- | --- |
| 显示的名称 | 父表格列 | 子表格列 |
| id = rid | id | rid |

### 引用tbl\_role\_user\_ibfk\_1

#### 引用tbl\_role\_user\_ibfk\_1的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_role\_user\_ibfk\_1 |
| 代码 | tbl\_role\_user\_ibfk\_1 |
| 子表格 | tbl\_role\_user |
| 父表格 | tbl\_role |
| 外键列 | rid |
| 父角色 |  |
| 子角色 |  |

#### 包含引用tbl\_role\_user\_ibfk\_1的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 引用tbl\_role\_user\_ibfk\_1的引用接入的清单

|  |  |  |
| --- | --- | --- |
| 显示的名称 | 父表格列 | 子表格列 |
| id = rid | id | rid |

### 引用tbl\_user\_ibfk\_1

#### 引用tbl\_user\_ibfk\_1的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_user\_ibfk\_1 |
| 代码 | tbl\_user\_ibfk\_1 |
| 子表格 | tbl\_user |
| 父表格 | tbl\_dept |
| 外键列 | deptid |
| 父角色 |  |
| 子角色 |  |

#### 包含引用tbl\_user\_ibfk\_1的图的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| PhysicalDiagram\_1 | PhysicalDiagram\_1 |

#### 引用tbl\_user\_ibfk\_1的引用接入的清单

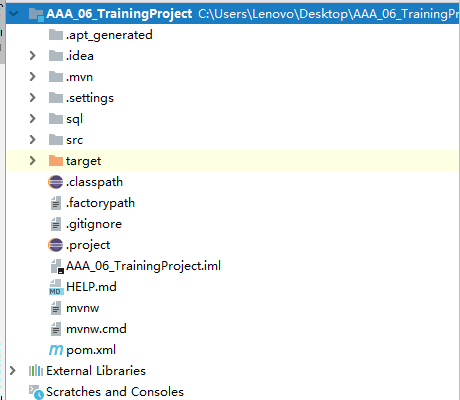
|  |  |  |
| --- | --- | --- |
| 显示的名称 | 父表格列 | 子表格列 |
| id = deptid | id | deptid |

## 代码开发

# 创建数据库

# 集成项目

## 创建项目



# 用户登陆

### 生成

User UserMapper UserMapper.xmlUserServiceUserServiceImplUserController

### 创建ActiverUser

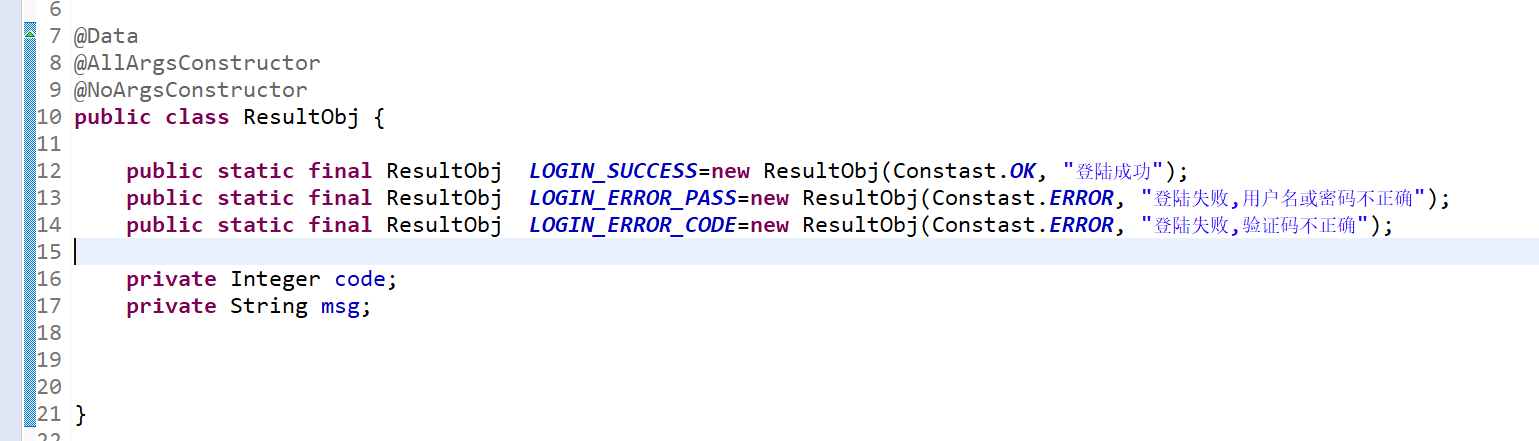


### 创建UserRealm

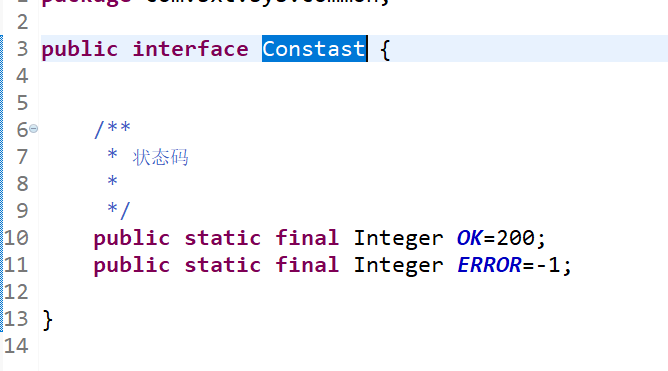
### 创建ShiroAutoConfigruation

|  |
| --- |
| /\*\*  \*  \* **@author** LJH  \*  \*/  @Configuration  @ConditionalOnWebApplication(type = Type.***SERVLET***)  @ConditionalOnClass(value = { SecurityManager.**class** })  @ConfigurationProperties(prefix = "shiro")  @Data  **publicclass**ShiroAutoConfiguration {  **privatestaticfinal** String ***SHIRO\_DIALECT*** = "shiroDialect";  **privatestaticfinal** String ***SHIRO\_FILTER*** = "shiroFilter";  **private** String hashAlgorithmName = "md5";// 加密方式  **privateint**hashIterations = 2;// 散列次数  **private** String loginUrl = "/index.html";// 默认的登陆页面  **private**String[] anonUrls;  **private** String logOutUrl;  **private**String[] authcUlrs;  /\*\*  \* 声明凭证匹配器  \*/  @Bean("credentialsMatcher")  **public**HashedCredentialsMatcherhashedCredentialsMatcher() {  HashedCredentialsMatchercredentialsMatcher = **new**HashedCredentialsMatcher();  credentialsMatcher.setHashAlgorithmName(hashAlgorithmName);  credentialsMatcher.setHashIterations(hashIterations);  **return**credentialsMatcher;  }  /\*\*  \* 声明userRealm  \*/  @Bean("userRealm")  **public**UserRealmuserRealm(CredentialsMatchercredentialsMatcher) {  UserRealmuserRealm = **new**UserRealm();  // 注入凭证匹配器  userRealm.setCredentialsMatcher(credentialsMatcher);  **return**userRealm;  }  /\*\*  \* 配置SecurityManager  \*/  @Bean("securityManager")  **public**SecurityManagersecurityManager(UserRealmuserRealm) {  DefaultWebSecurityManagersecurityManager = **new**DefaultWebSecurityManager();  // 注入userRealm  securityManager.setRealm(userRealm);  **return**securityManager;  }  /\*\*  \* 配置shiro的过滤器  \*/  @Bean(***SHIRO\_FILTER***)  **public**ShiroFilterFactoryBeanshiroFilterFactoryBean(SecurityManagersecurityManager) {  ShiroFilterFactoryBeanfactoryBean = **new**ShiroFilterFactoryBean();  // 设置安全管理器  factoryBean.setSecurityManager(securityManager);  // 设置未登陆的时要跳转的页面  factoryBean.setLoginUrl(loginUrl);  Map<String, String>filterChainDefinitionMap = **new** HashMap<>();  // 设置放行的路径  **if** (anonUrls != **null**&&anonUrls.length> 0) {  **for** (String anon :anonUrls) {  filterChainDefinitionMap.put(anon, "anon");  }  }  // 设置登出的路径  **if** (**null** != logOutUrl) {  filterChainDefinitionMap.put(logOutUrl, "logout");  }  // 设置拦截的路径  **if** (authcUlrs != **null**&&authcUlrs.length> 0) {  **for** (String authc :authcUlrs) {  filterChainDefinitionMap.put(authc, "authc");  }  }  Map<String, Filter>filters=**new** HashMap<>();  // filters.put("authc", new ShiroLoginFilter());  //配置过滤器  factoryBean.setFilters(filters);  factoryBean.setFilterChainDefinitionMap(filterChainDefinitionMap);  **return**factoryBean;  }  /\*\*  \* 注册shiro的委托过滤器，相当于之前在web.xml里面配置的  \*  \* **@return**  \*/  @Bean  **public**FilterRegistrationBean<DelegatingFilterProxy>delegatingFilterProxy() {  FilterRegistrationBean<DelegatingFilterProxy>filterRegistrationBean = **new**FilterRegistrationBean<DelegatingFilterProxy>();  DelegatingFilterProxyproxy = **new**DelegatingFilterProxy();  proxy.setTargetFilterLifecycle(**true**);  proxy.setTargetBeanName(***SHIRO\_FILTER***);  filterRegistrationBean.setFilter(proxy);  **return**filterRegistrationBean;  }  /\* 加入注解的使用，不加入这个注解不生效--开始 \*/  /\*\*  \*  \* **@param**securityManager  \* **@return**  \*/  @Bean  **public**AuthorizationAttributeSourceAdvisorauthorizationAttributeSourceAdvisor(SecurityManagersecurityManager) {  AuthorizationAttributeSourceAdvisorauthorizationAttributeSourceAdvisor = **new**AuthorizationAttributeSourceAdvisor();  authorizationAttributeSourceAdvisor.setSecurityManager(securityManager);  **return**authorizationAttributeSourceAdvisor;  }  @Bean  **public**DefaultAdvisorAutoProxyCreatorgetDefaultAdvisorAutoProxyCreator() {  DefaultAdvisorAutoProxyCreatoradvisorAutoProxyCreator = **new**DefaultAdvisorAutoProxyCreator();  advisorAutoProxyCreator.setProxyTargetClass(**true**);  **return**advisorAutoProxyCreator;  }  /\* 加入注解的使用，不加入这个注解不生效--结束 \*/  /\*\*  \* 这里是为了能在html页面引用shiro标签，上面两个函数必须添加，不然会报错  \*  \* **@return**  \*/  @Bean(name = ***SHIRO\_DIALECT***)  **public**ShiroDialectshiroDialect() {  **returnnew**ShiroDialect();  }  } |

### 创建ResultObj



### 创建Constast



### 创建SystemController

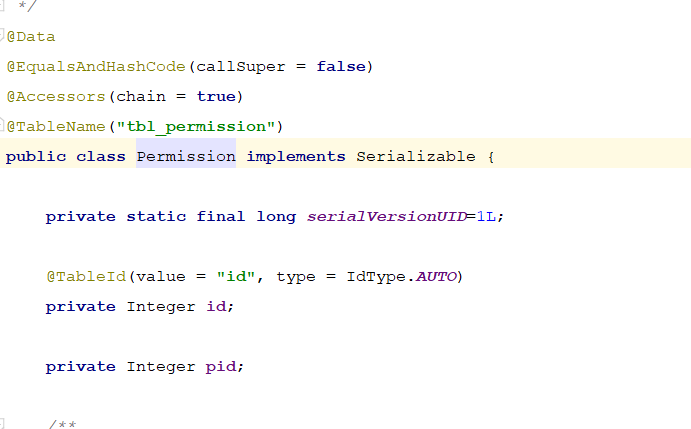


### 创建LoginController



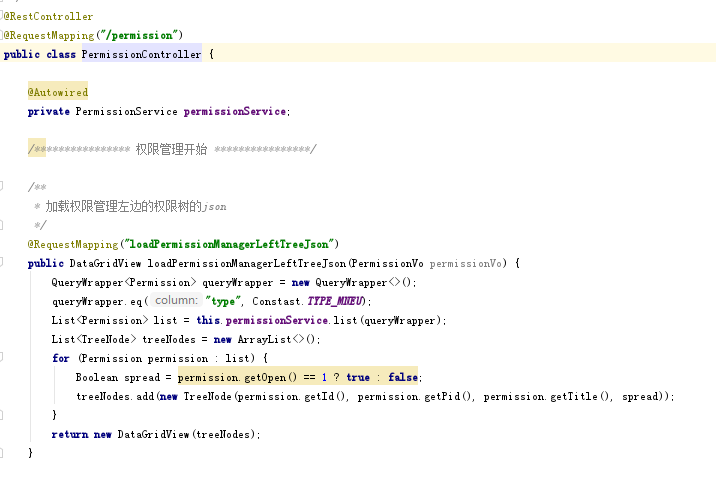
# 加载系统主页的左边的导航树

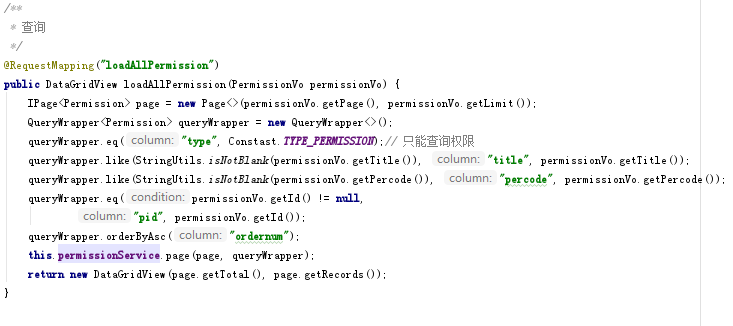
## 生成Permission



## 生成PermissionMapper,PermissionMapper.xml,PermissionService ,PermissionServiceImpl

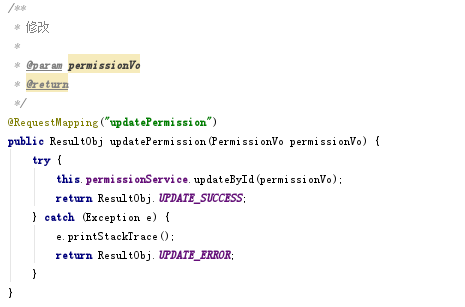
## 生成PermissionController









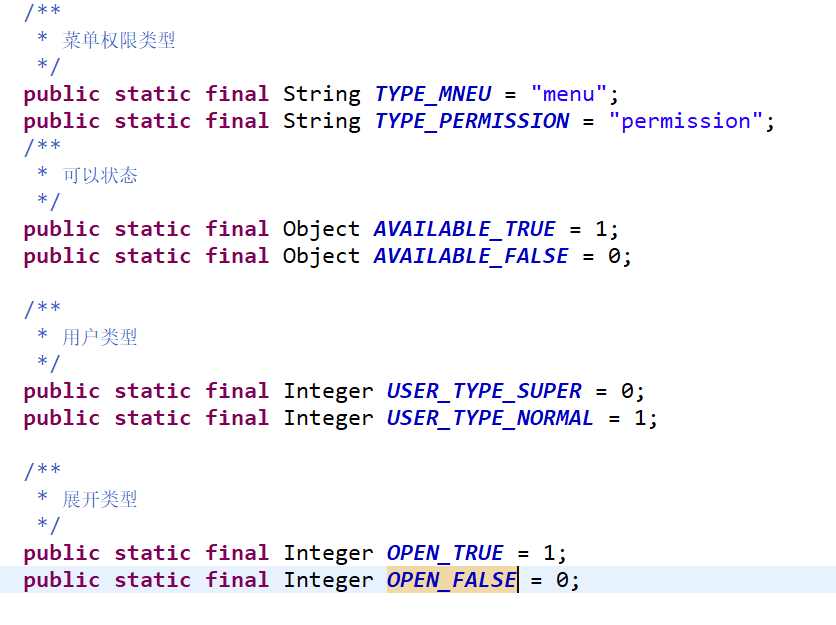




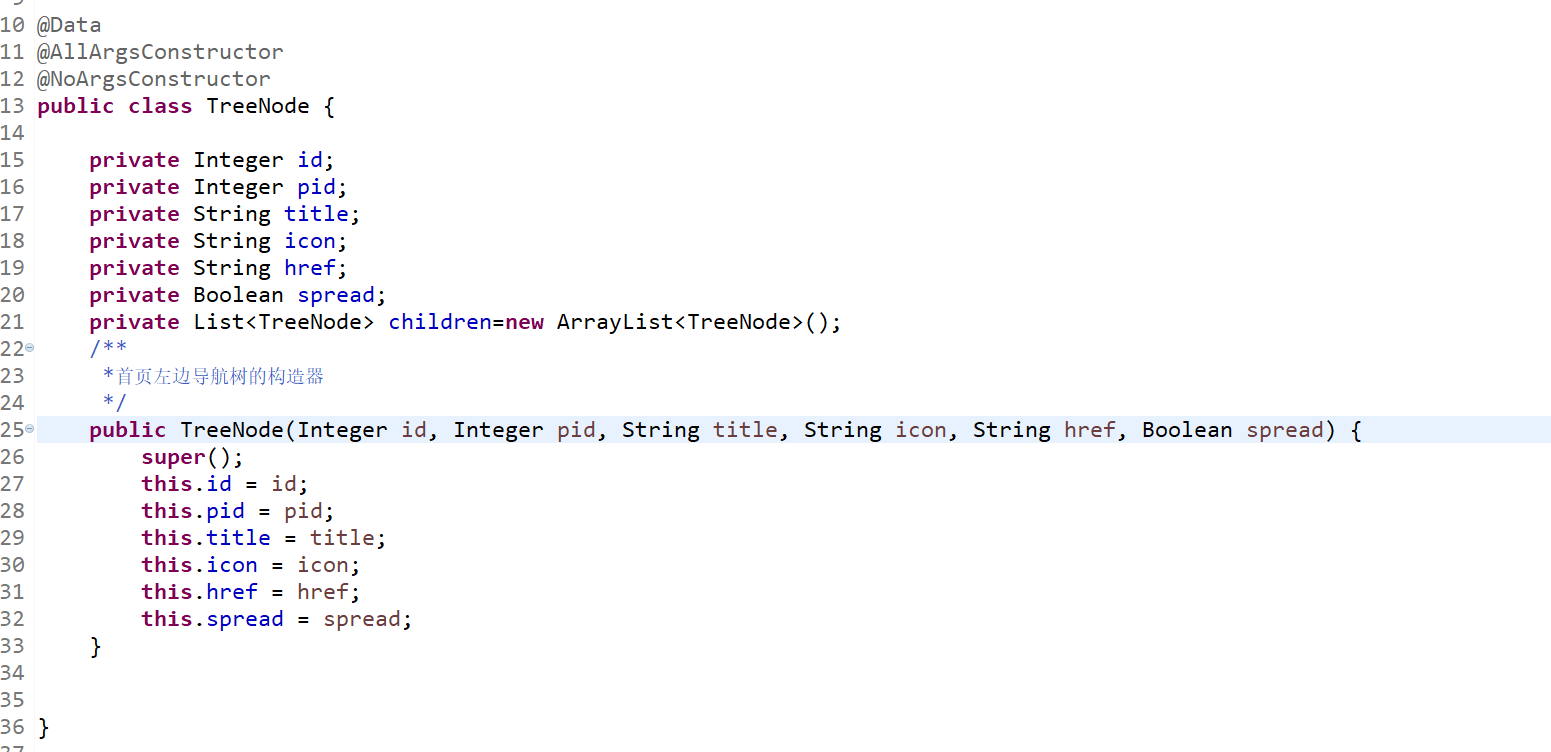
## 创建DataGridView



## 修改Constast



## 创建TreeNode



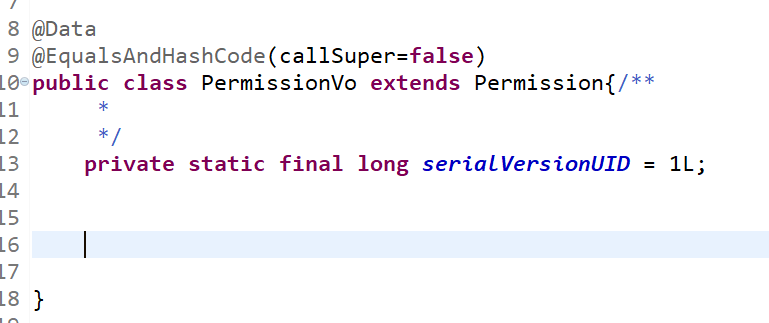
## 创建TreeNodeBuilder



## 创建MenuController

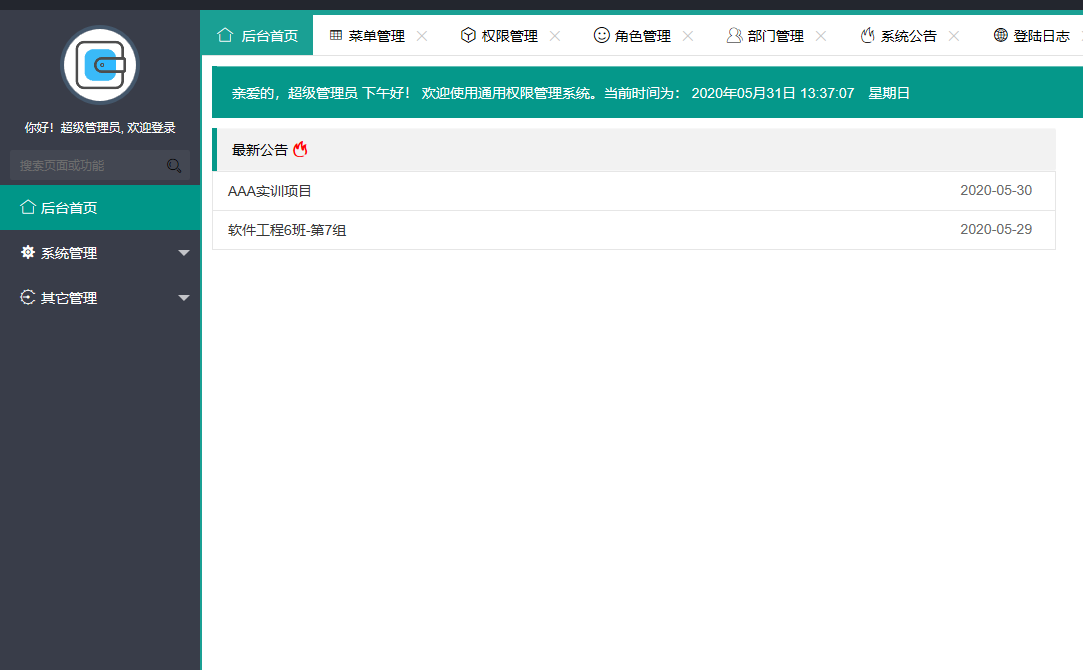
|  |
| --- |
| /\*\*  \* <p>  \* 菜单管理前端控制器  \* </p>  \*  \*/  @RestController  @RequestMapping("/menu")  **publicclass**MenuController {      @Autowired  **private**PermissionServicepermissionService;      @RequestMapping("loadIndexLeftMenuJson")  **public**DataGridViewloadIndexLeftMenuJson(PermissionVopermissionVo) {  //查询所有菜单  QueryWrapper<Permission>queryWrapper=**new**QueryWrapper<>();  //设置只能查询菜单  queryWrapper.eq("type",Constast.***TYPE\_MNEU***);  queryWrapper.eq("available", Constast.***AVAILABLE\_TRUE***);    User user = (User) WebUtils.*getSession*().getAttribute("user");  List<Permission>list=**null**;  **if**(user.getType()==Constast.***USER\_TYPE\_SUPER***) {  list = permissionService.list(queryWrapper);  }**else** {  //根据用户ID+角色+权限去查询  list = permissionService.list(queryWrapper);  }    List<TreeNode>treeNodes=**new**ArrayList<>();  **for** (Permission p :list) {  Integer id=p.getId();  Integer pid=p.getPid();  String title=p.getTitle();  String icon=p.getIcon();  String href=p.getHref();  Boolean spread=p.getOpen()==Constast.***OPEN\_TRUE***?**true**:**false**;  treeNodes.add(**new**TreeNode(id, pid, title, icon, href, spread));  }  //构造层级关系  List<TreeNode>list2 = TreeNodeBuilder.*build*(treeNodes, 1);  **returnnew**DataGridView(list2);  }    } |

## 创建PermissionVo



# 工作台的管理

## 创建deskManager.html



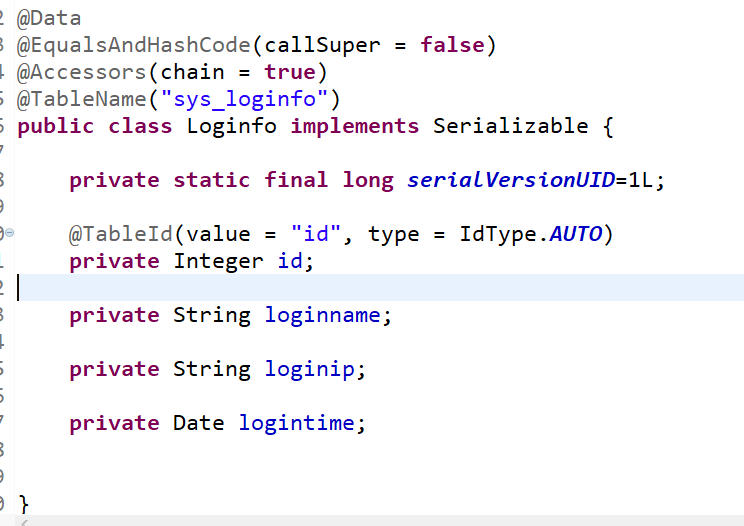
## 修改SystemController



# 日志管理

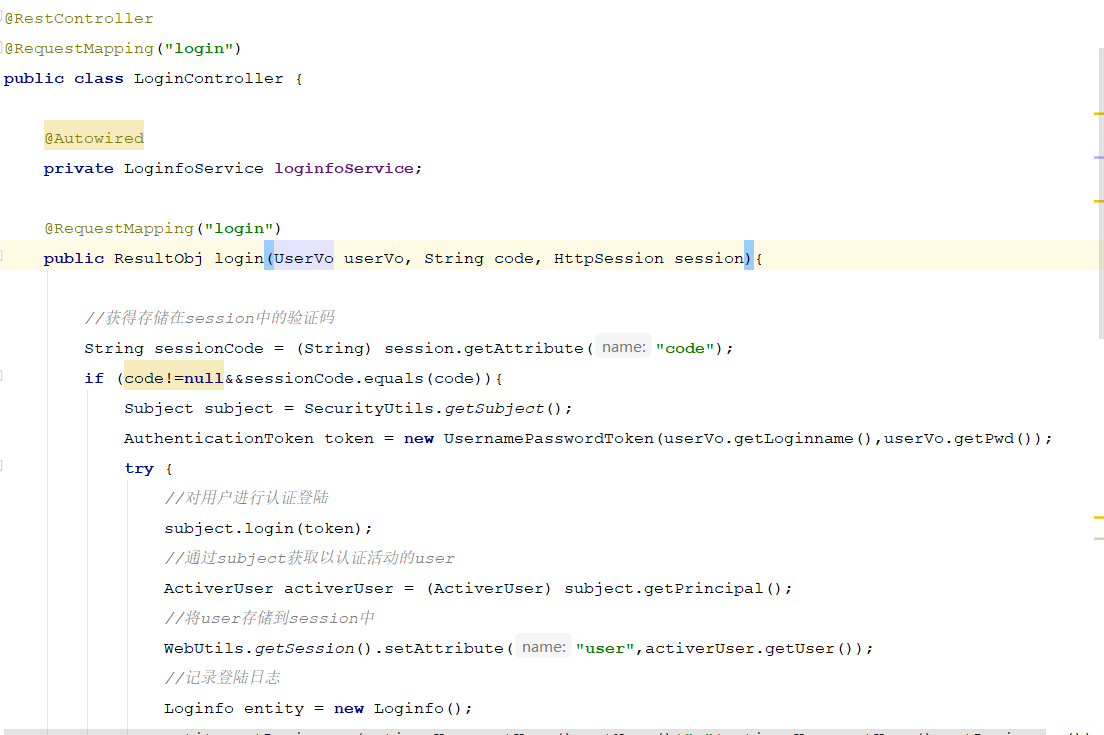
## 查询

### 生成Loginfo

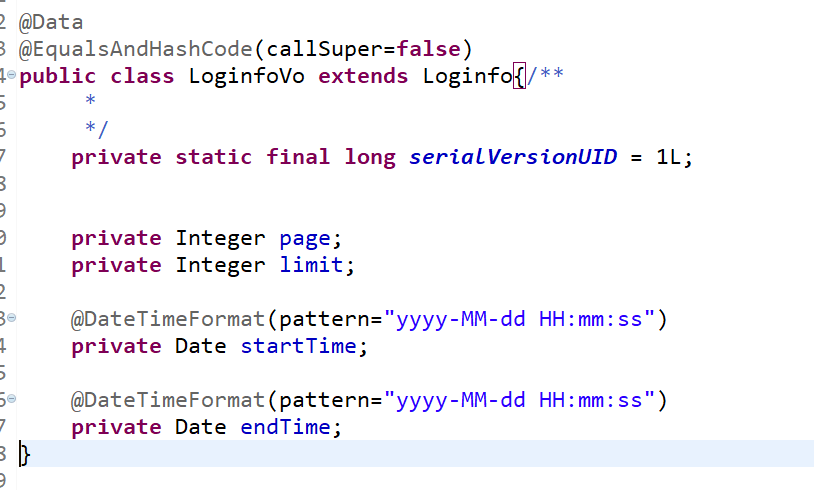


### 生成LoginfoMapper， LoginfoMapper.xml， LoginfoService， LoginfoServiceImpl

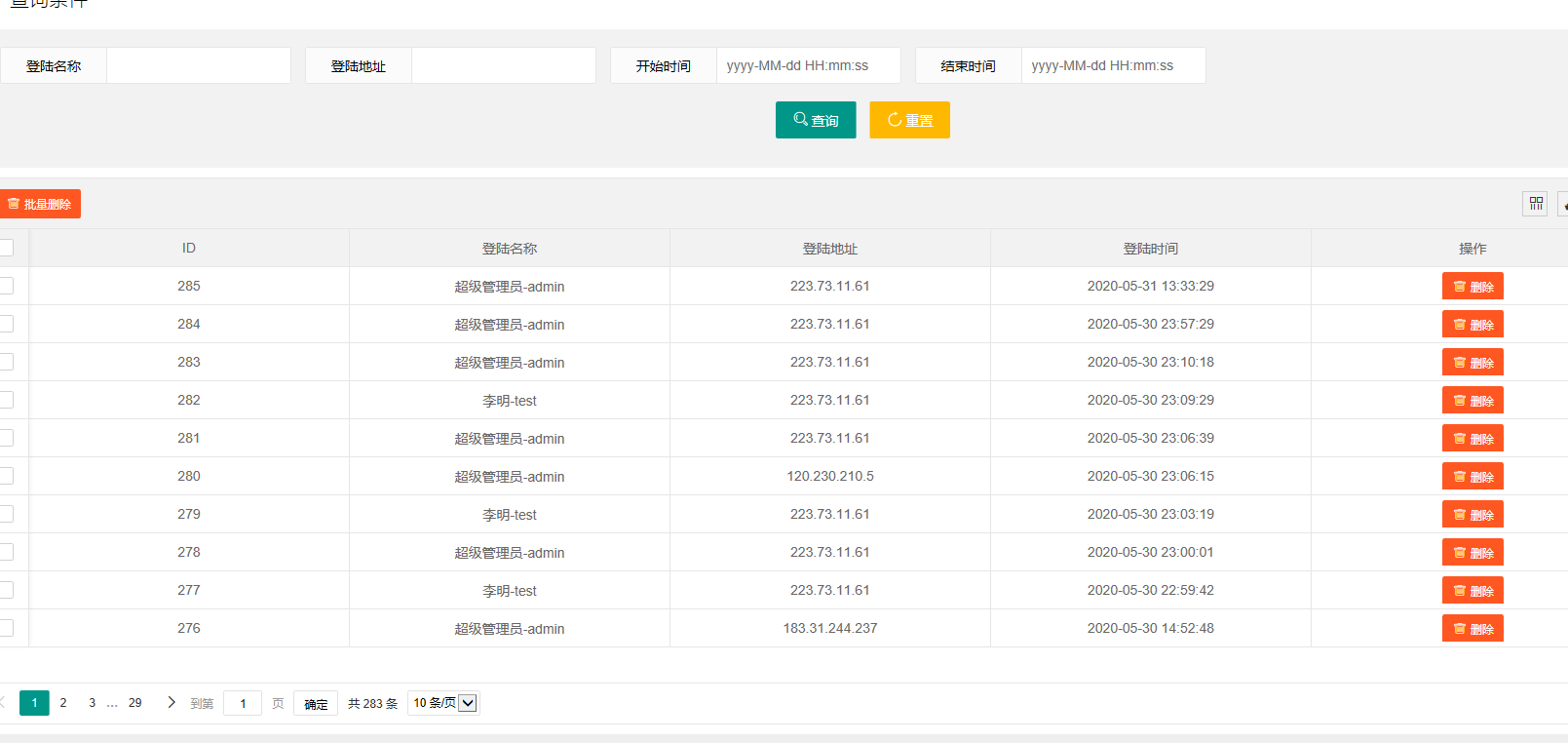
### 生成LoginfoController



### 创建LoginfoVo



### 创建system/loginfo/loginfoManager.html

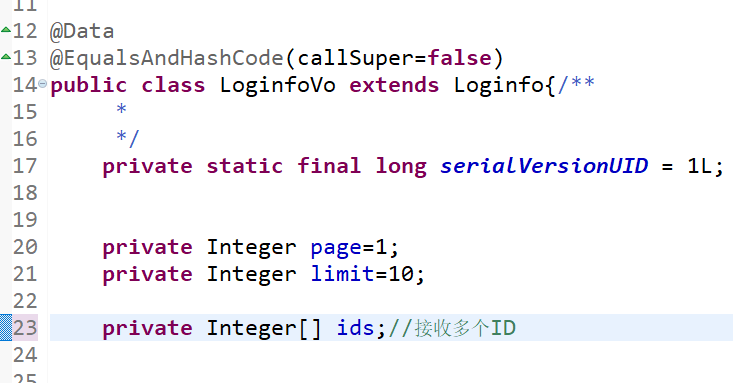


### 修改菜单地址



## 删除和批量删除

### 修改loginfoVo



### 修改ResultObj



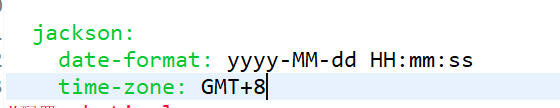
### 修改LogInfoController



## 记录登陆日志

### 修改LoginController

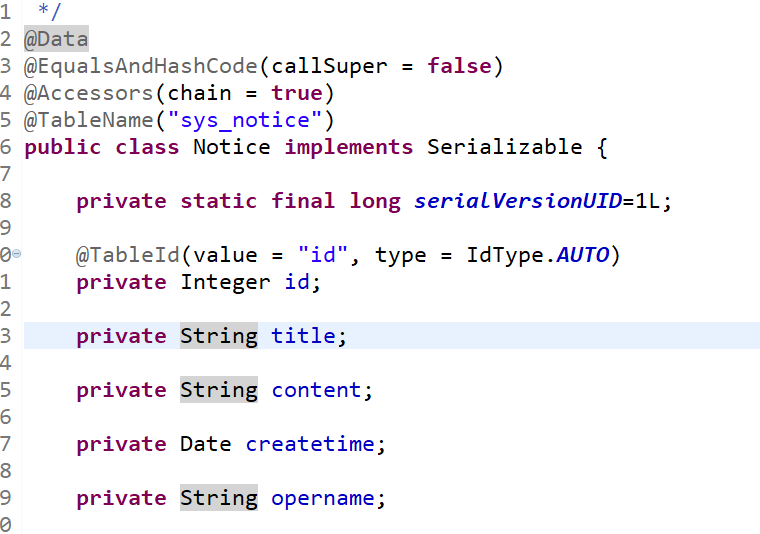




# 公告管理

## 查询

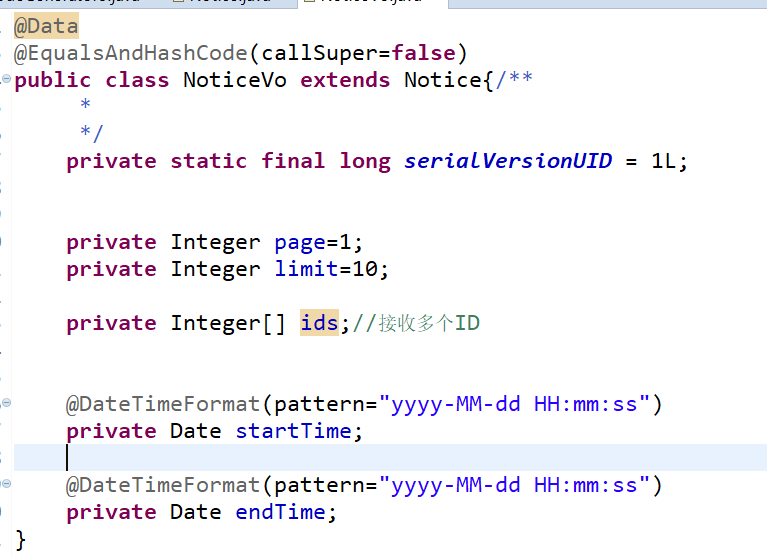
### 生成Notice



### 生成

### NoticeMapper,NoticeMapper.xml,NoticeService,NoticeServiceImpl

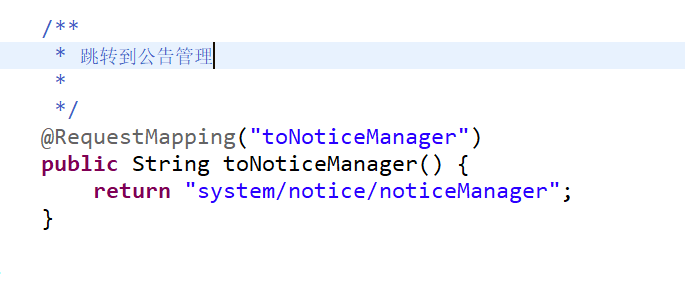
### 创建NoticeVo



### 生成NoticeController



### 修改SystemController



### 修改菜单地址



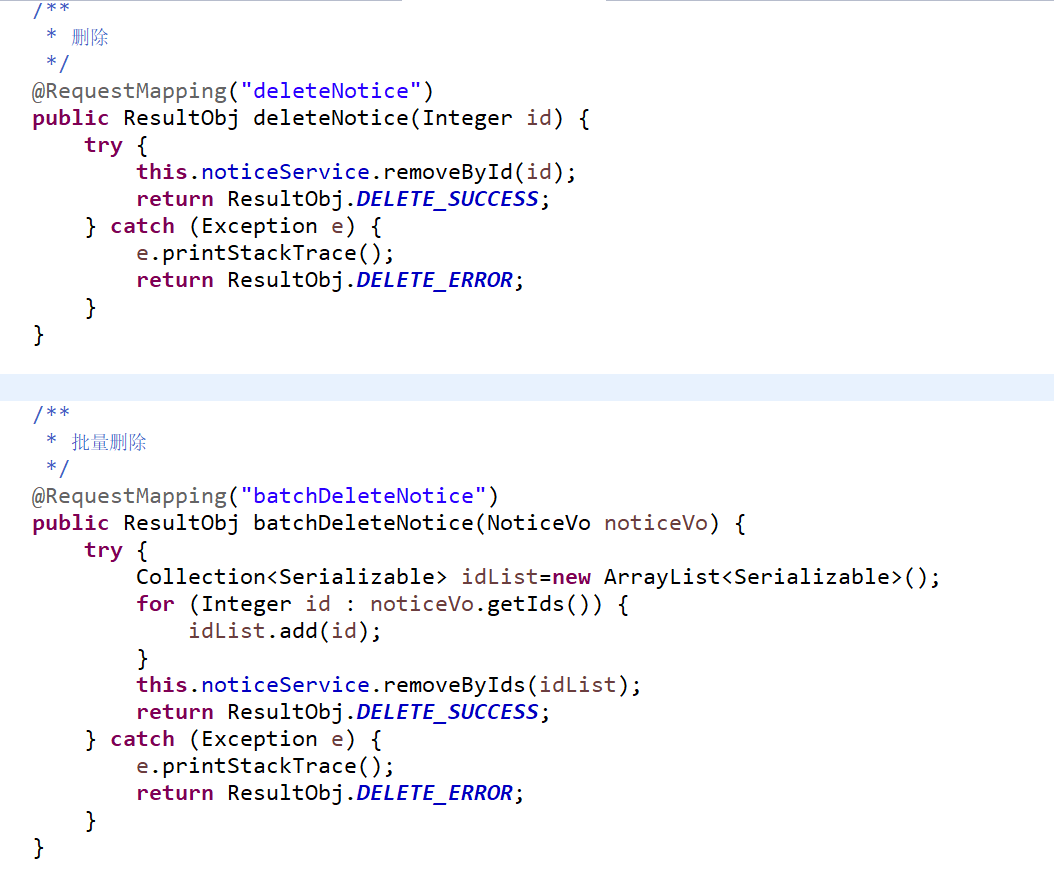
### 创建system/notice/noticeManager.html



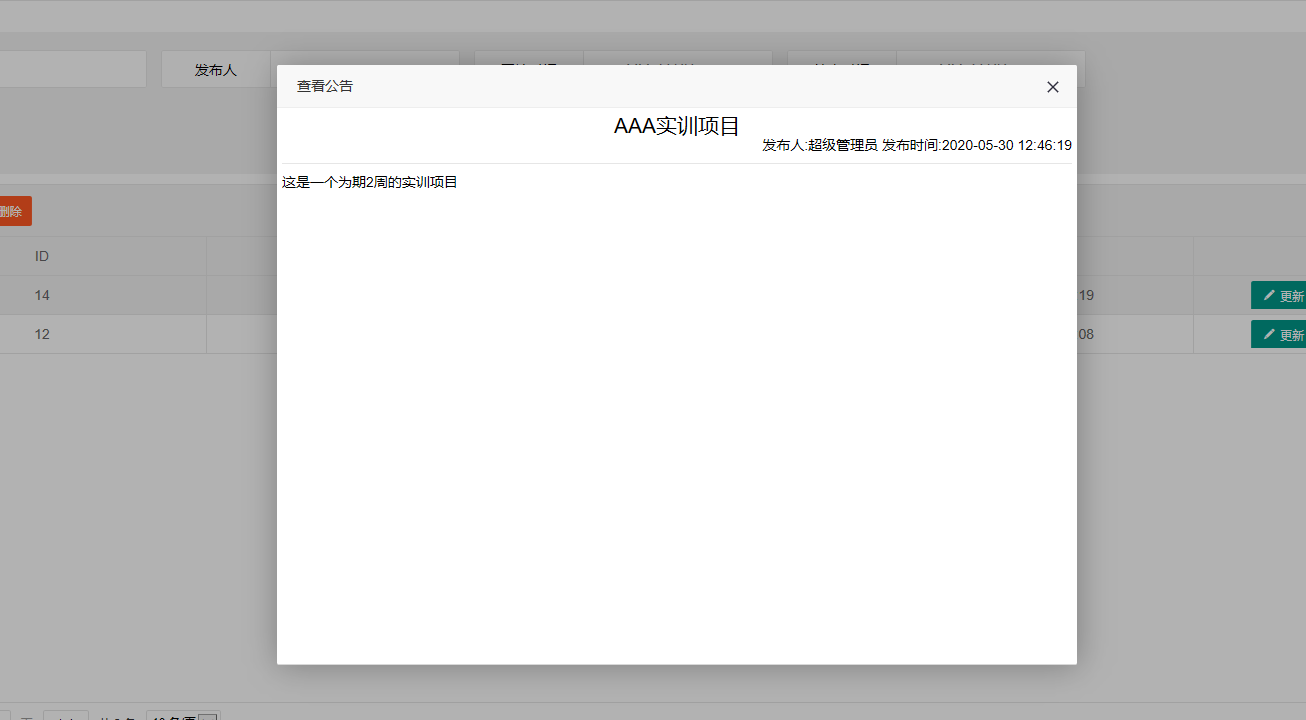
## 添加修改

### 修改NoticeController





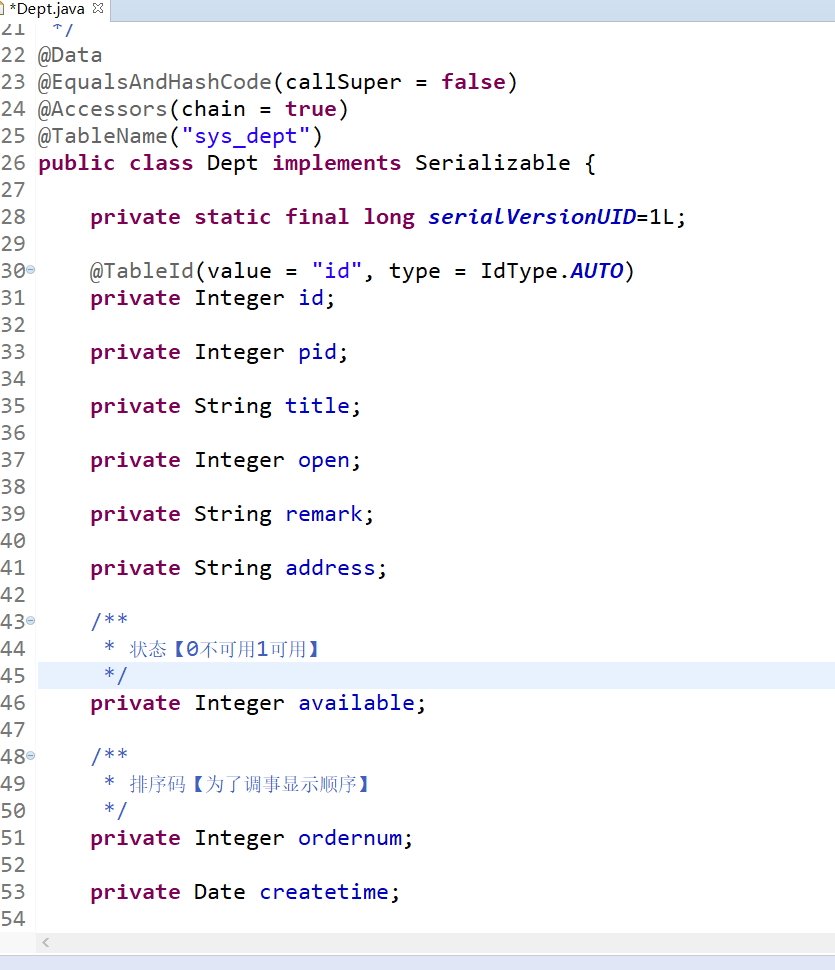
## 查看



# 部门管理

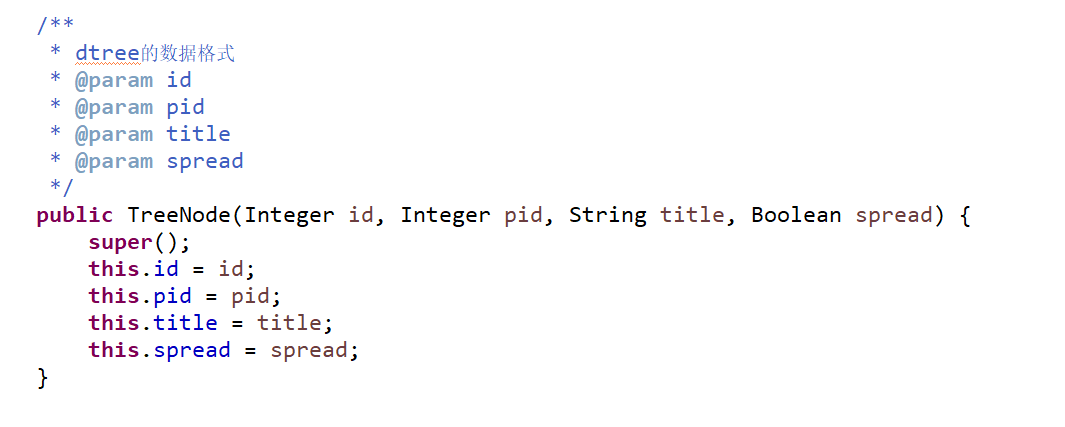
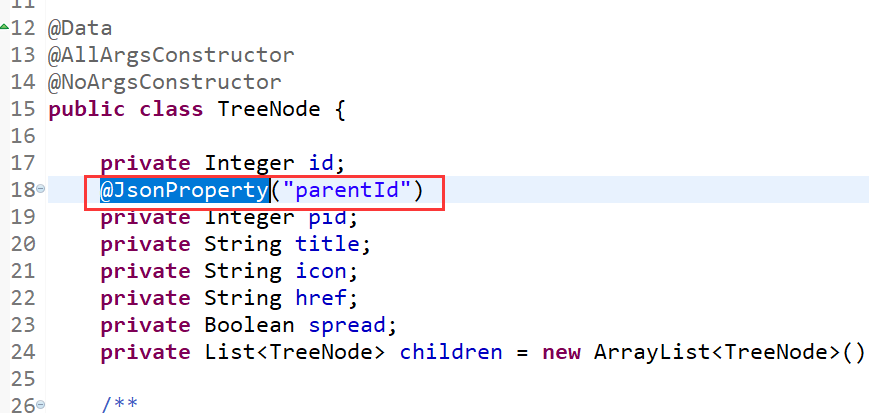
## 查询

### 生成Dept



### 生成DeptMapper DeptMapper.xml DeptService,DeptService,DeptServiceImpl

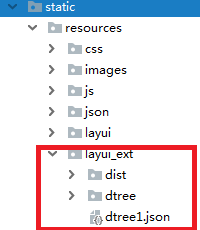
### 修改TreeNode



### 创建DeptController



### 引入dtree



### 修改SystemController



### 修改菜单地址



## 添加修改

### 修改DeptController



## 删除

### 修改DeptController



## 缓存处理

### 创建CacheAspect

|  |
| --- |
| @Aspect  @Component  @EnableAspectJAutoProxy  **publicclass**CacheAspect {    //声明一个缓存容器  **private** Map<String,Object>CACHE\_CONTAINER=**new** HashMap<>();      //声明切面表达式  **privatestaticfinal** String ***POINTCUT\_DEPT\_UPDATE***="execution(\* com.sxt.sys.service.impl.DeptServiceImpl.updateById(..))";  **privatestaticfinal** String ***POINTCUT\_DEPT\_GET***="execution(\* com.sxt.sys.service.impl.DeptServiceImpl.getOne(..))";  **privatestaticfinal** String ***POINTCUT\_DEPT\_DELETE***="execution(\* com.sxt.sys.service.impl.DeptServiceImpl.removeById(..))";      **privatestaticfinal** String ***CACHE\_DEPT\_PROFIX***="dept:";      /\*\*  \* 查询切入  \* **@throws** Throwable  \*/  @Around(value=***POINTCUT\_DEPT\_GET***)  **public** Object cacheDeptGet(ProceedingJoinPointjoinPoint) **throws** Throwable {  //取出第一个参数  Integer object = (Integer) joinPoint.getArgs()[0];  //从缓存里面取  Object res1 = CACHE\_CONTAINER.get(***CACHE\_DEPT\_PROFIX***+object);  **if**(res1!=**null**) {  **return**res1;  }**else** {  Dept res2 = (Dept) joinPoint.proceed();  CACHE\_CONTAINER.put(***CACHE\_DEPT\_PROFIX***+res2.getId(), res2);  **return**res2;  }  }  /\*\*  \* 更新切入  \* **@throws** Throwable  \*/  @Around(value=***POINTCUT\_DEPT\_UPDATE***)  **public** Object cacheDeptUpdate(ProceedingJoinPointjoinPoint) **throws** Throwable {  //取出第一个参数  DeptVodeptVo = (DeptVo) joinPoint.getArgs()[0];  Boolean isSuccess =(Boolean) joinPoint.proceed();  **if**(isSuccess) {  Dept dept = (Dept) CACHE\_CONTAINER.get(***CACHE\_DEPT\_PROFIX***+deptVo.getId());  **if**(**null**==dept) {  dept=**new**Dept();    }  BeanUtils.*copyProperties*(deptVo, dept);  CACHE\_CONTAINER.put(***CACHE\_DEPT\_PROFIX***+dept.getId(), dept);  }  **return**isSuccess;  }    /\*\*  \* 删除切入  \* **@throws** Throwable  \*/  @Around(value=***POINTCUT\_DEPT\_DELETE***)  **public** Object cacheDeptDelete(ProceedingJoinPointjoinPoint) **throws** Throwable {  //取出第一个参数  Integer id = (Integer) joinPoint.getArgs()[0];  Boolean isSuccess =(Boolean) joinPoint.proceed();  **if**(isSuccess) {  //删除缓存  CACHE\_CONTAINER.remove(***CACHE\_DEPT\_PROFIX***+id);  }  **return**isSuccess;  }    } |

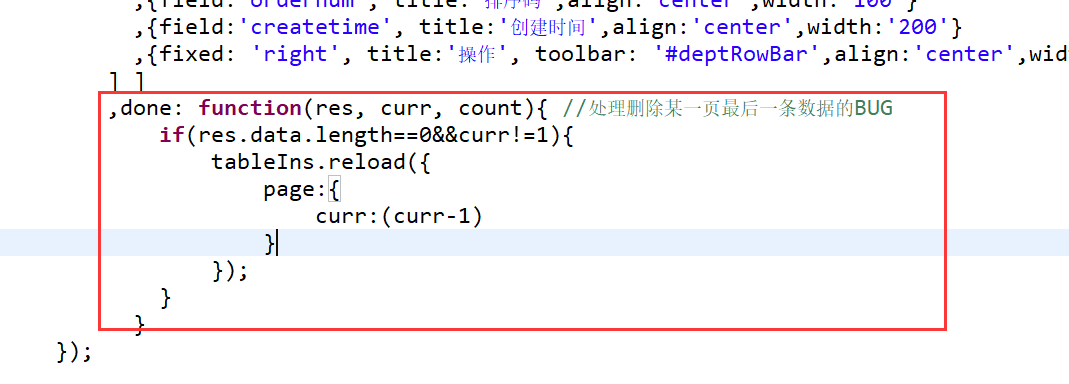
# 菜单管理CRUD

## 修改MenuController

|  |
| --- |
| /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单管理开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/    /\*\*  \* 加载菜单管理左边的菜单树的json  \*/  @RequestMapping("loadMenuManagerLeftTreeJson")  **public**DataGridViewloadMenuManagerLeftTreeJson(PermissionVopermissionVo) {  QueryWrapper<Permission>queryWrapper=**new**QueryWrapper<>();  queryWrapper.eq("type", Constast.***TYPE\_MNEU***);  List<Permission>list = **this**.permissionService.list(queryWrapper);  List<TreeNode>treeNodes=**new**ArrayList<>();  **for** (Permission menu :list) {  Boolean spread=menu.getOpen()==1?**true**:**false**;  treeNodes.add(**new**TreeNode(menu.getId(), menu.getPid(), menu.getTitle(), spread));  }  **returnnew**DataGridView(treeNodes);  }    /\*\*  \* 查询  \*/  @RequestMapping("loadAllMenu")  **public**DataGridViewloadAllMenu(PermissionVopermissionVo) {  IPage<Permission>page=**new** Page<>(permissionVo.getPage(), permissionVo.getLimit());  QueryWrapper<Permission>queryWrapper=**new**QueryWrapper<>();  queryWrapper.eq("type", Constast.***TYPE\_MNEU***);//只能查询菜单  queryWrapper.like(StringUtils.*isNotBlank*(permissionVo.getTitle()), "title", permissionVo.getTitle());  queryWrapper.eq(permissionVo.getId()!=**null**, "id", permissionVo.getId()).or().eq(permissionVo.getId()!=**null**,"pid", permissionVo.getId());  queryWrapper.orderByAsc("ordernum");  **this**.permissionService.page(page, queryWrapper);  **returnnew**DataGridView(page.getTotal(), page.getRecords());  }    /\*\*  \* 加载最大的排序码  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("loadMenuMaxOrderNum")  **public** Map<String,Object>loadMenuMaxOrderNum(){  Map<String, Object>map=**new** HashMap<String, Object>();    QueryWrapper<Permission>queryWrapper=**new**QueryWrapper<>();  queryWrapper.orderByDesc("ordernum");  IPage<Permission>page=**new** Page<>(1, 1);  List<Permission>list = **this**.permissionService.page(page, queryWrapper).getRecords();  **if**(list.size()>0) {  map.put("value", list.get(0).getOrdernum()+1);  }**else** {  map.put("value", 1);  }  **return**map;  }      /\*\*  \* 添加  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("addMenu")  **public**ResultObjaddMenu(PermissionVopermissionVo) {  **try** {  permissionVo.setType(Constast.***TYPE\_MNEU***);//设置添加类型  **this**.permissionService.save(permissionVo);  **return**ResultObj.***ADD\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***ADD\_ERROR***;  }  }    /\*\*  \* 修改  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("updateMenu")  **public**ResultObjupdateMenu(PermissionVopermissionVo) {  **try** {  **this**.permissionService.updateById(permissionVo);  **return**ResultObj.***UPDATE\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***UPDATE\_ERROR***;  }  }      /\*\*  \* 查询当前的ID的菜单有没有子菜单  \*/  @RequestMapping("checkMenuHasChildrenNode")  **public** Map<String,Object>checkMenuHasChildrenNode(PermissionVopermissionVo){  Map<String, Object>map=**new** HashMap<String, Object>();    QueryWrapper<Permission>queryWrapper=**new**QueryWrapper<>();  queryWrapper.eq("pid", permissionVo.getId());  List<Permission>list = **this**.permissionService.list(queryWrapper);  **if**(list.size()>0) {  map.put("value", **true**);  }**else** {  map.put("value", **false**);  }  **return**map;  }    /\*\*  \* 删除  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("deleteMenu")  **public**ResultObjdeleteMenu(PermissionVopermissionVo) {  **try** {  **this**.permissionService.removeById(permissionVo.getId());  **return**ResultObj.***DELETE\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***DELETE\_ERROR***;  }  }    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*菜单管理结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |

## 解决删除的问题

## 解决删除最后一行数据的分页问题



# 权限管理CRUD

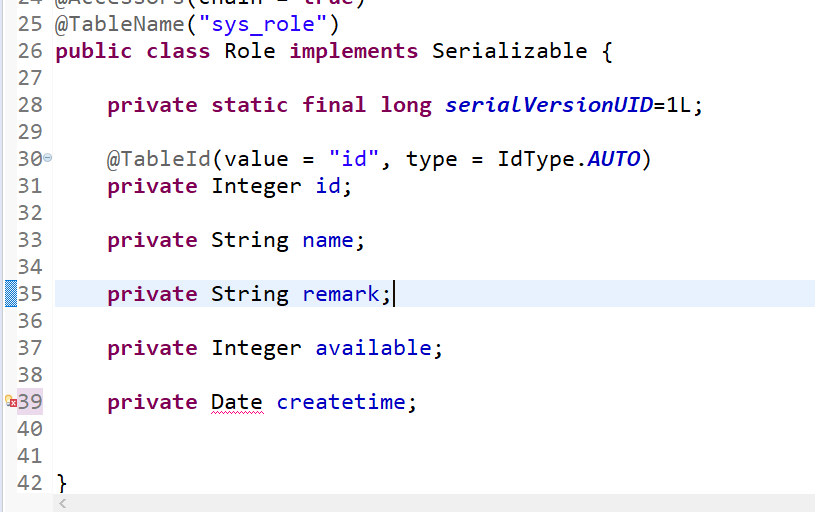
## 修改PermissionController

|  |
| --- |
| /\*\*  \* <p>  \* 前端控制器  \* </p>  \*  \*/  @RestController  @RequestMapping("/permission")  **publicclass**PermissionController {  @Autowired  **private**PermissionServicepermissionService;  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 权限管理开始 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*\*  \* 加载权限管理左边的权限树的json  \*/  @RequestMapping("loadPermissionManagerLeftTreeJson")  **public**DataGridViewloadPermissionManagerLeftTreeJson(PermissionVopermissionVo) {  QueryWrapper<Permission>queryWrapper = **new**QueryWrapper<>();  queryWrapper.eq("type", Constast.***TYPE\_MNEU***);  List<Permission>list = **this**.permissionService.list(queryWrapper);  List<TreeNode>treeNodes = **new**ArrayList<>();  **for** (Permission permission :list) {  Boolean spread = permission.getOpen() == 1 ? **true** : **false**;  treeNodes.add(**new**TreeNode(permission.getId(), permission.getPid(), permission.getTitle(), spread));  }  **returnnew**DataGridView(treeNodes);  }  /\*\*  \* 查询  \*/  @RequestMapping("loadAllPermission")  **public**DataGridViewloadAllPermission(PermissionVopermissionVo) {  IPage<Permission>page = **new** Page<>(permissionVo.getPage(), permissionVo.getLimit());  QueryWrapper<Permission>queryWrapper = **new**QueryWrapper<>();  queryWrapper.eq("type", Constast.***TYPE\_PERMISSION***);// 只能查询权限  queryWrapper.like(StringUtils.*isNotBlank*(permissionVo.getTitle()), "title", permissionVo.getTitle());  queryWrapper.like(StringUtils.*isNotBlank*(permissionVo.getPercode()), "percode", permissionVo.getPercode());  queryWrapper.eq(permissionVo.getId() != **null**, "id", permissionVo.getId()).or().eq(permissionVo.getId() != **null**,  "pid", permissionVo.getId());  queryWrapper.orderByAsc("ordernum");  **this**.permissionService.page(page, queryWrapper);  **returnnew**DataGridView(page.getTotal(), page.getRecords());  }  /\*\*  \* 加载最大的排序码  \*  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("loadPermissionMaxOrderNum")  **public** Map<String, Object>loadPermissionMaxOrderNum() {  Map<String, Object>map = **new** HashMap<String, Object>();  QueryWrapper<Permission>queryWrapper = **new**QueryWrapper<>();  queryWrapper.orderByDesc("ordernum");  IPage<Permission>page = **new** Page<>(1, 1);  List<Permission>list = **this**.permissionService.page(page, queryWrapper).getRecords();  **if** (list.size() > 0) {  map.put("value", list.get(0).getOrdernum() + 1);  } **else** {  map.put("value", 1);  }  **return**map;  }  /\*\*  \* 添加  \*  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("addPermission")  **public**ResultObjaddPermission(PermissionVopermissionVo) {  **try** {  permissionVo.setType(Constast.***TYPE\_PERMISSION***);// 设置添加类型  **this**.permissionService.save(permissionVo);  **return**ResultObj.***ADD\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***ADD\_ERROR***;  }  }  /\*\*  \* 修改  \*  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("updatePermission")  **public**ResultObjupdatePermission(PermissionVopermissionVo) {  **try** {  **this**.permissionService.updateById(permissionVo);  **return**ResultObj.***UPDATE\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***UPDATE\_ERROR***;  }  }  /\*\*  \* 删除  \*  \* **@param**permissionVo  \* **@return**  \*/  @RequestMapping("deletePermission")  **public**ResultObjdeletePermission(PermissionVopermissionVo) {  **try** {  **this**.permissionService.removeById(permissionVo.getId());  **return**ResultObj.***DELETE\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***DELETE\_ERROR***;  }  }  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 权限管理结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  } |

# 角色管理

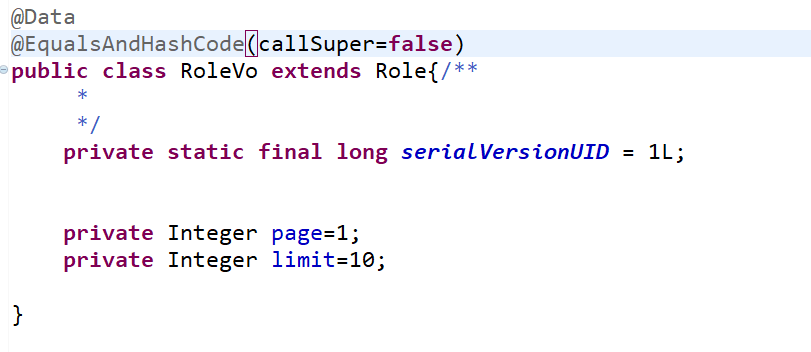
## 角色的CRUD

### 生成Role



### 生成RoleMapper,RoleMapper.xml,RoleService, RoleServiceImpl

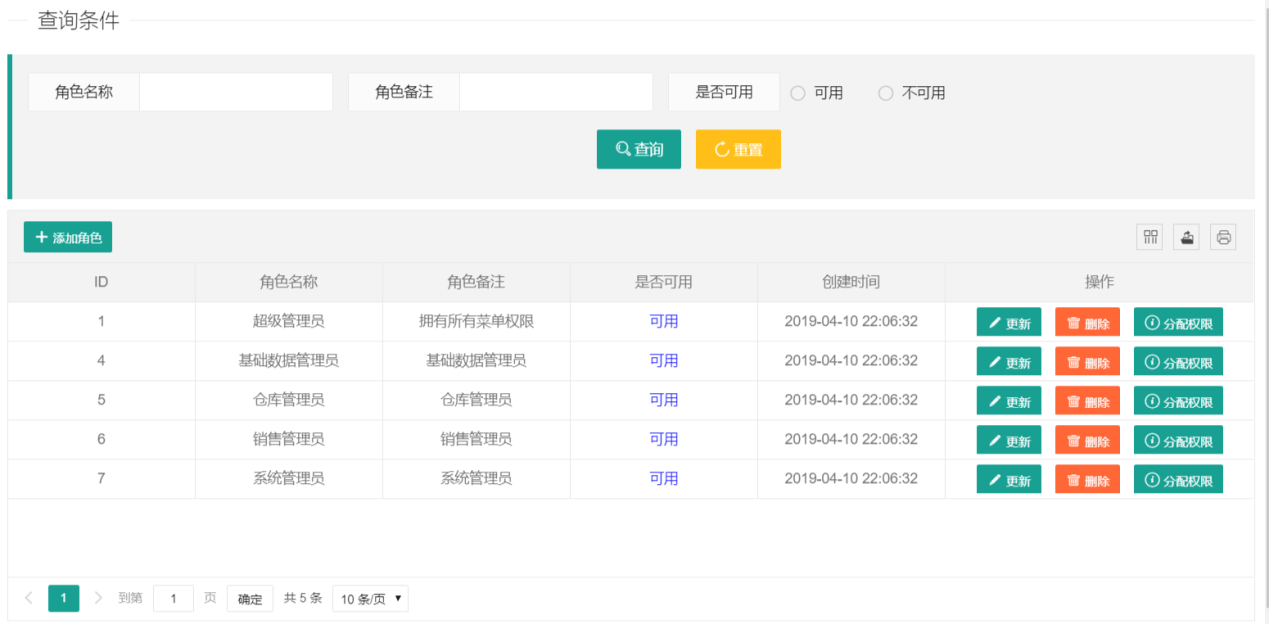
### 创建RoleVo



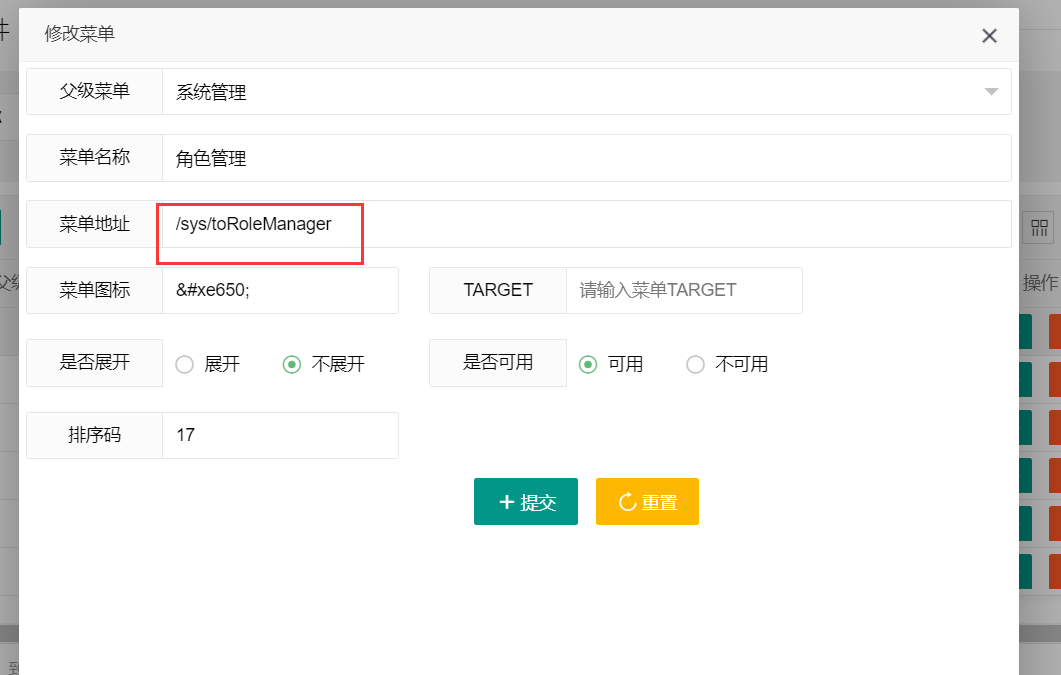
### 生成RoleController

|  |
| --- |
| /\*\*  \* <p>  \* 前端控制器  \* </p>  \*/  @RestController  @RequestMapping("/role")  **publicclass**RoleController {  @Autowired  **private**RoleServiceroleService;      /\*\*  \* 查询  \*/  @RequestMapping("loadAllRole")  **public**DataGridViewloadAllRole(RoleVoroleVo) {  IPage<Role>page=**new** Page<>(roleVo.getPage(), roleVo.getLimit());  QueryWrapper<Role>queryWrapper=**new**QueryWrapper<>();  queryWrapper.like(StringUtils.*isNotBlank*(roleVo.getName()), "name", roleVo.getName());  queryWrapper.like(StringUtils.*isNotBlank*(roleVo.getRemark()), "remark", roleVo.getRemark());  queryWrapper.eq(roleVo.getAvailable()!=**null**, "available", roleVo.getAvailable());  queryWrapper.orderByDesc("createtime");  **this**.roleService.page(page, queryWrapper);  **returnnew**DataGridView(page.getTotal(), page.getRecords());  }      /\*\*  \* 添加  \*/  @RequestMapping("addRole")  **public**ResultObjaddRole(RoleVoroleVo) {  **try** {  roleVo.setCreatetime(**new**Date());  **this**.roleService.save(roleVo);  **return**ResultObj.***ADD\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***ADD\_ERROR***;  }  }  /\*\*  \* 修改  \*/  @RequestMapping("updateRole")  **public**ResultObjupdateRole(RoleVoroleVo) {  **try** {  **this**.roleService.updateById(roleVo);  **return**ResultObj.***UPDATE\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***UPDATE\_ERROR***;  }  }    /\*\*  \* 删除  \*/  @RequestMapping("deleteRole")  **public**ResultObjdeleteRole(Integer id) {  **try** {  **this**.roleService.removeById(id);  **return**ResultObj.***DELETE\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***DELETE\_ERROR***;  }  }  } |

### 创建role/roleManager.html



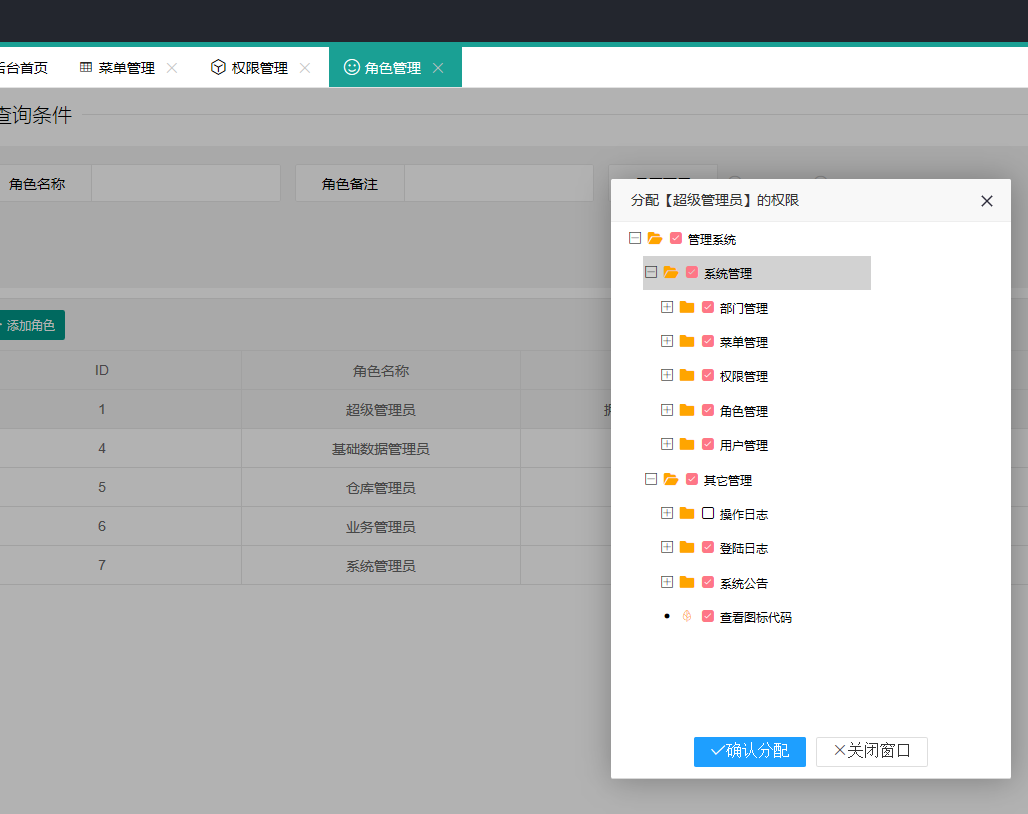
### 修改菜单地址



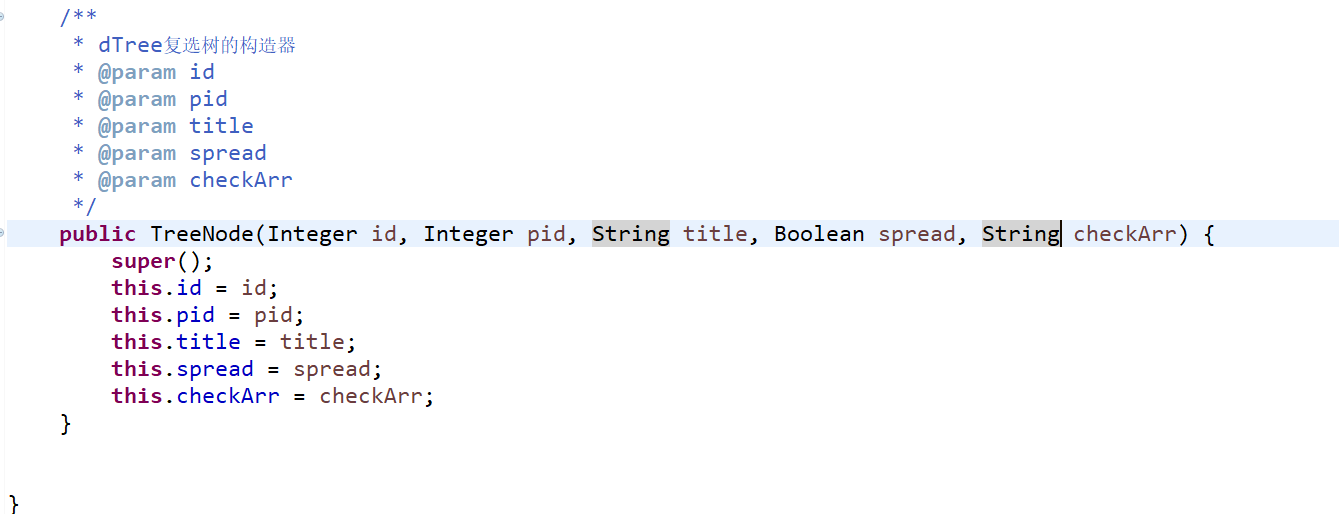
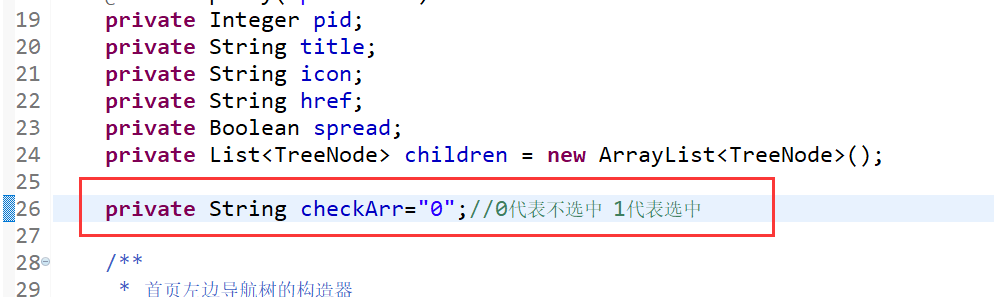
## 角色分配权限

### 修改role/roleManager.html





### 修改TreeNode



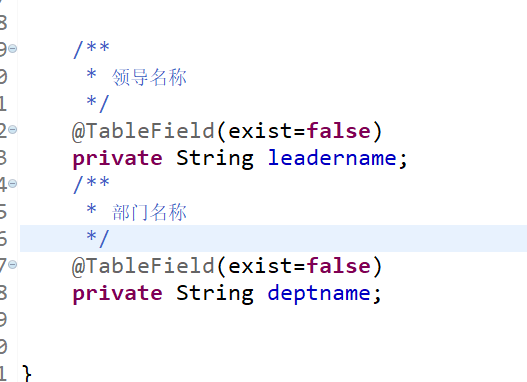
### 修改RoleController

|  |
| --- |
| /\*\*  \* 根据角色ID加载菜单和权限的树的json串  \*/  @RequestMapping("initPermissionByRoleId")  **public**DataGridViewinitPermissionByRoleId(Integer roleId) {  //查询所有可用的菜单和权限  QueryWrapper<Permission>queryWrapper=**new**QueryWrapper<>();  queryWrapper.eq("available", Constast.***AVAILABLE\_TRUE***);  List<Permission>allPermissions = permissionService.list(queryWrapper);  /\*\*  \* 1,根据角色ID查询当前角色拥有的所有的权限或菜单ID  \* 2,根据查询出来的菜单ID查询权限和菜单数据  \*/  List<Integer>currentRolePermissions=**this**.roleService.queryRolePermissionIdsByRid(roleId);  List<Permission>carrentPermissions=**null**;  **if**(currentRolePermissions.size()>0) { //如果有ID就去查询  queryWrapper.in("id", currentRolePermissions);  carrentPermissions = permissionService.list(queryWrapper);  }**else** {  carrentPermissions=**new**ArrayList<>();  }  //构造 List<TreeNode>  List<TreeNode>nodes=**new**ArrayList<>();  **for** (Permission p1 :allPermissions) {  String checkArr="0";  **for** (Permission p2 :carrentPermissions) {  **if**(p1.getId()==p2.getId()) {  checkArr="1";  **break**;  }  }  Boolean spread=(p1.getOpen()==**null**||p1.getOpen()==1)?**true**:**false**;  nodes.add(**new**TreeNode(p1.getId(), p1.getPid(), p1.getTitle(), spread, checkArr));  }  **returnnew**DataGridView(nodes);  }    /\*\*  \* 保存角色和菜单权限之间的关系  \*/  @RequestMapping("saveRolePermission")  **public**ResultObjsaveRolePermission(Integer rid,Integer[] ids) {  **try** {  **this**.roleService.saveRolePermission(rid,ids);  **return**ResultObj.***DISPATCH\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***DISPATCH\_ERROR***;  }  } |

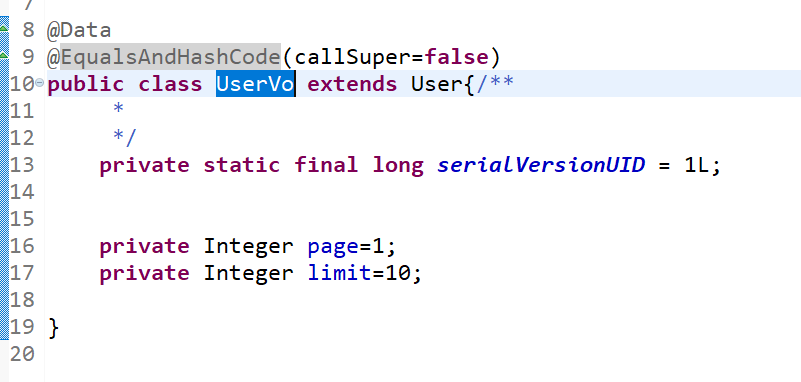
# 用户管理

## 查询

### 修改User

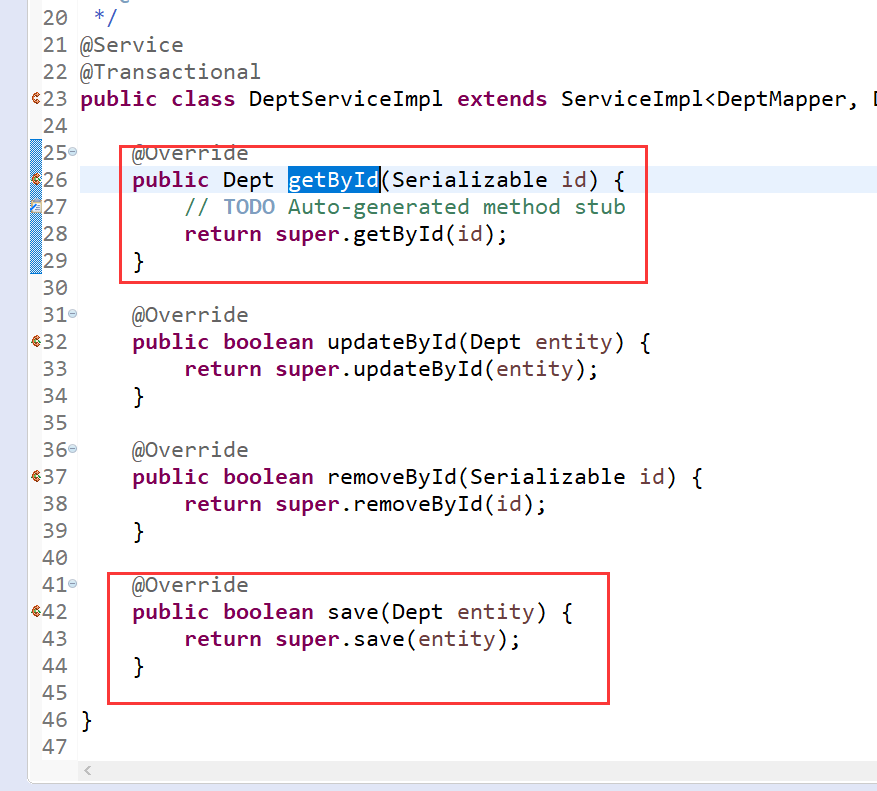


### 创建UserVo



### 修改UserServiceImpl

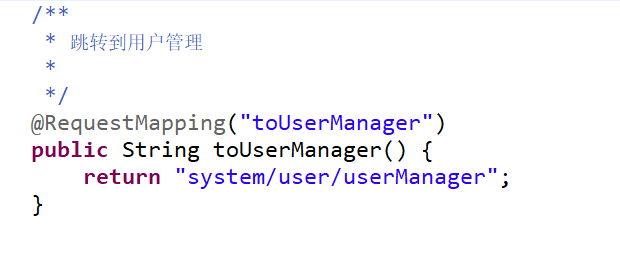
### 修改DeptServiceImpl



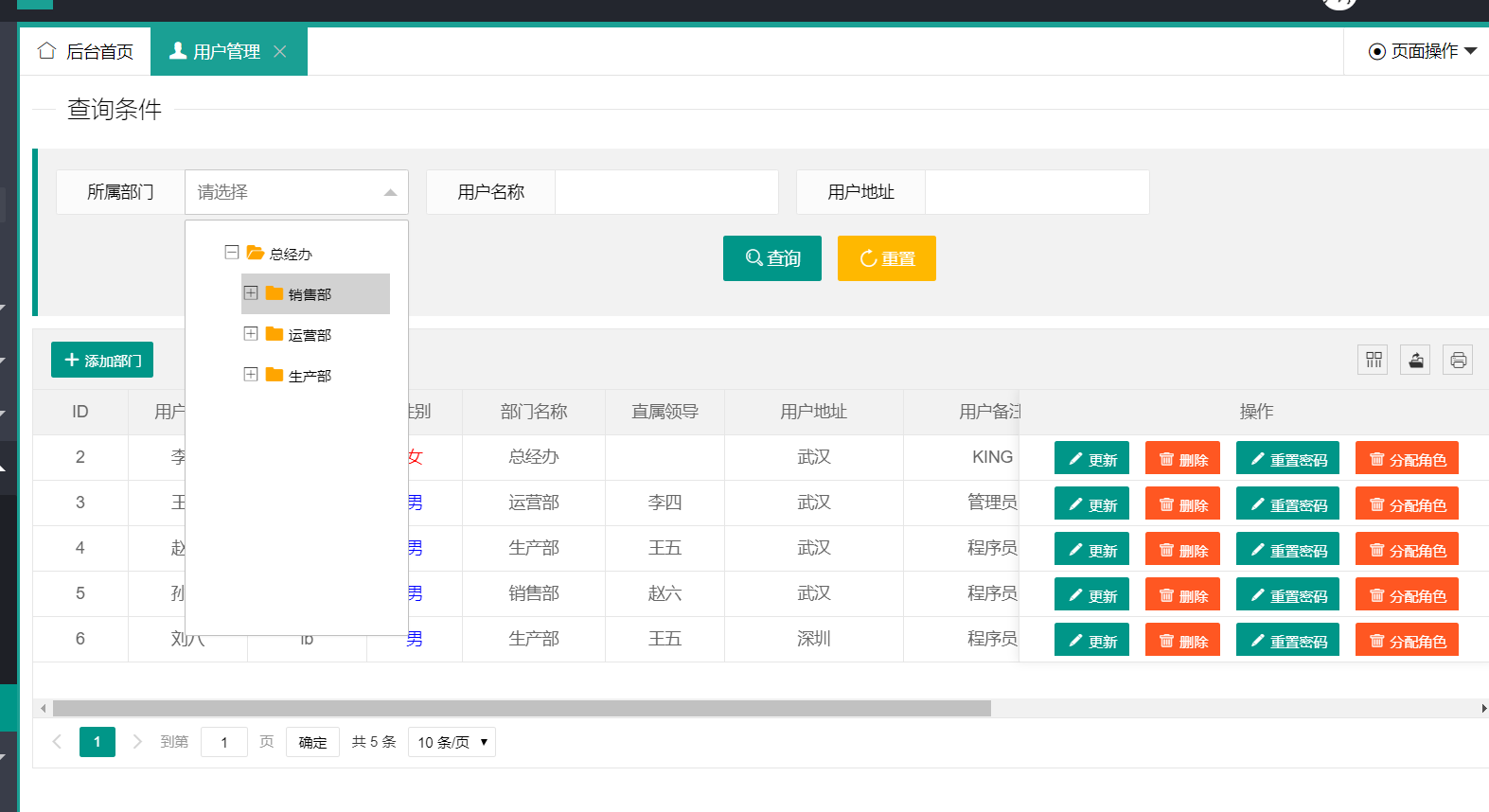
### 创建UserController

|  |
| --- |
| /\*\*  \* <p>  \* 前端控制器  \* </p>  @RestController  @RequestMapping("/user")  **publicclass**UserController {    @Autowired  **private**UserServiceuserService;    @Autowired  **private**DeptServicedeptService;    /\*\*  \* 用户全查询  \*/  @RequestMapping("loadAllUser")  **public**DataGridViewloadAllUser(UserVouserVo) {  IPage<User>page=**new** Page<>(userVo.getPage(), userVo.getLimit());  QueryWrapper<User>queryWrapper=**new**QueryWrapper<>();  queryWrapper.eq(StringUtils.*isNotBlank*(userVo.getName()), "loginname", userVo.getName()).or().eq(StringUtils.*isNotBlank*(userVo.getName()), "name", userVo.getName());  queryWrapper.eq(StringUtils.*isNotBlank*(userVo.getAddress()), "address", userVo.getAddress());  queryWrapper.eq("type", Constast.***USER\_TYPE\_NORMAL***);//查询系统用户  queryWrapper.eq(userVo.getDeptid()!=**null**, "deptid",userVo.getDeptid());  **this**.userService.page(page, queryWrapper);    List<User>list = page.getRecords();  **for** (User user :list) {  Integer deptid = user.getDeptid();  **if**(deptid!=**null**) {  Dept one =deptService.getById(deptid);  user.setDeptname(one.getTitle());  }  Integer mgr = user.getMgr();  **if**(mgr!=**null**) {  User one = **this**.userService.getById(mgr);  user.setLeadername(one.getName());  }  }    **returnnew**DataGridView(page.getTotal(), list);    } |

### 创建SystemController



### 创建user/userManager.html



## 添加用户

### 修改userControlller

|  |
| --- |
| /\*\*  \* 加载最大的排序码  \* **@param**deptVo  \* **@return**  \*/  @RequestMapping("loadUserMaxOrderNum")  **public** Map<String,Object>loadUserMaxOrderNum(){  Map<String, Object>map=**new** HashMap<String, Object>();    QueryWrapper<User>queryWrapper=**new**QueryWrapper<>();  queryWrapper.orderByDesc("ordernum");  IPage<User>page=**new** Page<>(1, 1);  List<User>list = **this**.userService.page(page, queryWrapper).getRecords();  **if**(list.size()>0) {  map.put("value", list.get(0).getOrdernum()+1);  }**else** {  map.put("value", 1);  }  **return**map;  }      /\*\*  \* 根据部门ID查询用户  \*/  @RequestMapping("loadUsersByDeptId")  **public**DataGridViewloadUsersByDeptId(Integer deptid) {  QueryWrapper<User>queryWrapper=**new**QueryWrapper<>();  queryWrapper.eq(deptid!=**null**, "deptid", deptid);  queryWrapper.eq("available", Constast.***AVAILABLE\_TRUE***);  queryWrapper.eq("type", Constast.***USER\_TYPE\_NORMAL***);  List<User>list = **this**.userService.list(queryWrapper);  **returnnew**DataGridView(list);  }      /\*\*  \* 把用户名转成拼音  \*/  @RequestMapping("changeChineseToPinyin")  **public** Map<String,Object>changeChineseToPinyin(String username){  Map<String,Object>map=**new** HashMap<>();  **if**(**null**!=username) {  map.put("value", PinyinUtils.*getPingYin*(username));  }**else** {  map.put("value", "");  }  **return**map;  }    /\*\*  \* 添加用户  \*/  @RequestMapping("addUser")  **public**ResultObjaddUser(UserVouserVo) {  **try** {  userVo.setType(Constast.***USER\_TYPE\_NORMAL***);//设置类型  userVo.setHiredate(**new**Date());  String salt=IdUtil.*simpleUUID*().toUpperCase();  userVo.setSalt(salt);//设置盐  userVo.setPwd(**new** Md5Hash(Constast.***USER\_DEFAULT\_PWD***, salt, 2).toString());//设置密码  **this**.userService.save(userVo);  **return**ResultObj.***ADD\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***ADD\_ERROR***;  }  }  } |

### 修改pom.xml

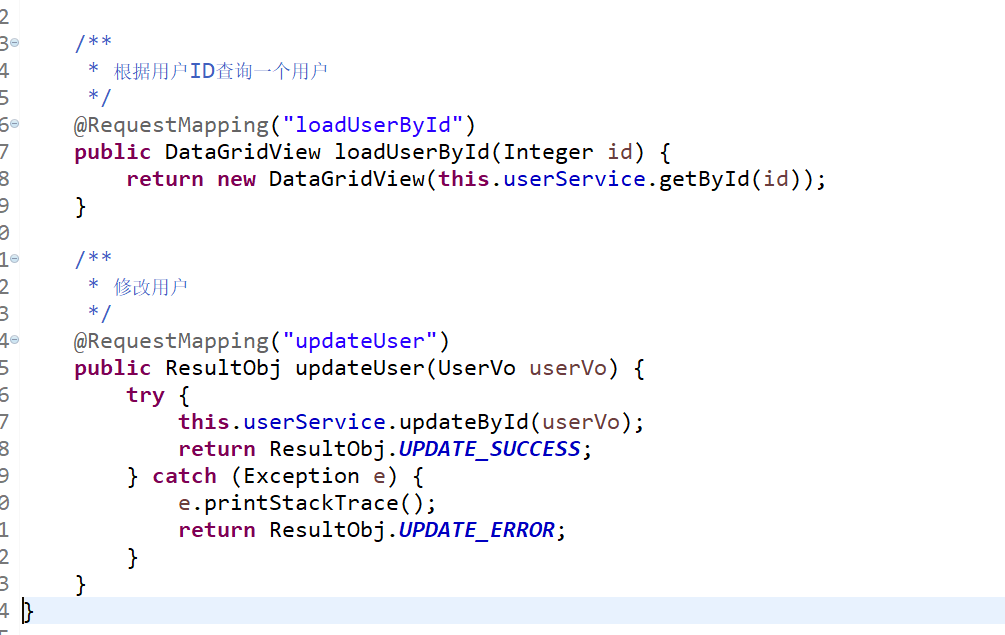
<pinyin4j.version>2.5.1</pinyin4j.version>

|  |
| --- |
| <!-- pinyin4j -->  <dependency>  <groupId>com.belerweb</groupId>  <artifactId>pinyin4j</artifactId>  <version>${pinyin4j.version}</version>  </dependency> |

### 创建PinyinUtils

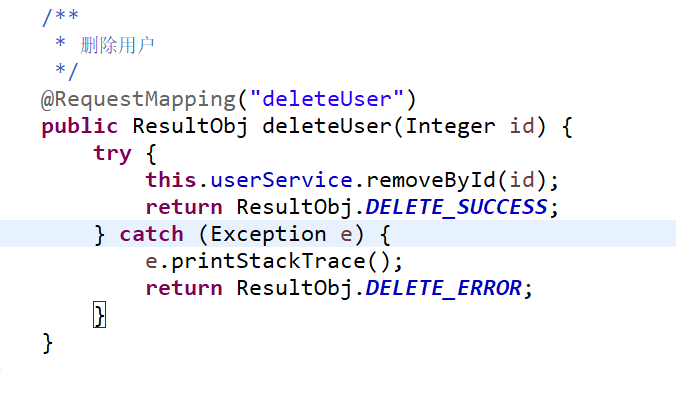
## 修改用户

### 修改userController

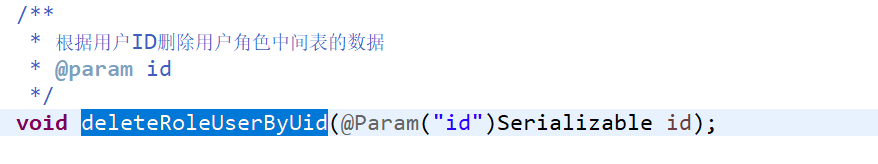


## 删除用户

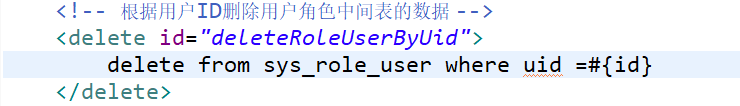
### 修改userController



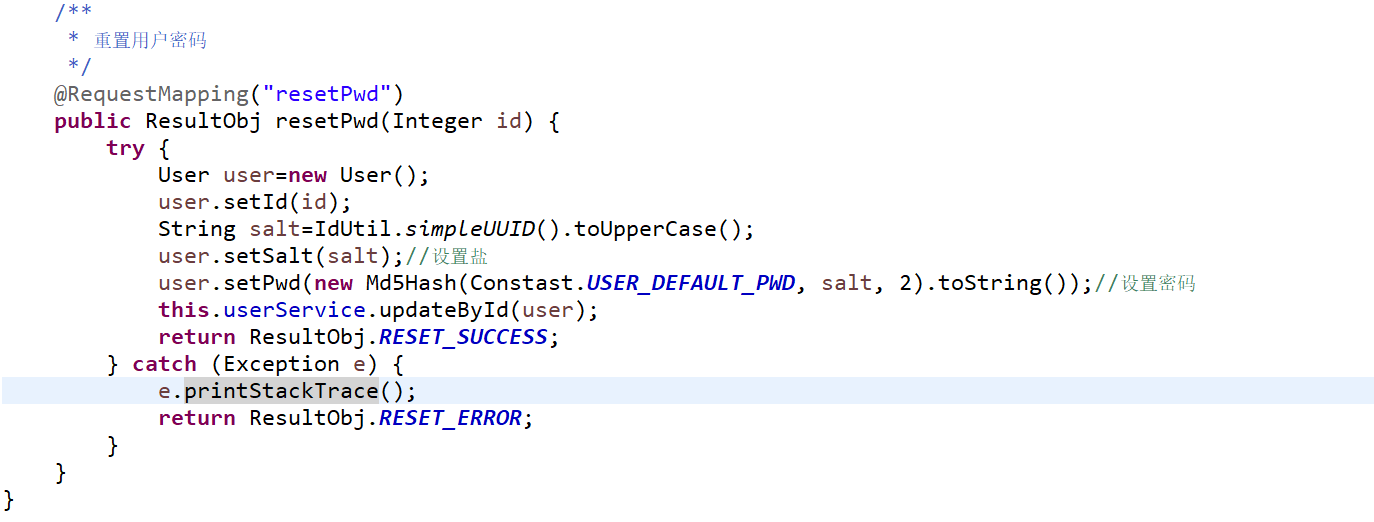
### 修改RoleMapper



### 修改RoleMapper.xml



### 修改UserController

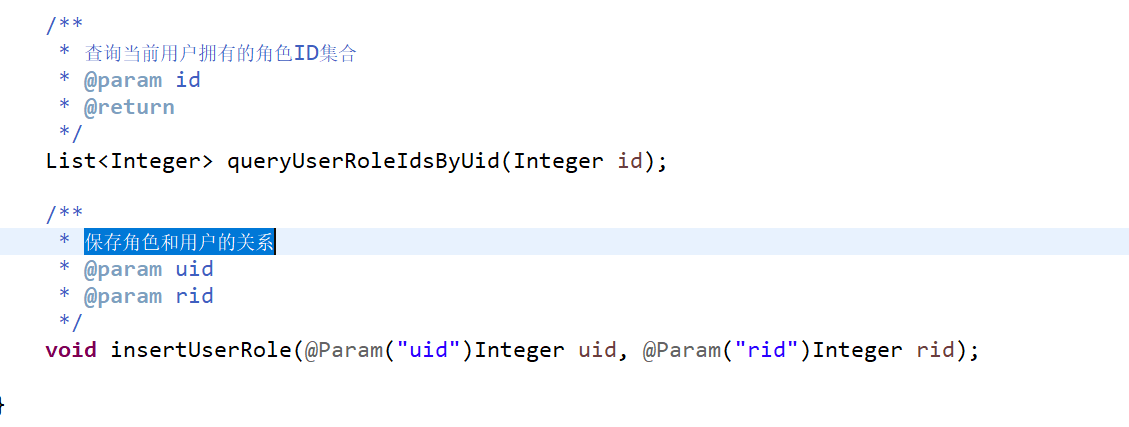


## 分配角色

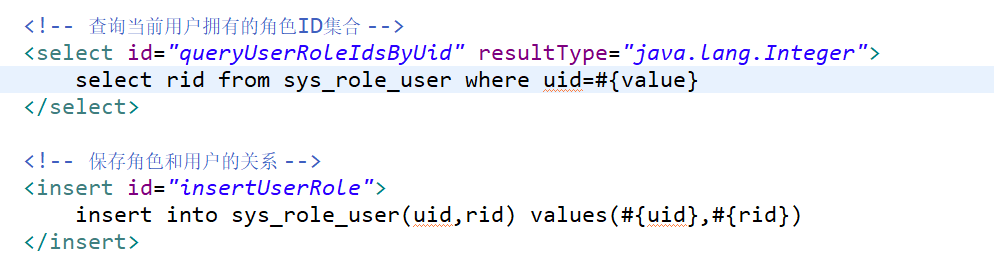
### 修改userController

|  |
| --- |
| /\*\*  \* 根据用户ID查询角色并选中已拥有的角色  \*/  @RequestMapping("initRoleByUserId")  **public**DataGridViewinitRoleByUserId(Integer id) {  //1,查询所有可用的角色  QueryWrapper<Role>queryWrapper=**new**QueryWrapper<>();  queryWrapper.eq("available", Constast.***AVAILABLE\_TRUE***);  List<Map<String, Object>>listMaps = **this**.roleService.listMaps(queryWrapper);    //2,查询当前用户拥有的角色ID集合  List<Integer>currentUserRoleIds=**this**.roleService.queryUserRoleIdsByUid(id);  **for** (Map<String, Object>map :listMaps) {  Boolean LAY\_CHECKED=**false**;  Integer roleId=(Integer) map.get("id");  **for** (Integer rid :currentUserRoleIds) {  **if**(rid==roleId) {  LAY\_CHECKED=**true**;  **break**;  }  }  map.put("LAY\_CHECKED", LAY\_CHECKED);  }  **returnnew**DataGridView(Long.*valueOf*(listMaps.size()), listMaps);  }    /\*\*  \* 保存用户和角色的关系  \*/  @RequestMapping("saveUserRole")  **public**ResultObjsaveUserRole(Integer uid,Integer[] ids) {  **try** {  **this**.userService.saveUserRole(uid,ids);  **return**ResultObj.***DISPATCH\_SUCCESS***;  } **catch** (Exception e) {  e.printStackTrace();  **return**ResultObj.***DISPATCH\_ERROR***;  }    } |

### 修改RoleMapper



### 修改RoleMapper.xml



# 登陆关联菜单和权限

## 关联菜单



## 关联权限

### 修改UserRealm

|  |
| --- |
| **publicclass**UserRealm**extends**AuthorizingRealm {  @Autowired  @Lazy//只有使用的时候才会加载  **private**UserServiceuserService;    @Autowired  @Lazy  **private**PermissionServicepermissionService;    @Autowired  @Lazy  **private**RoleServiceroleService;  @Override  **public** String getName() {  **returnthis**.getClass().getSimpleName();  }  /\*\*  \* 认证  \*/  @Override  **protected**AuthenticationInfodoGetAuthenticationInfo(AuthenticationTokentoken) **throws**AuthenticationException {  QueryWrapper<User>queryWrapper = **new**QueryWrapper<>();  queryWrapper.eq("loginname", token.getPrincipal().toString());  User user = userService.getOne(queryWrapper);  **if** (**null** != user) {  ActiverUseractiverUser = **new**ActiverUser();  activerUser.setUser(user);    //根据用户ID查询percode  //查询所有菜单  QueryWrapper<Permission>qw=**new**QueryWrapper<>();  //设置只能查询菜单  qw.eq("type",Constast.***TYPE\_PERMISSION***);  qw.eq("available", Constast.***AVAILABLE\_TRUE***);    //根据用户ID+角色+权限去查询  Integer userId=user.getId();  //根据用户ID查询角色  List<Integer>currentUserRoleIds = roleService.queryUserRoleIdsByUid(userId);  //根据角色ID取到权限和菜单ID  Set<Integer>pids=**new** HashSet<>();  **for** (Integer rid :currentUserRoleIds) {  List<Integer>permissionIds = roleService.queryRolePermissionIdsByRid(rid);  pids.addAll(permissionIds);  }  List<Permission>list=**new**ArrayList<>();  //根据角色ID查询权限  **if**(pids.size()>0) {  qw.in("id", pids);  list=permissionService.list(qw);  }  List<String>percodes=**new**ArrayList<>();  **for** (Permission permission :list) {  percodes.add(permission.getPercode());  }  //放到  activerUser.setPermissions(percodes);    ByteSourcecredentialsSalt = ByteSource.Util.*bytes*(user.getSalt());  SimpleAuthenticationInfoinfo = **new**SimpleAuthenticationInfo(activerUser, user.getPwd(), credentialsSalt,  **this**.getName());  **return**info;  }  **returnnull**;  }  /\*\*  \* 授权  \*/  @Override  **protected**AuthorizationInfodoGetAuthorizationInfo(PrincipalCollectionprincipals) {  SimpleAuthorizationInfoauthorizationInfo=**new**SimpleAuthorizationInfo();  ActiverUseractiverUser=(ActiverUser) principals.getPrimaryPrincipal();  User user=activerUser.getUser();  List<String>permissions = activerUser.getPermissions();  **if**(user.getType()==Constast.***USER\_TYPE\_SUPER***) {  authorizationInfo.addStringPermission("\*:\*");  }**else** {  **if**(**null**!=permissions&&permissions.size()>0) {  authorizationInfo.addStringPermissions(permissions);  }  }  **return**authorizationInfo;  }  } |