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

## 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**

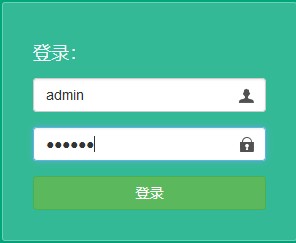
功能权限定义：为可见、可以操作的功能范围。在目录、菜单上加权限控制，有权限的就可以访问对应模块，没有的连菜单名都看不到。在业务模块的功能按钮上加权限控制，最小粒度的控制用户行为。

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（原型设计）

1. 登录功能
   1. 登录页面

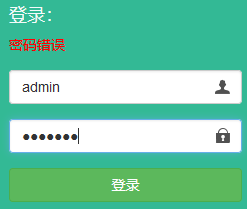


* 1. 异常登录提示错误信息

用户名错误



密码错误



1.3主页面（index页面）



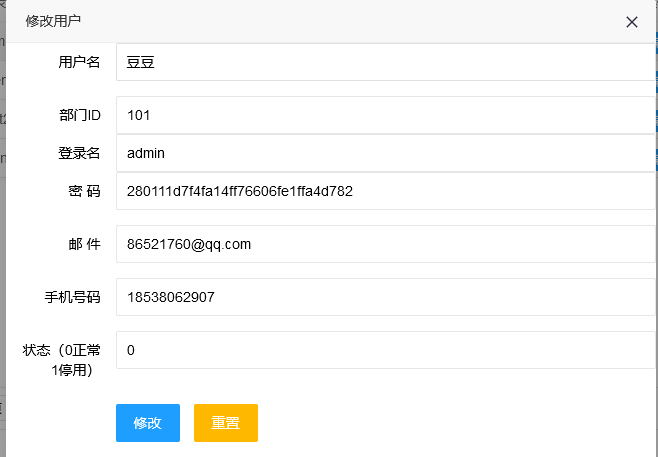
1. 用户管理
   1. 查询用户



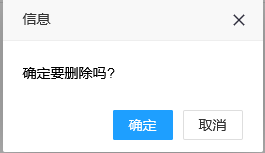
* 1. 添加用户



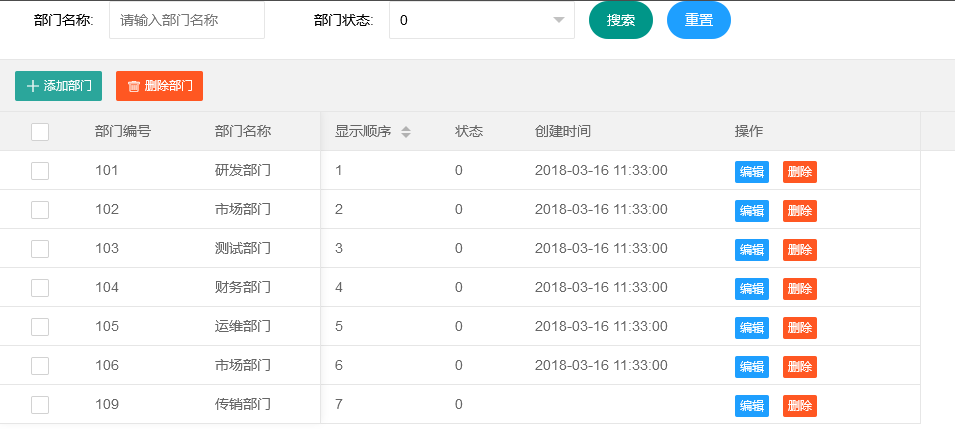
* 1. 修改用户，禁用用户



* 1. 删除用户



1. 部门管理
   1. 查询部门



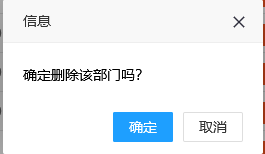
* 1. 添加部门



* 1. 修改部门，禁用部门



* 1. 删除部门



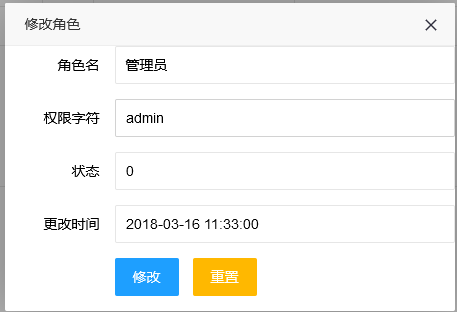
1. 角色管理
   1. 查询角色



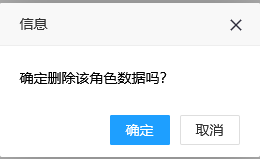
* 1. 添加角色



* 1. 修改角色



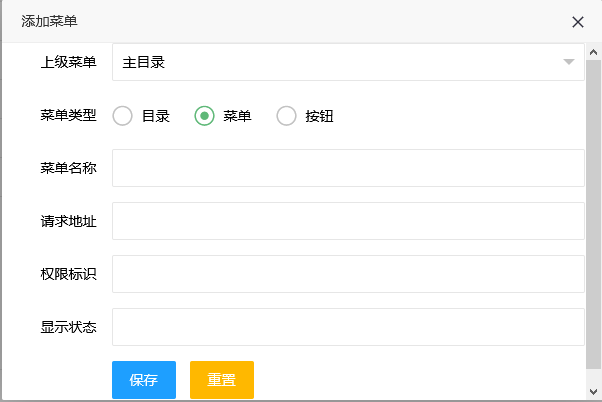
* 1. 删除角色



1. 菜单管理
   1. 查询菜单



* 1. 添加菜单

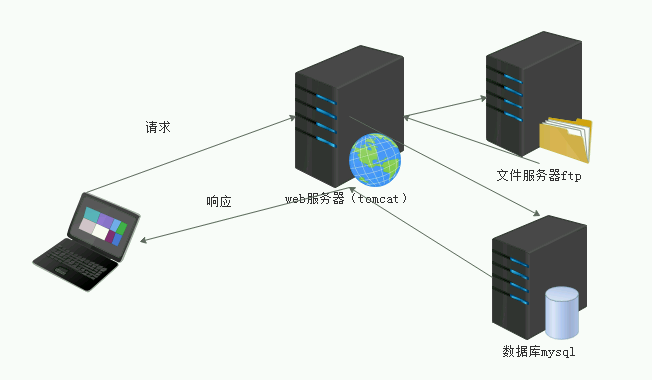


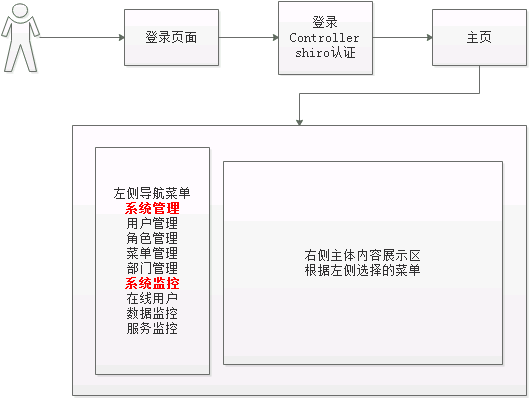
* 1. 修改菜单



## 4、架构设计

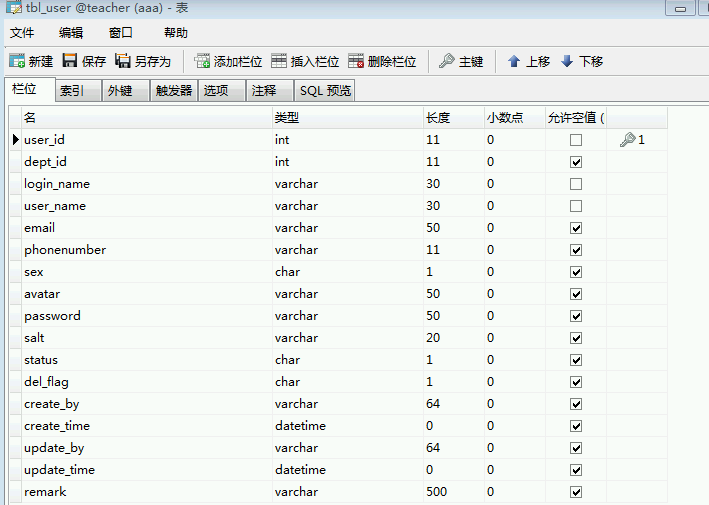
### 4.1、整体架构设计



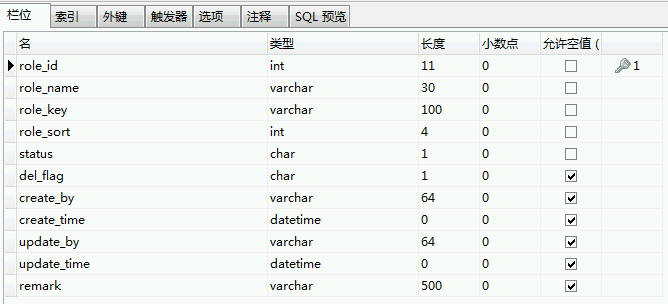


### 4.2、数据模型设计

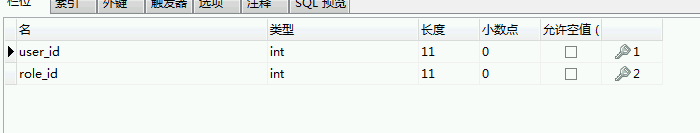
#### 4.2.1、用户表



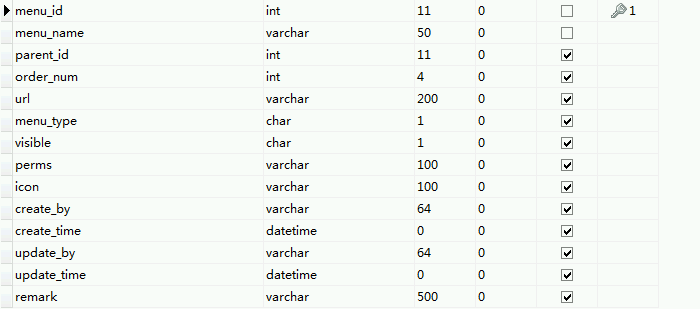
#### 4.2.2、角色表



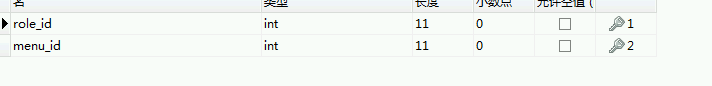
#### 4.2.3、用户角色表（中间表）



#### 4.2.4、菜单表



#### 4.2.5、角色菜单表



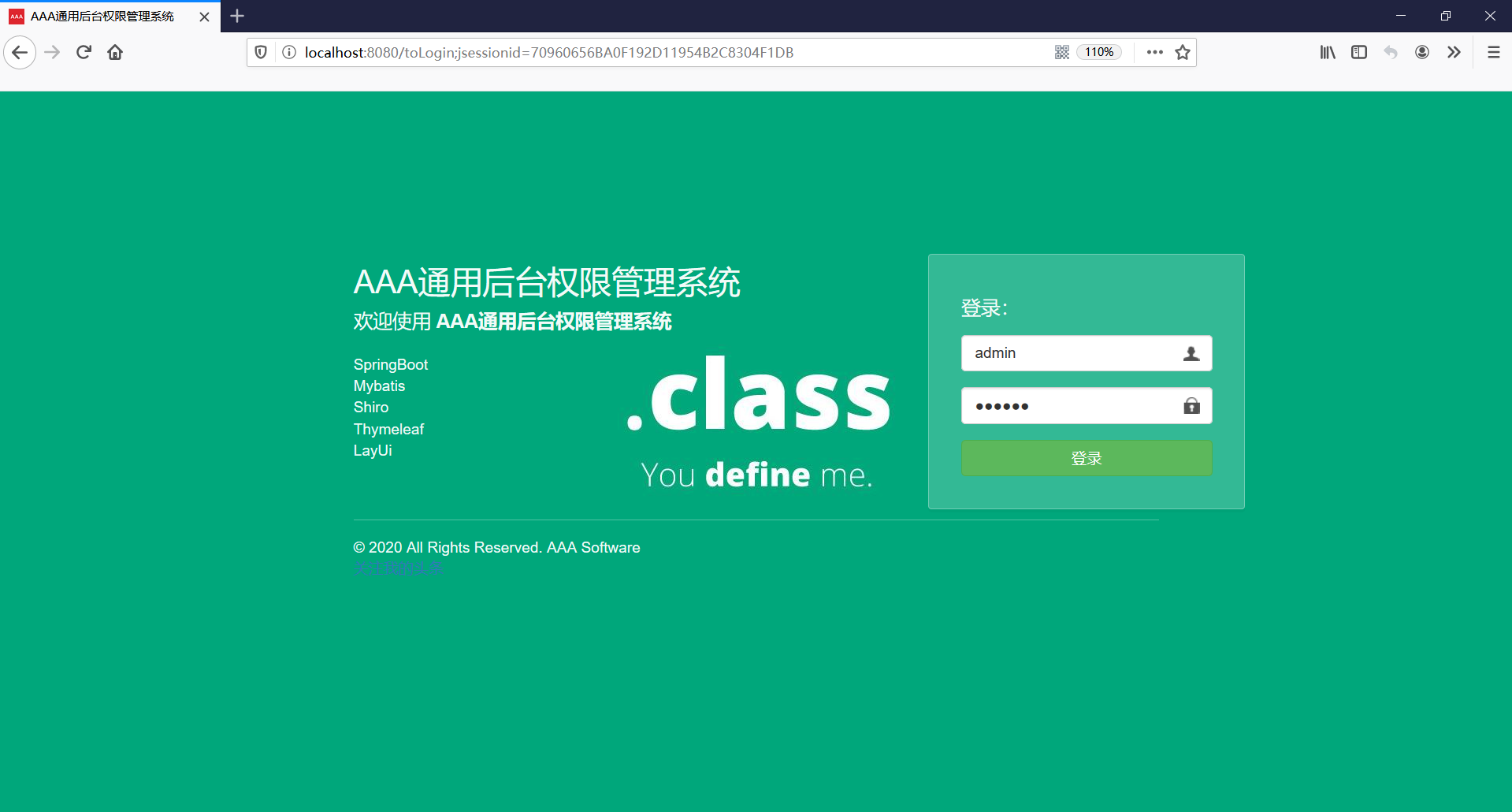
#### 部门表



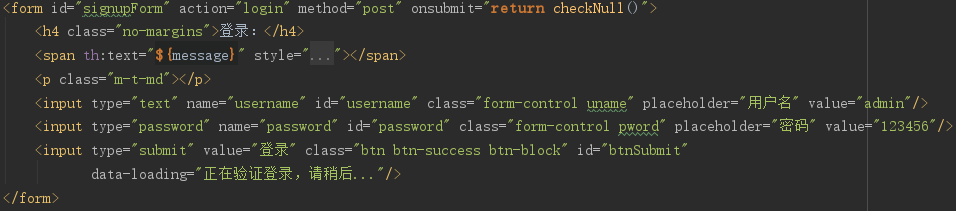
## 5、代码开发

1. 登录功能

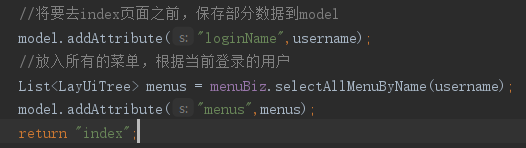
登录页面：



该页面为通用后台权限管理系统的登录页面，由主页login.html文件存储前端界面，主要功能是实现登录进入后台系统，首先要进入该页面可在地址栏输入本地地址：localhost：8080（本地mysql端口）即可自动跳转，因为后台实例设置了所有本地地址都要先通过login页面登陆才能正确到达。这里主要说明登陆框中的主要代码：

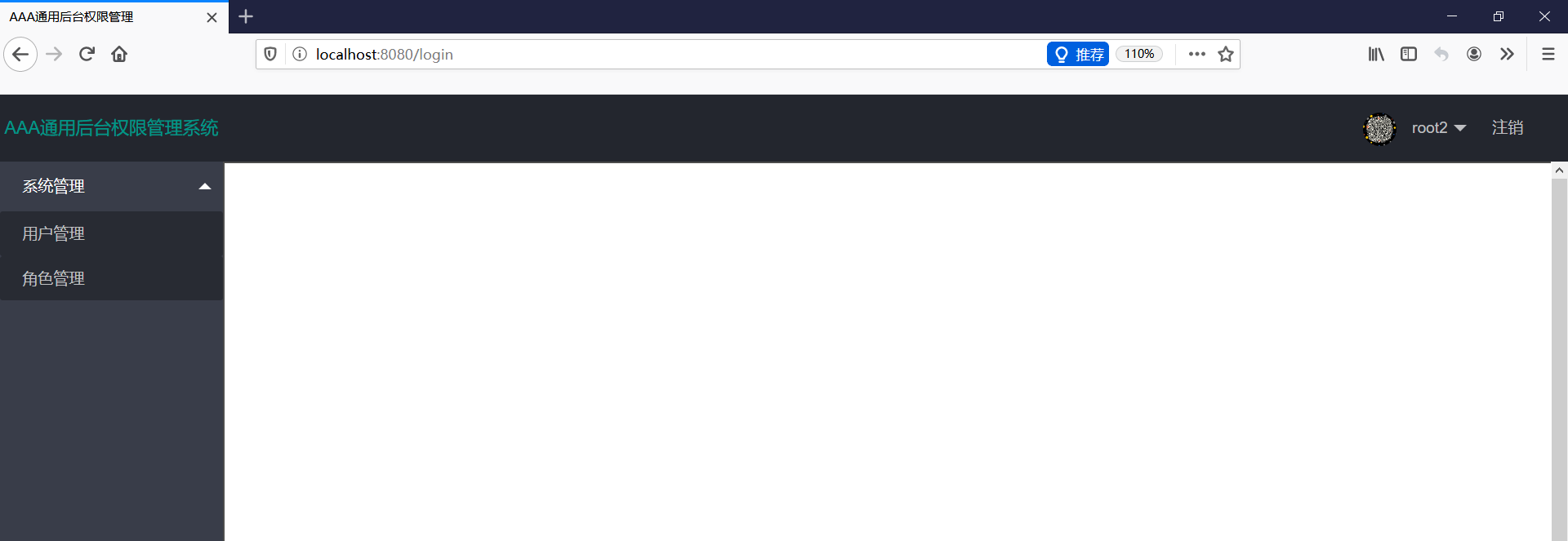


由from表单提交登陆数据，通过id=“signupForm”来传递数据，在LoginController实体类中通过login类来验证登陆名是否正确，在LoginController实体类中,下面代码实现了通过所登陆的用户名来显示对应的管理菜单到index.html中：



例如，我们所登陆的用户名是admin，那么在数据库中我们设计的admin的role\_id=1，那么我们对应返回的菜单则是role\_id对应的menu\_id所对应的菜单；当我们登陆的账号为root的时候，那返回的菜单则是root2数据库中对应的role\_id所对应的menu\_id，在数据库中我们由tbl\_role\_menu来实现菜单对应用户功能。如下图所示



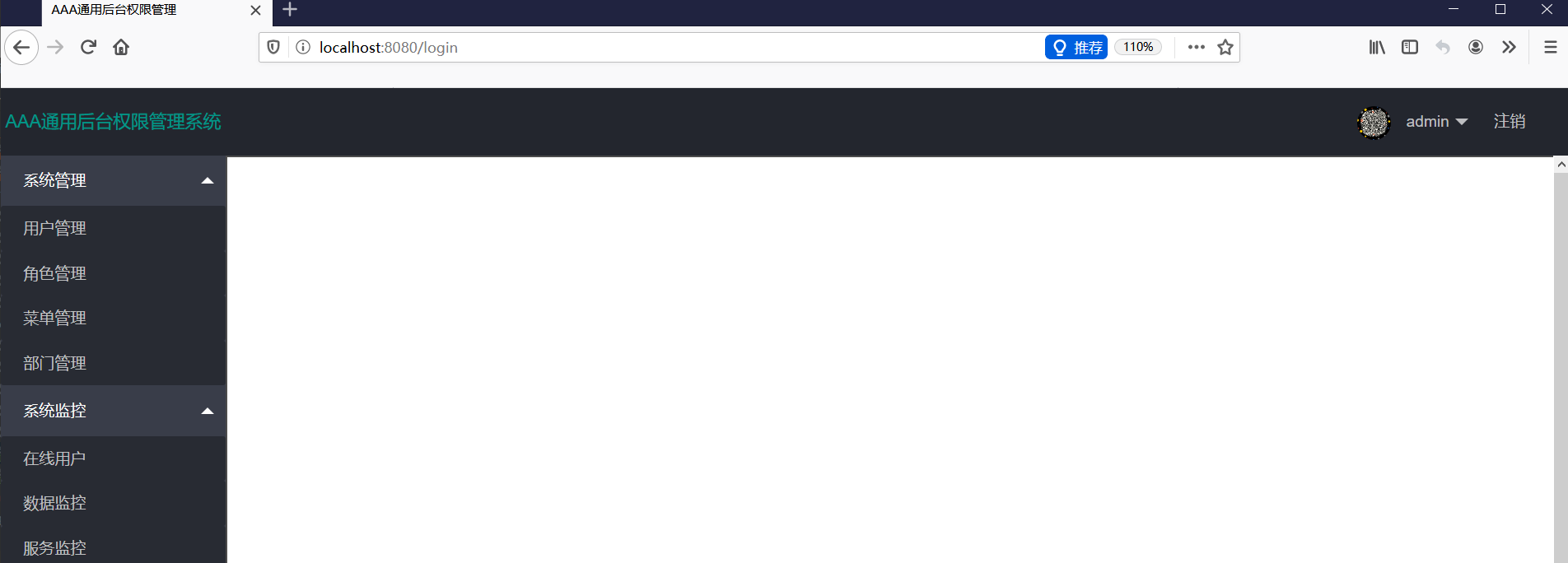


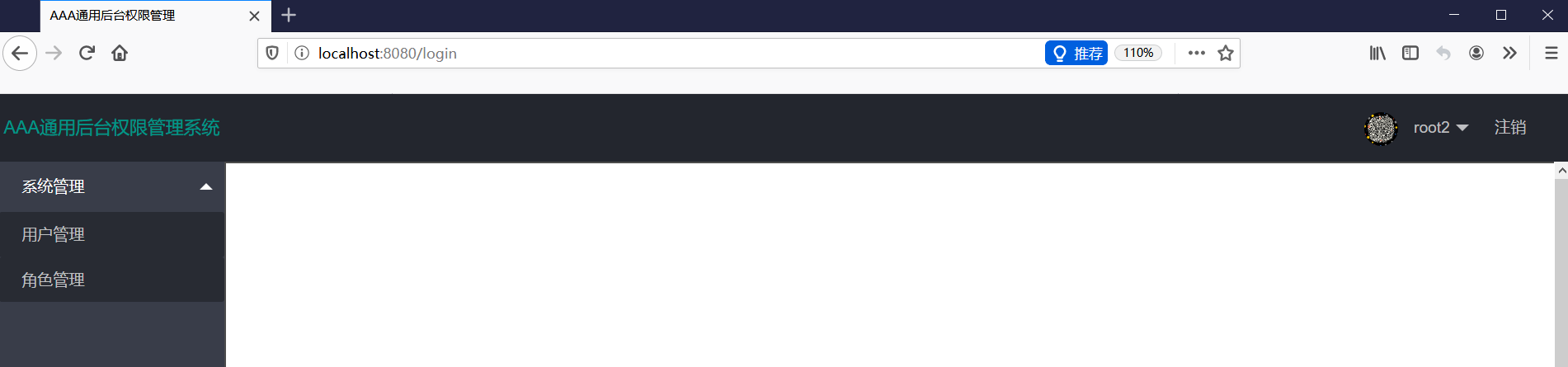
管理主页：

当成功登陆上后台管理系统之后，则会显示后台管理系统文件，主要页面由index.html来实现，管理主页由左侧菜单右侧显示菜单对应信息



这部分主要代码，由${menus}获取菜单，由${menu.title}获取菜单名并以超链接的形式返回。由${menu.children}获取子菜单，由${child.url}和${child.title}获取子菜单连接和名字。下面的<iframe></iframe>为数据显示的区域实现标签。管理主页页面如下：





1. 用户管理

首先，你点击用户管理会显示出

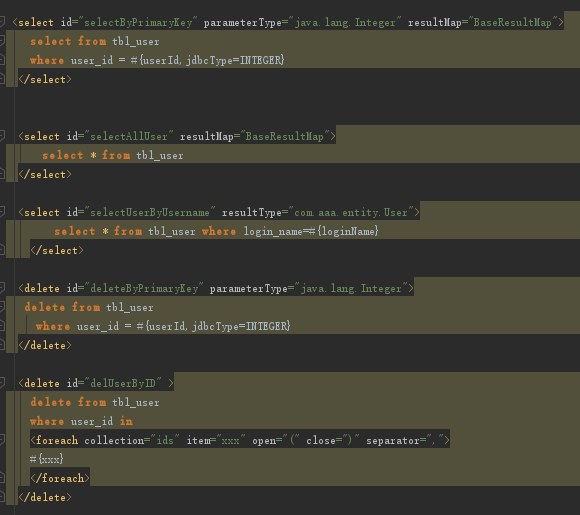
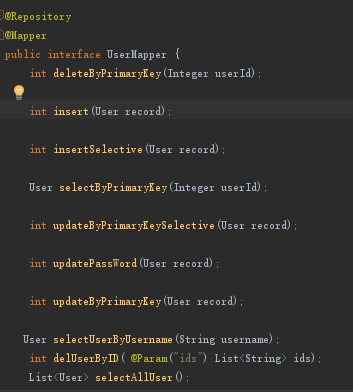


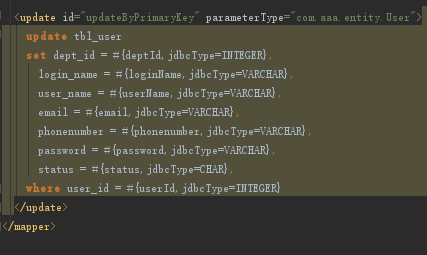
代码实现：

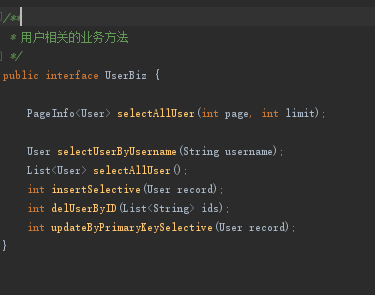
使用layUI的表格重载访问showUser数据接口获得查询后的用户信息。

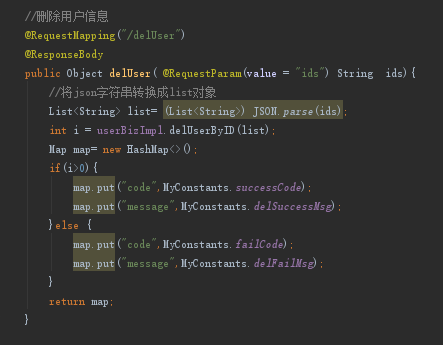
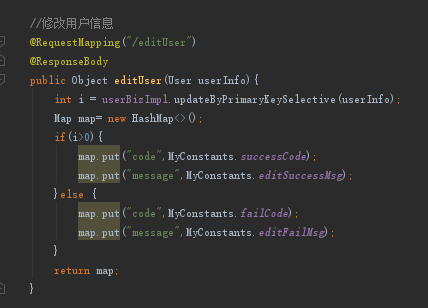


在后端实现增删改查分别在dao的UserMapper和Usermapper.xml、Userbiz、Userbizimpl的接口和实现类中实现相关方法，同时在controller中提供相关的访问接口来调用增删改查的方法，原理使用的都是老师上课讲的来实现的，html页面代码参考老师视频和layUI官网写的





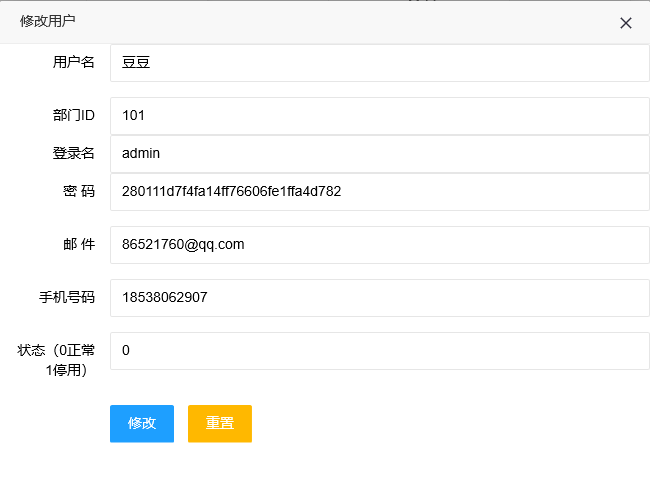
****

****

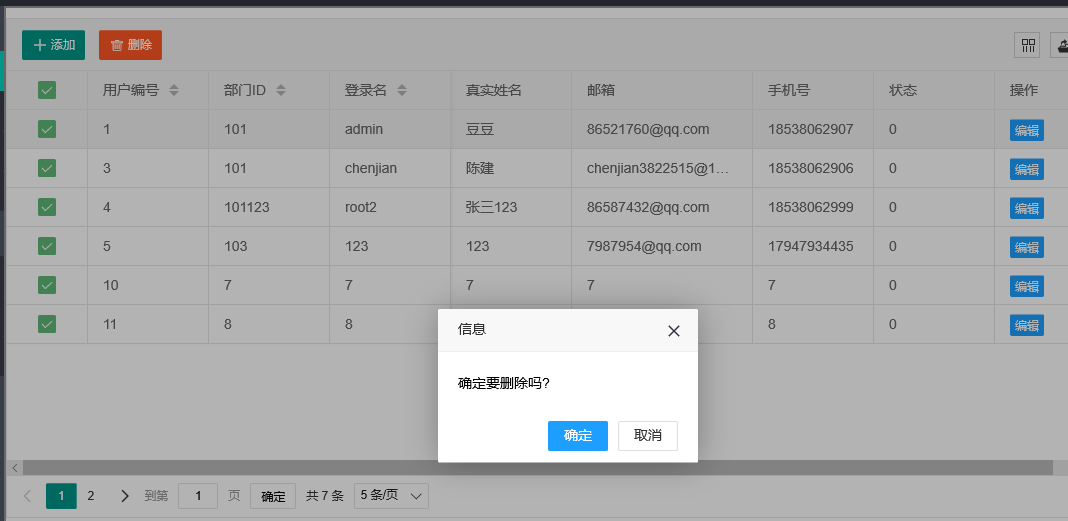
添加用户：

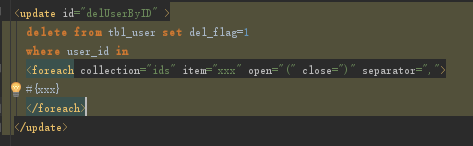


编辑用户：点击编辑出现原有值，直接在其上面修改即可



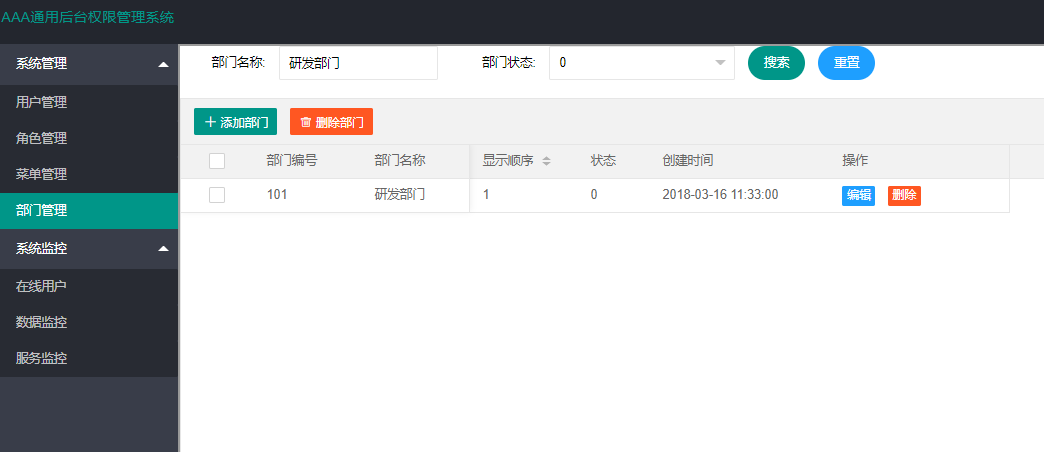
删除用户：方法是通过修改状态的： 0 和 1 ，0就代表正常，1代表停用





1. 部门管理**：**

根据部门名称和状态查询部门信息：

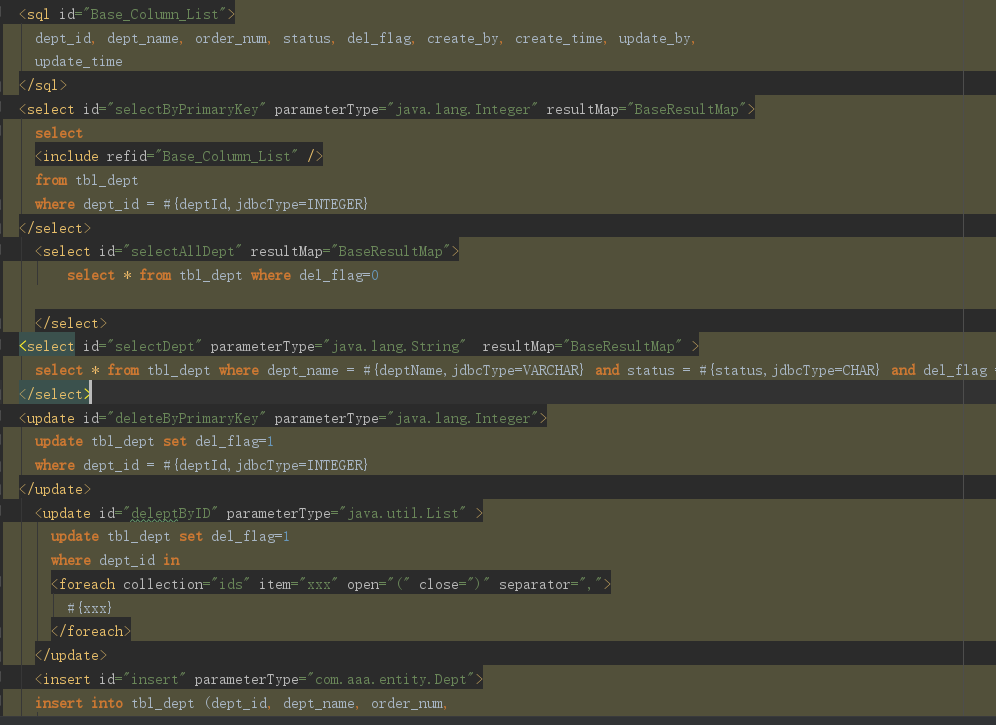


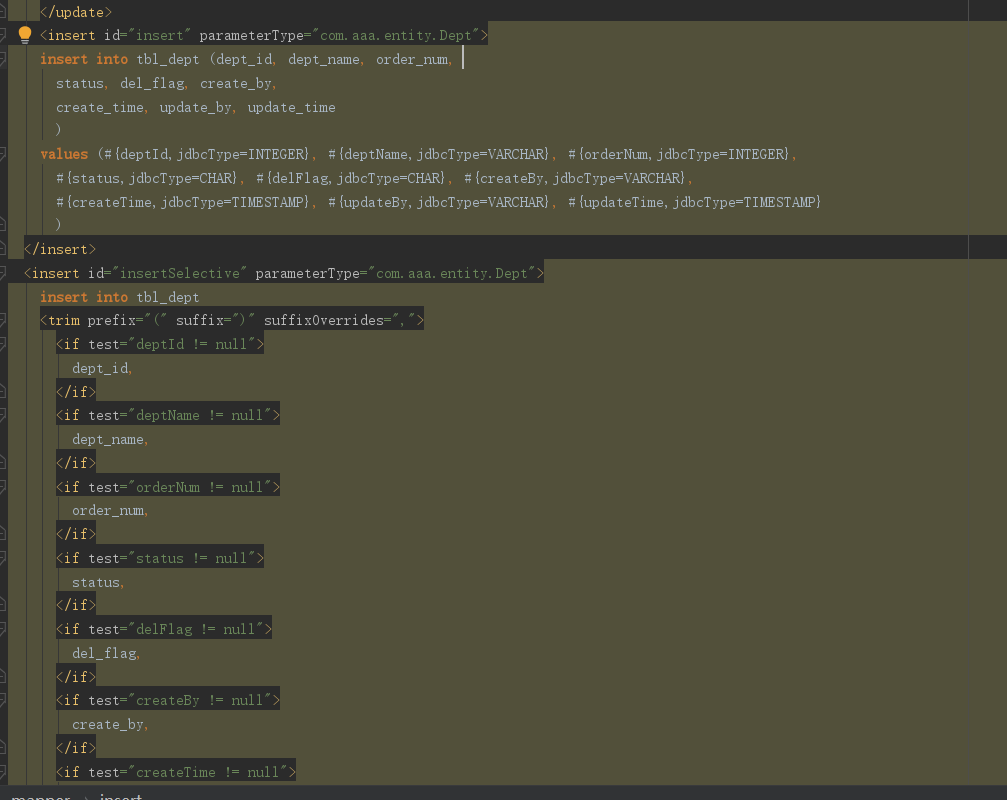
代码实现：

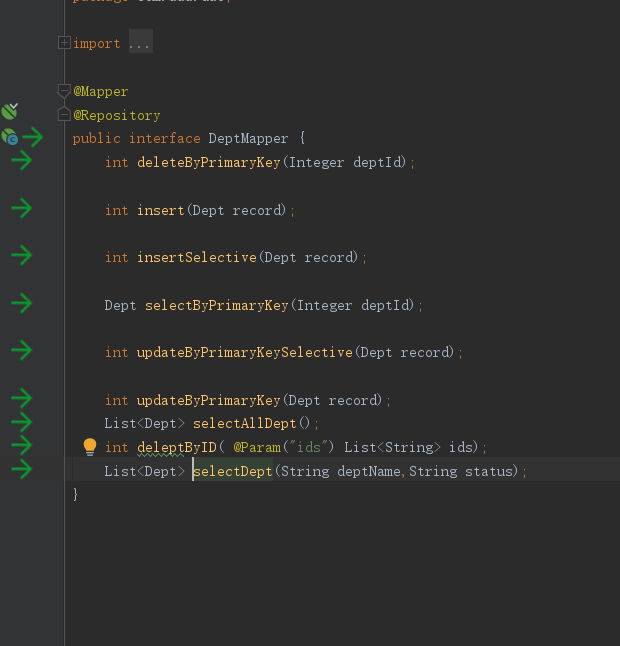
使用layUI的表格重载访问showDeptrow数据接口获得查询后的部门信息。

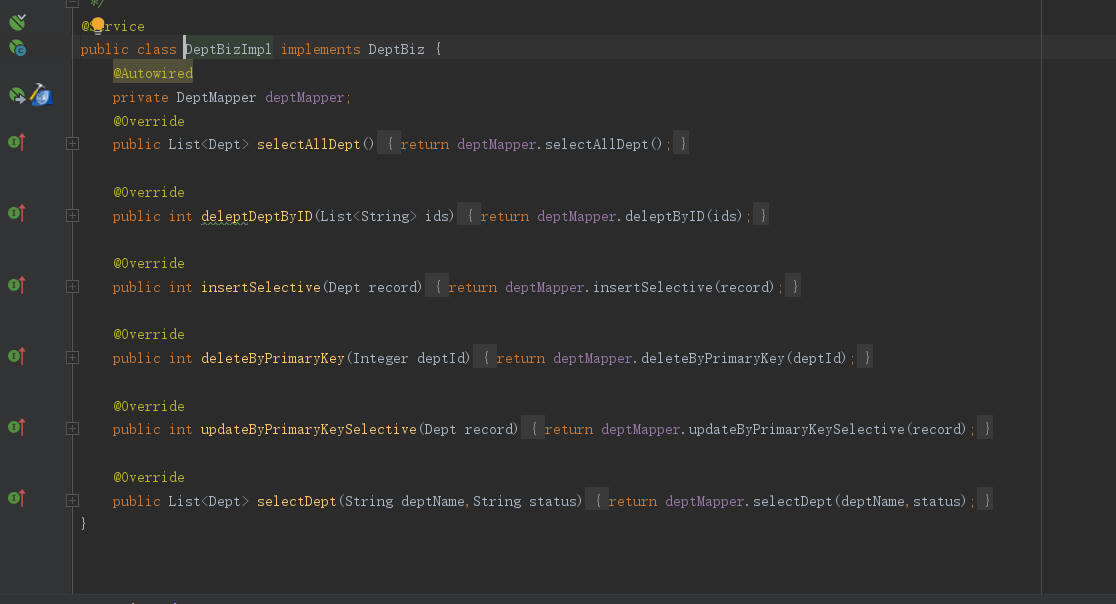


在后端实现增删改查分别在dao的deptmapper和deptmapper.xml、deptbiz、deptbizimpl的接口和实现类中实现相关方法，同时在controller中提供相关的访问接口来调用增删改查的方法，原理使用的都是老师上课讲的来实现的，html页面代码参考老师视频和layUI官网写的，下文不再赘述。



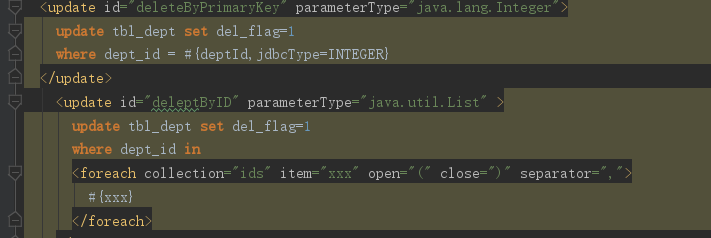




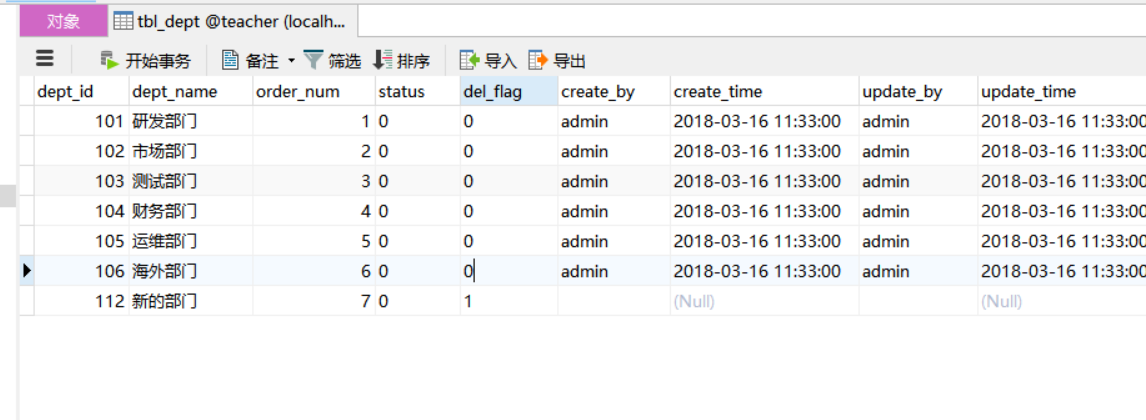


|  |
| --- |
| @Controller @RequestMapping("/dept") public class DeptController {  @Autowired  private DeptBiz deptBizImpl;  @RequestMapping("/toShowDept")  public String toShowDept(){  return "dept/showDept";  }   @RequestMapping("/showDept")  @ResponseBody  public LayUiTable showDept() {  //开始查询  List<Dept> deptList = deptBizImpl.selectAllDept();  LayUiTable layUiTable = new LayUiTable();  layUiTable.setCode(0);  layUiTable.setMsg("返回消息");  layUiTable.setData(deptList);  return layUiTable;  }   @RequestMapping("/showDeptrow")  @ResponseBody  public LayUiTable showDeptrow(String deptName,String status) {  //开始查询  List<Dept> deptList = deptBizImpl.selectDept(deptName,status);  LayUiTable layUiTable = new LayUiTable();  layUiTable.setCode(0);  layUiTable.setMsg("返回消息");  layUiTable.setData(deptList);  return layUiTable;  }   @RequestMapping("/saveDept")  @ResponseBody  public Object saveDept(Dept dept){  int i = deptBizImpl.insertSelective(dept);  Map map= new HashMap<>();  if(i>0){  map.put("code",MyConstants.*successCode*);  map.put("message",MyConstants.*saveSuccessMsg*);  }else {  map.put("code",MyConstants.*failCode*);  map.put("message",MyConstants.*saveFailMsg*);  }  return map;  }   */\*\*  \* 修改用户信息  \** ***@param*** *DeptInfo  \** ***@return*** *\*/* @RequestMapping("/editDept")  @ResponseBody  public Object editDept(Dept dept){  int i = deptBizImpl.updateByPrimaryKeySelective(dept);  Map map= new HashMap<>();  if(i>0){  map.put("code",MyConstants.*successCode*);  map.put("message",MyConstants.*editSuccessMsg*);  }else {  map.put("code",MyConstants.*failCode*);  map.put("message",MyConstants.*editFailMsg*);  }  return map;  }   @RequestMapping("/delDeptrow")  @ResponseBody  public Object delDeptrow(Integer deptId){  int i = deptBizImpl.deleteByPrimaryKey(deptId);  Map map= new HashMap<>();  if(i>0){  map.put("code",MyConstants.*successCode*);  map.put("message",MyConstants.*delSuccessMsg*);  }else {  map.put("code",MyConstants.*failCode*);  map.put("message", MyConstants.*delFailMsg*);  }  return map;  }   @RequestMapping("/delDept")  @ResponseBody  public Object delDept( @RequestParam(value = "ids") String ids){  //将json字符串转换成list对象  List<String> list= (List<String>) JSON.*parse*(ids);  int i = deptBizImpl.deleptDeptByID(list);  Map map= new HashMap<>();  if(i>0){  map.put("code",MyConstants.*successCode*);  map.put("message",MyConstants.*delSuccessMsg*);  }else {  map.put("code",MyConstants.*failCode*);  map.put("message", MyConstants.*delFailMsg*);  }  return map;  }  } |

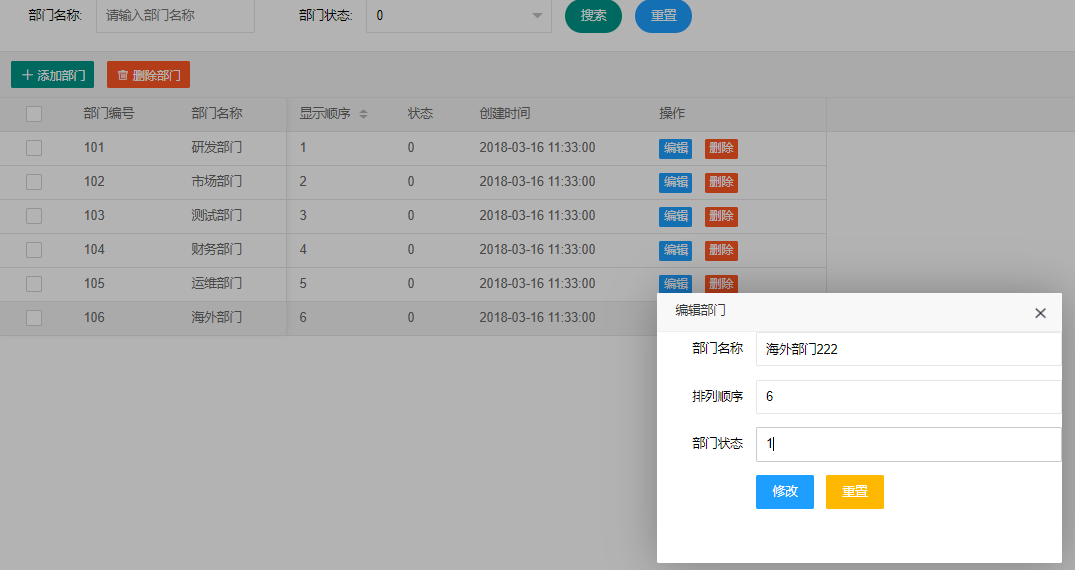
删除部门：

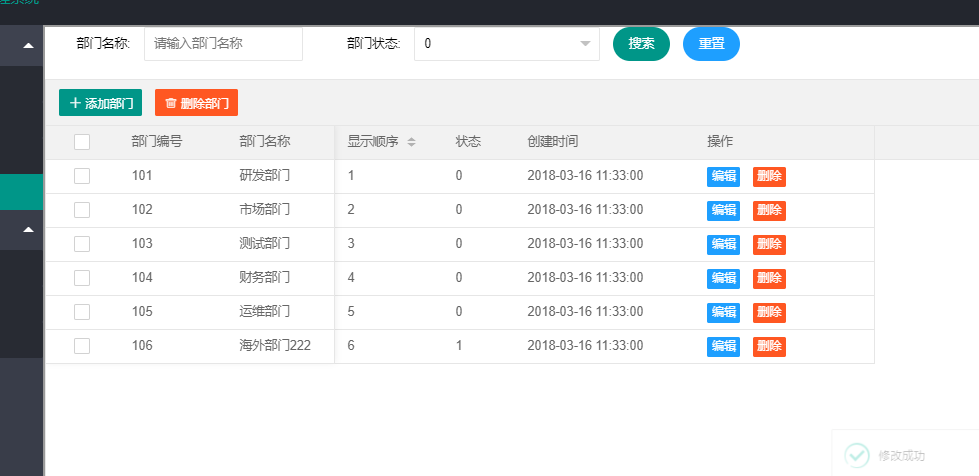


实现逻辑删除，可复原：



编辑部门：点击编辑出现原有值





1. 角色管理

4.1查询所有角色，并展示到页面中



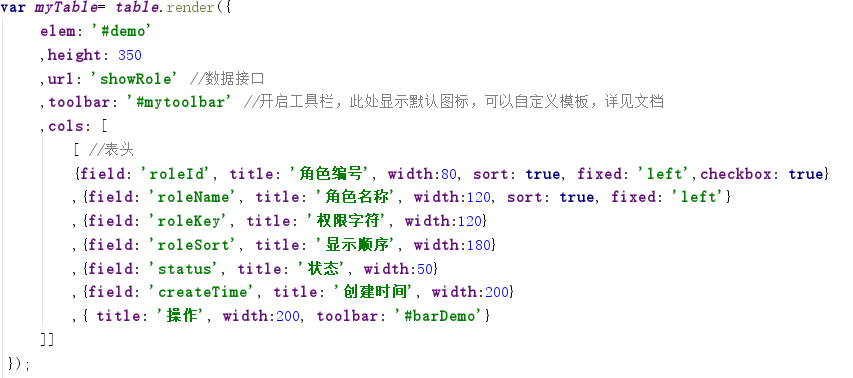
4.2添加角色



4.3修改角色



关键代码分析：

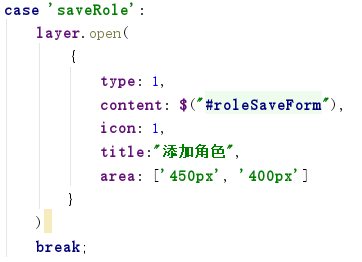
Showrole.html页面通过layui.render生成表格，其中field就是对应的列数据，title就是列名，数据通过[url:showRole连接到rolecontroller.java](../maven/hn-thymeleaf-layui-master/src/main/resources/doc/showRole连接到rolecontroller.java)的方法传到页面上。

对角色的添加关键代码：

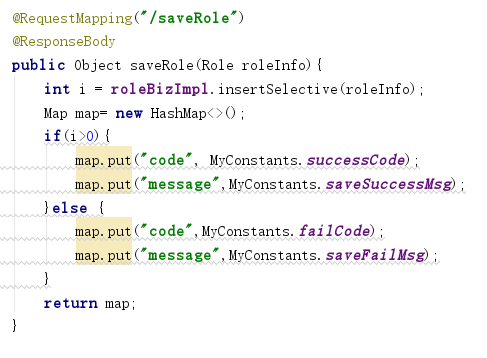
在js中加入添加按钮，并设置在点击时触发事件saveRole,



而在监听头工具栏事件的方法中，当接到触发事件的时候弹出#roleSaveForm的输入框。Title是输入框的标题，area两个直分别为宽和高。



在rolecontroller.java中，把弹出的输入框所输入的数据传到方法roleBizImpl中，然后再调用mapper里的SQL insert语句对数据库进行增加数据的操作。



对角色的编辑关键代码：

在showrole.html中，在每一行的数据后面都增加了一个按钮“编辑”



当点击按钮时，触发点击事件edit.在edit触发的时候就会被行工具监听事件监听到，然后在弹出的输入框中把数据放入文本框中以便对数据进行更改。



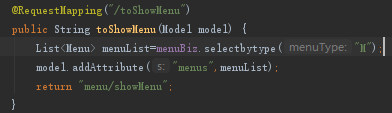
在rolecontrller.java中，同理把修改后的数据传到mapper的update方法中，以更改数据库的文件。



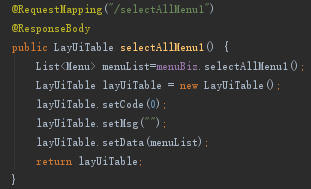
1. 菜单管理

5.1查询菜单

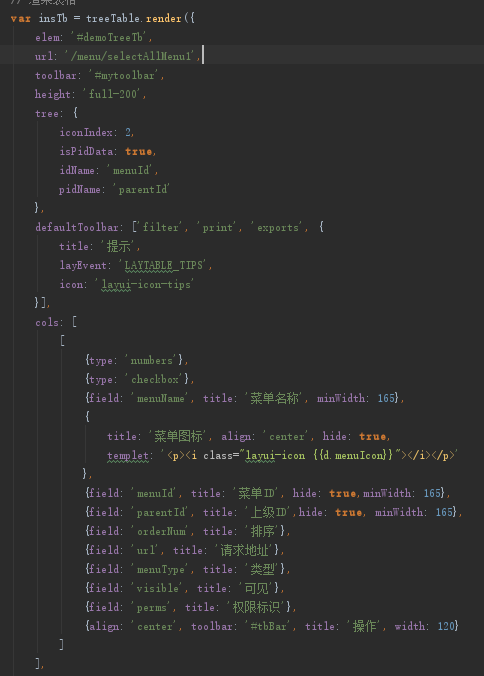
步骤一：点击菜单管理显示菜单管理页面



步骤二：查询所有菜单，并组装成LayUiTable对象



步骤三：在页面显示查询到的菜单



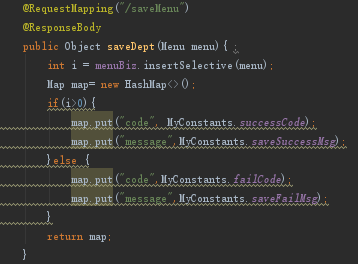
5.2 添加菜单

步骤一：监听toolbar头部点击添加菜单按钮事件，并显示添加菜单的表单





步骤二：监听表单MenuSaveForm里的保存按钮点击事件，把表单的信息传给saveMenu，然后添加入菜单表中，并对返回值进行判断。



5.3修改菜单

步骤一：监听工具列的修改点击事件，先把改行的值传给修改表单MenuEditForm,然后显示该表单



步骤二：监听表单MenuEditForm里的修改点击事件，把表单的值传给editMenu，然后在数据库根据MenuId进行修改，并对返回值进行判断



