天喵公司后台管理系统需求规格说明书

|  |  |
| --- | --- |
| 开发目标 | 开发大纲 |
| 1、掌握软件开发的基本流程 | 1、背景知识 |
| 2、掌握springboot框架 | 2、需求分析 |
| 3、掌握mybatis框架各知识点 | 3、功能分析 |
| 4、掌握shiro安全框架的认证和授权 | 4、架构分析 |
| 5、掌握html和javascript各知识点 | 5、数据库设计 |
| 6、掌握mysql各知识点 | 6、代码开发参考 |

## 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. **部门管理**

系统对应角色分配不同不同部门所属，以便公司进行合理的规划运营，促进企业合理运转。

1. **菜单管理**

菜单管理中通过用户名查询相应对应角色身份，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

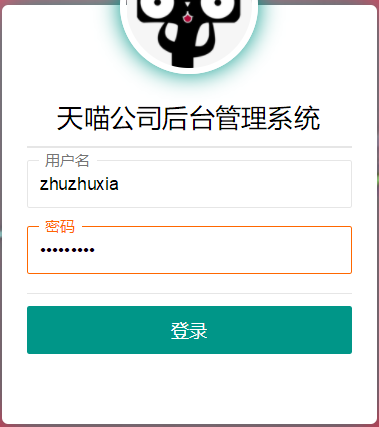
1. 登录功能
   1. 登录页面

使用用户名：猪猪侠，密码：zhuzhuxia 登录

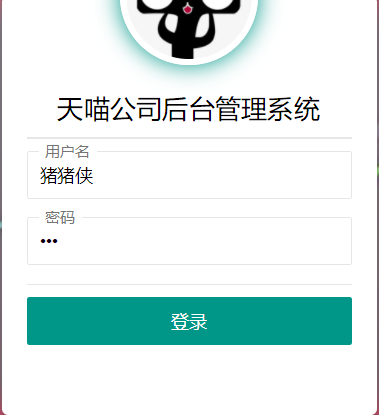


* 1. 异常登录提示错误信息

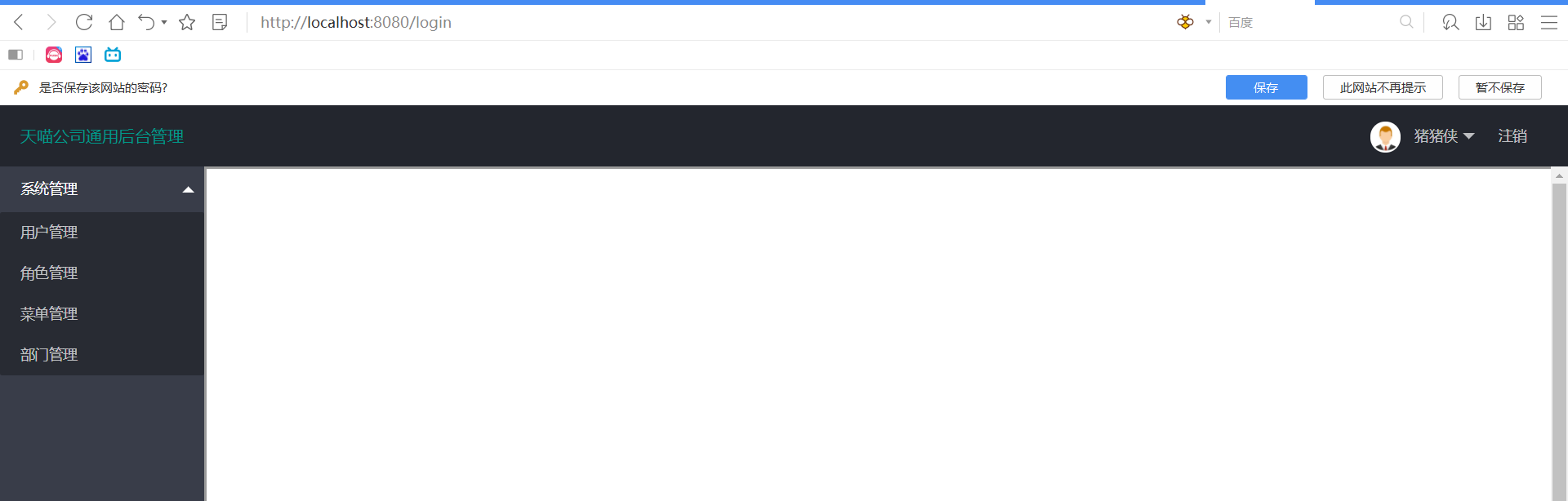
A、使用用户名：zhuzhuxia，密码：zhuzhuxia 登录，点击登录，提示：用户名错误！

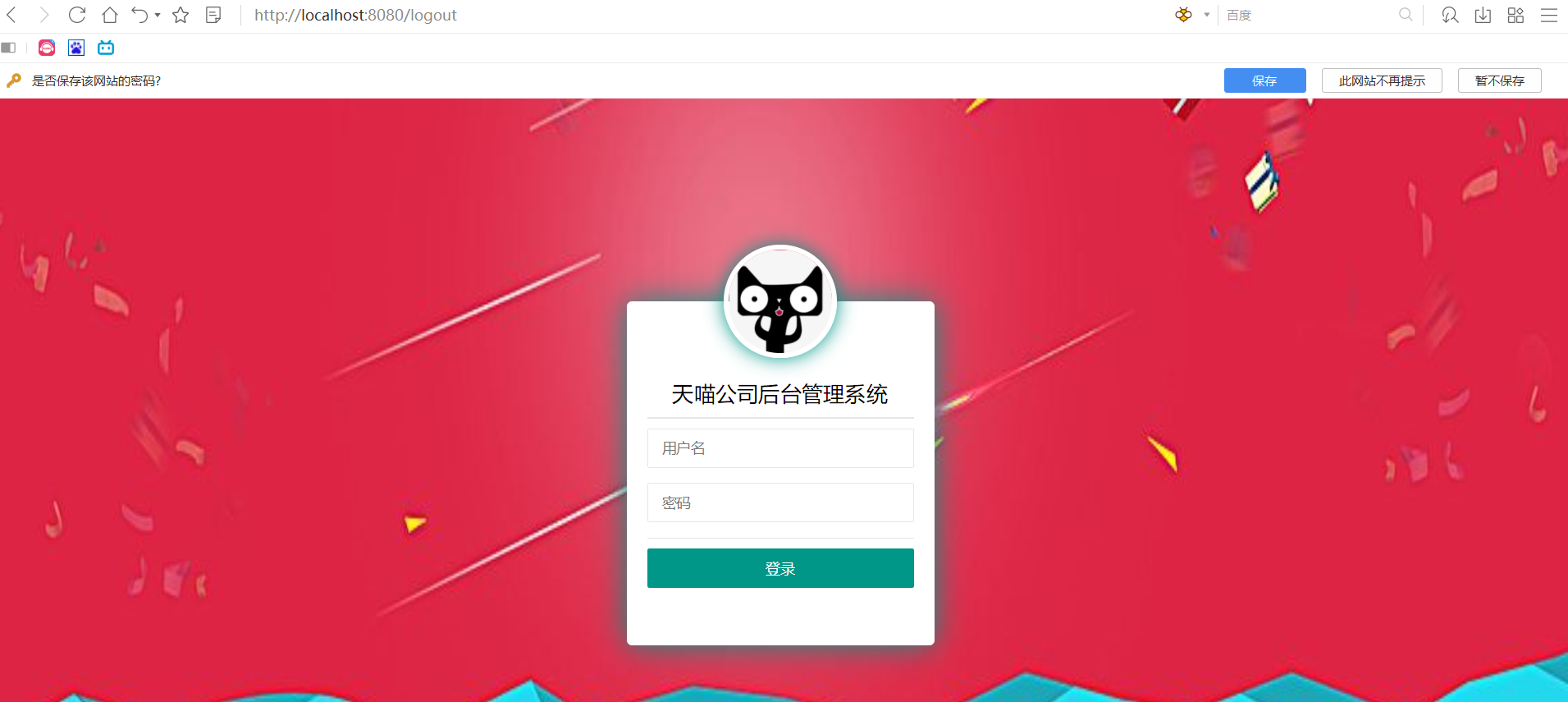
B、使用用户名：zhuzhuxia，密码：123 登录，点击登录，提示：密码错误！

* 1. 主页页面



1.4注销



1. 用户管理
   1. 点击二级表头：用户管理，并显示用户管理表



* 1. 用户管理表——添加功能

输入测试数据，点击保存

弹出提示：添加成功！



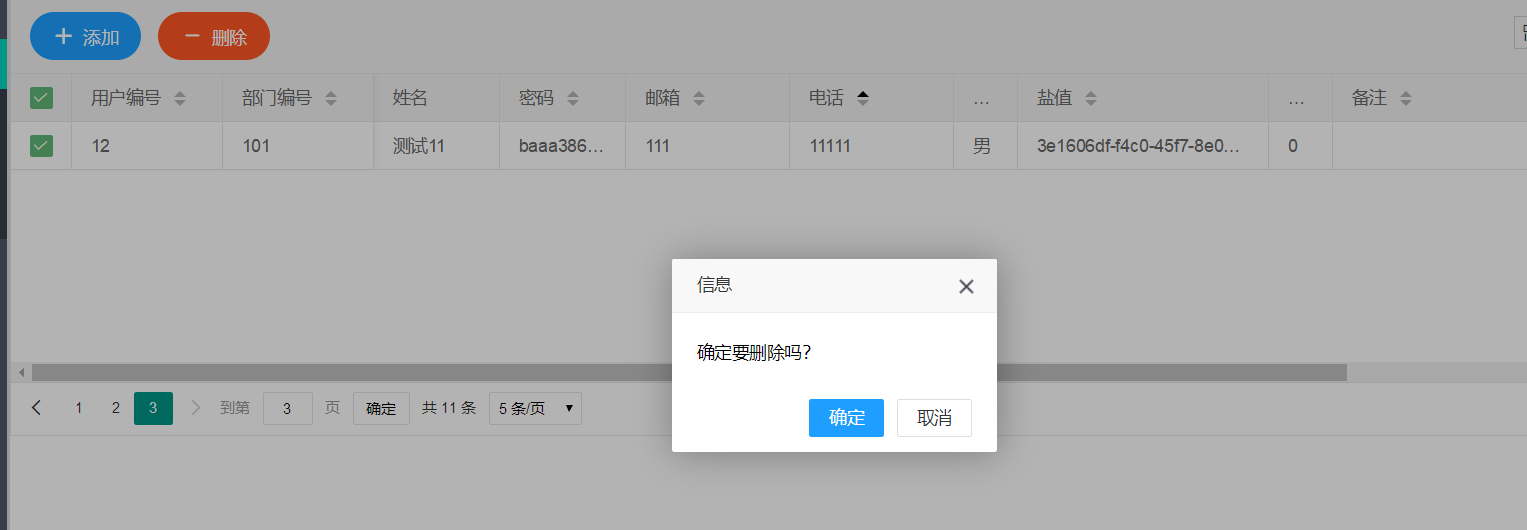
用户表中最后一页更新显示出新添数据



* 1. 用户管理表——删除用户功能

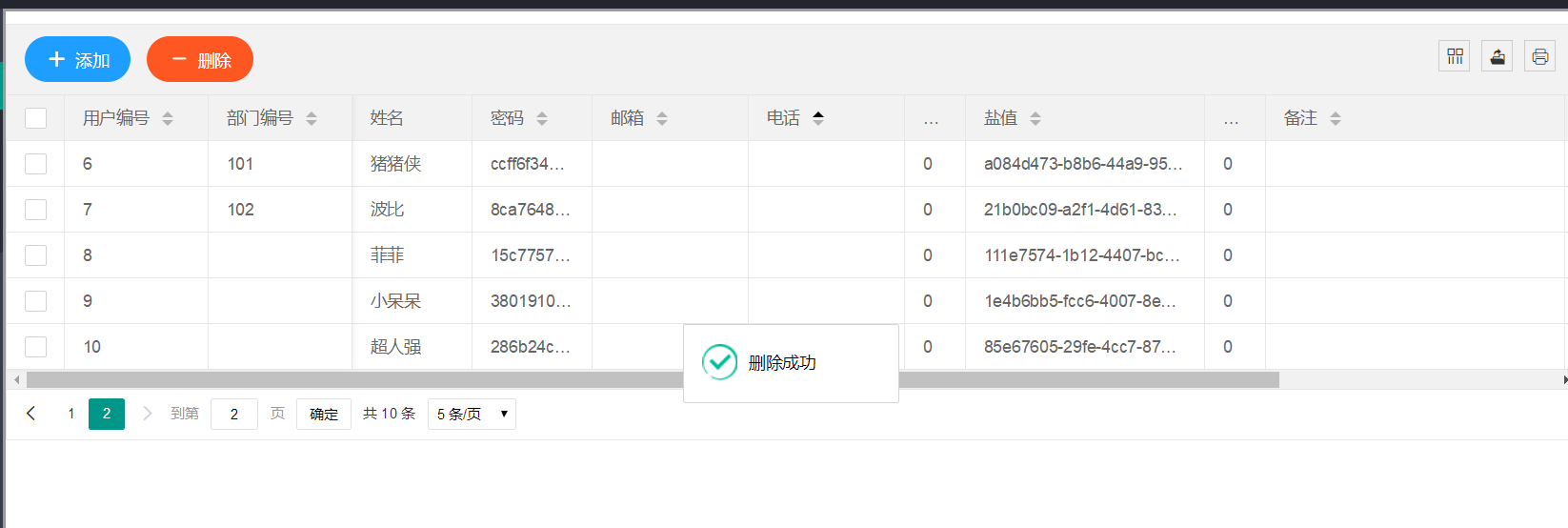
点击删除按钮，弹出提示：

选择任意行，再点击删除按钮：弹出如下提示：



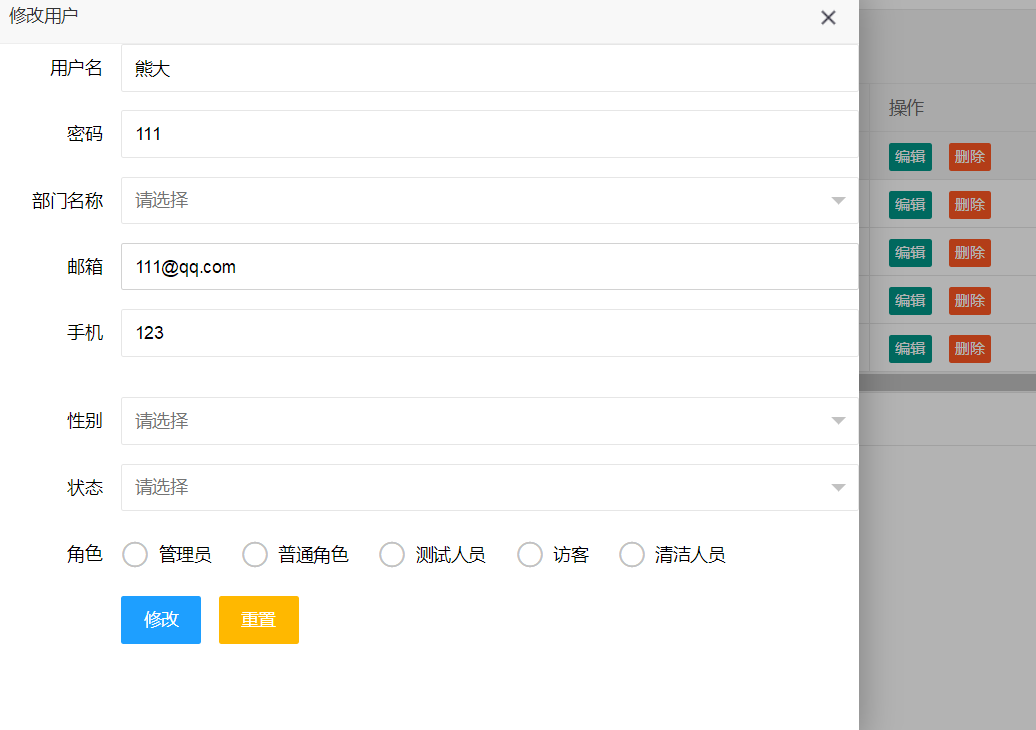
点击确定按钮，弹出提示：删除成功！

相应表中显示更新，并删除选择内容

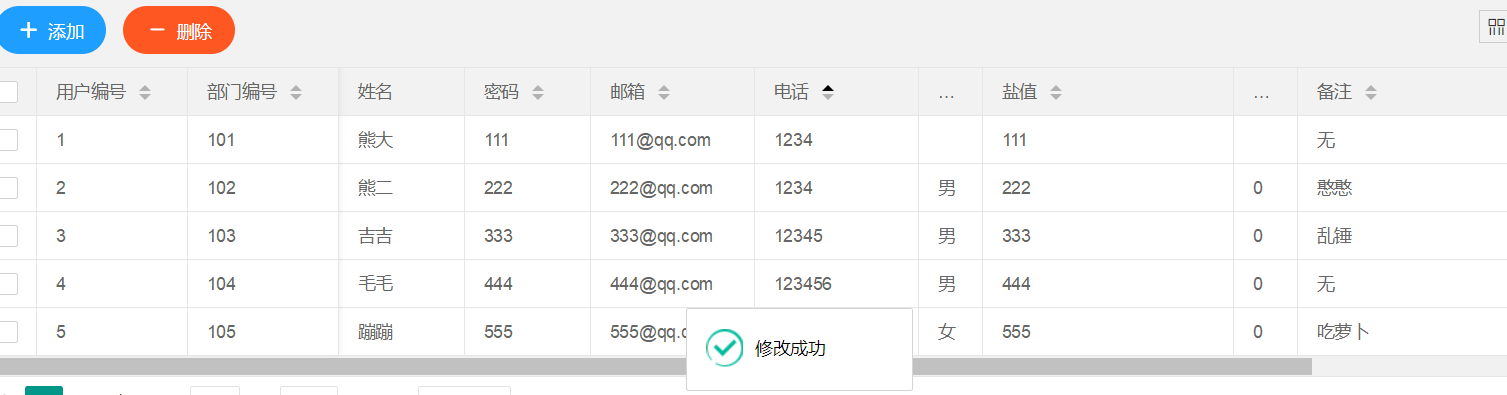


* 1. 用户管理表——编辑（修改）用户功能

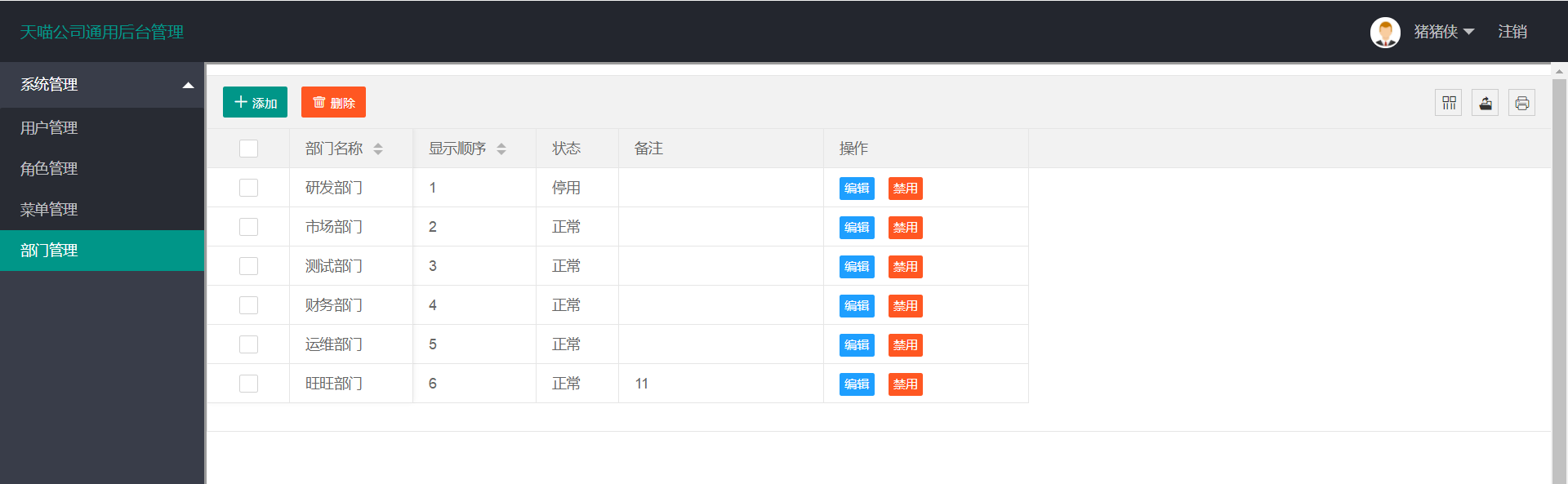
点击右侧编辑按钮，弹出相应对话框：任意修改，如更改手机号码，再点击修改按钮

点击后，产生响应，弹出提示：修改成功！



1. 部门管理
   1. 点击二级表头：部门管理，并显示部门管理表



* 1. 部门管理——添加部门功能

点击添加按钮，弹出相应对话框：输入任意测试数据后，点击保存按钮

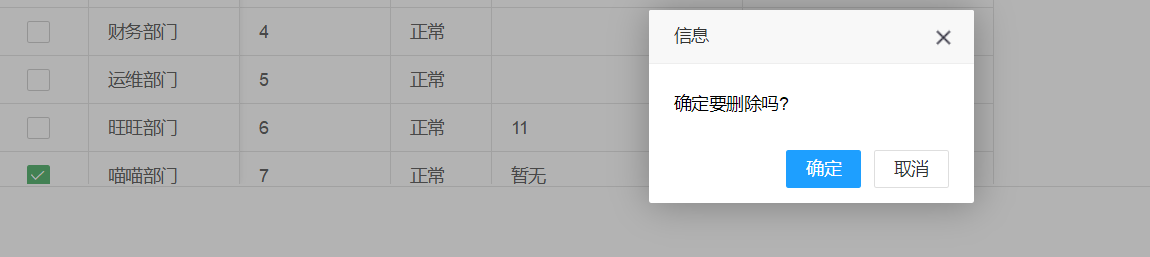
弹出提示：添加成功！表单中成功添加响应。



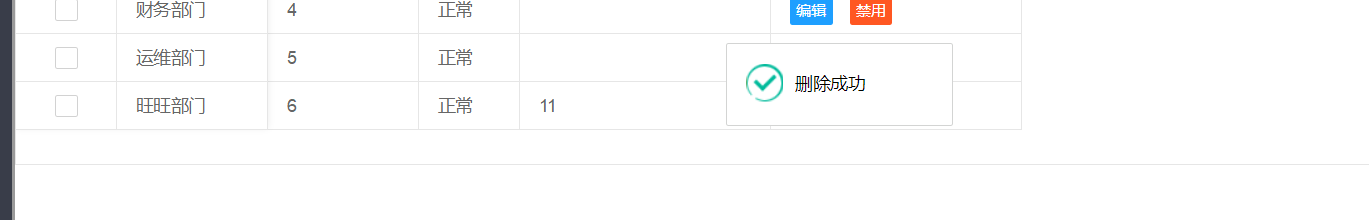
* 1. 部门管理——删除部门功能

点击删除按钮，弹出提示: 

选择任意行，再点击删除按钮：弹出如下提示：



点击确定按钮，相应表中显示更新，并删除选择内容



* 1. 部门管理——编辑（修改）部门功能

点击编辑按钮，跳出对话框，输入任意测试数据，如，更改部门名称



点击修改按钮，弹出提示：修改成功！

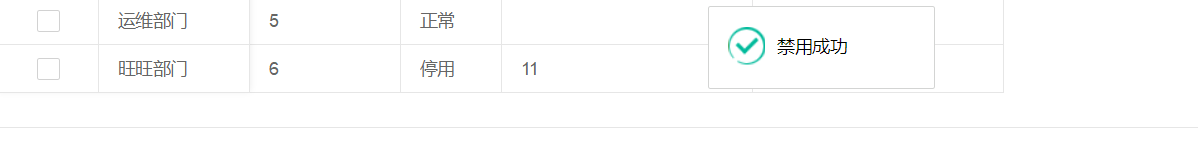


* 1. 部门管理——禁用部门功能

选择任意部门，点击禁用按钮，弹出提示：



点击确定按钮后，弹出提示：禁用成功！相应部门状态变更为：停用状态



1. 角色管理
   1. 点击二级表头：角色管理，并显示角色管理表



* 1. 角色管理表——添加角色功能

点击上方添加按钮，输入测试数据



点击保存按钮，弹出提示，并更新响应



* 1. 角色管理表——编辑（修改）角色功能

选择任意行，点击的操作栏的编辑按钮，如下图所示，输入测试更改数据



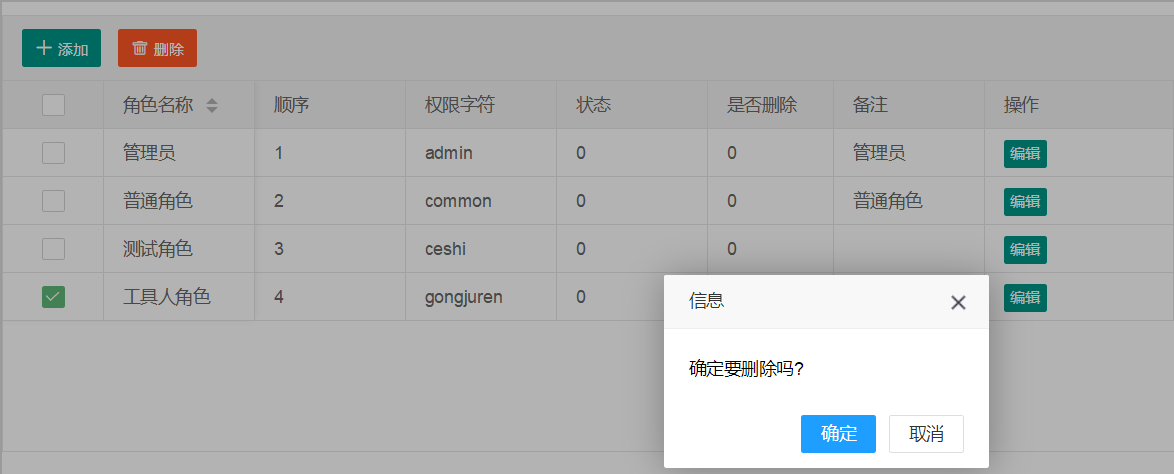
输入完毕，点击保存按钮，弹出提示框，并响应更新



* 1. 角色管理表——删除角色功能

点击删除按钮，弹出提示：

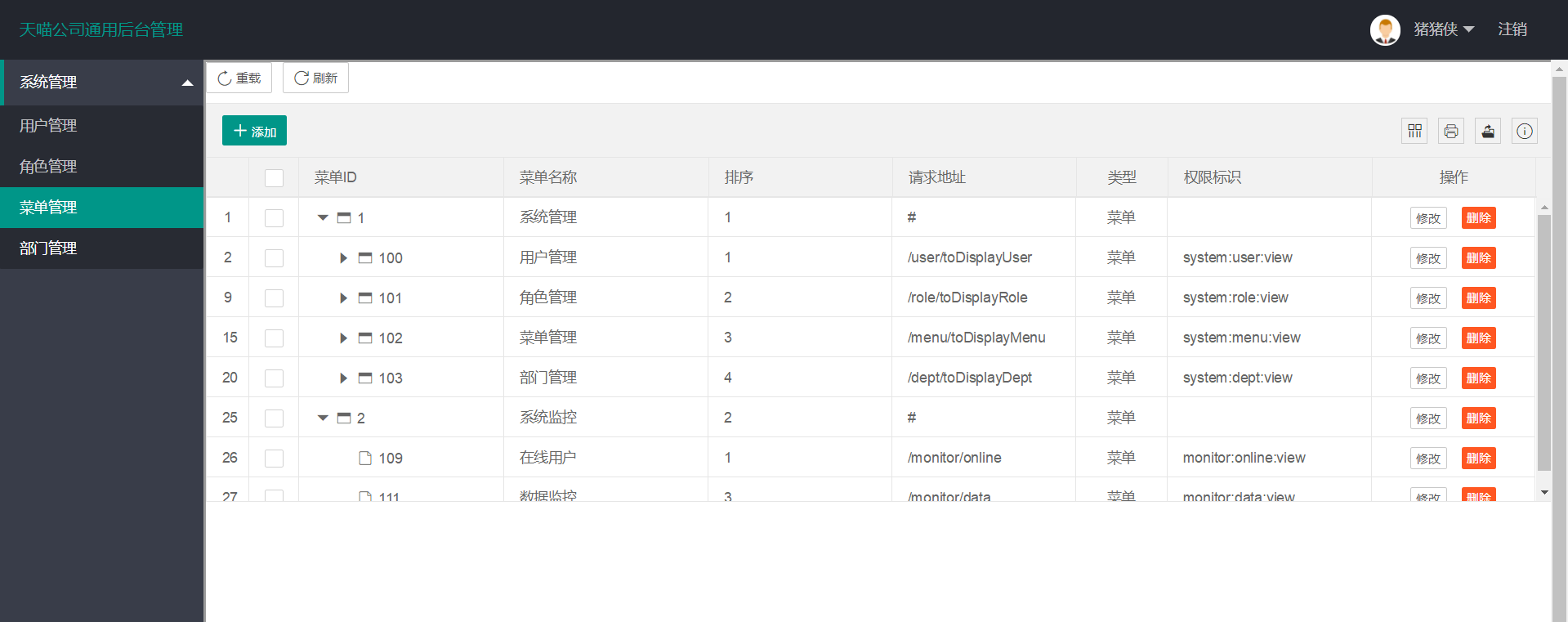
选择任意行，如最后一行，弹出对话框：



点击确认按钮，删除成功，并更新响应



1. 菜单管理
   1. 点击二级表头：菜单管理，并显示菜单管理表



* 1. 菜单管理——重载及刷新功能

点击重载或者刷新，菜单管理表单重新加载完毕



* 1. 菜单管理——添加菜单功能管理

点击添加按钮，弹出对话框，输入测试数据，如右下图所示：

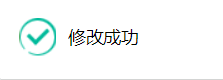
点击保存，添加一级菜单成功！



* 1. 菜单管理——修改菜单功能

点击任意栏的修改按钮，如，部门管理。输入更改数据



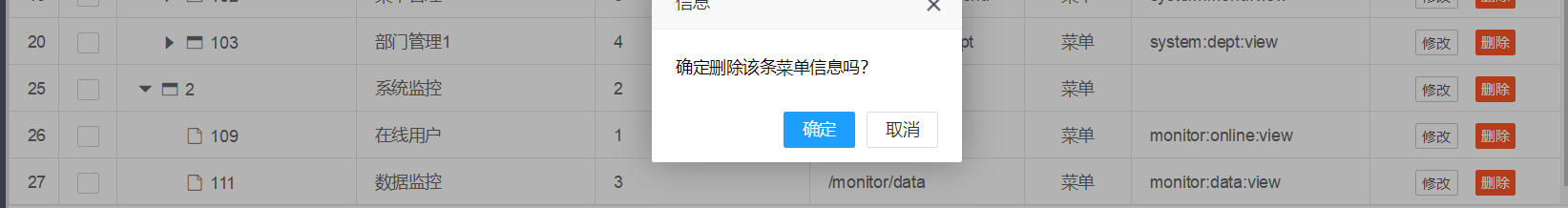
点击修改，弹出提示，

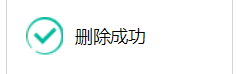
下拉后，修改数值更新成功。



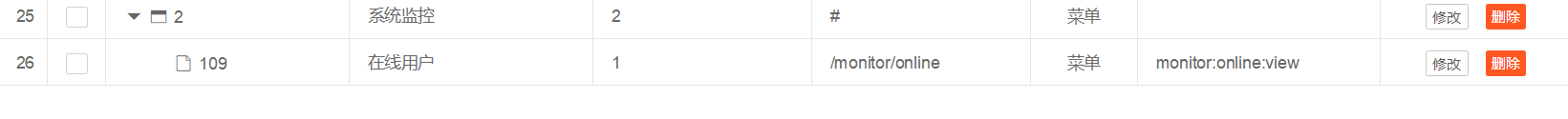
* 1. 菜单管理——删除菜单功能

如，点击数据监控菜单栏右侧的删除按钮，弹出如下对话框

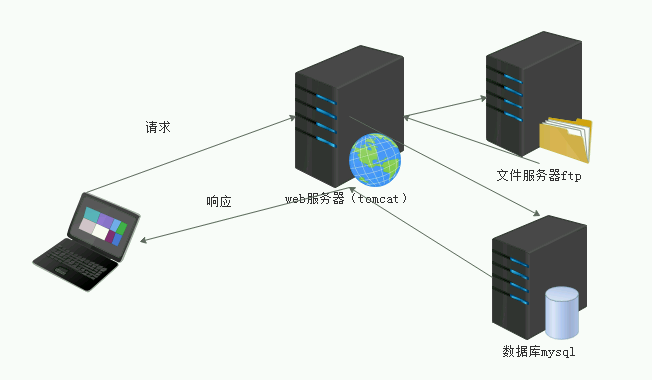


点击确定按钮，弹出提示：

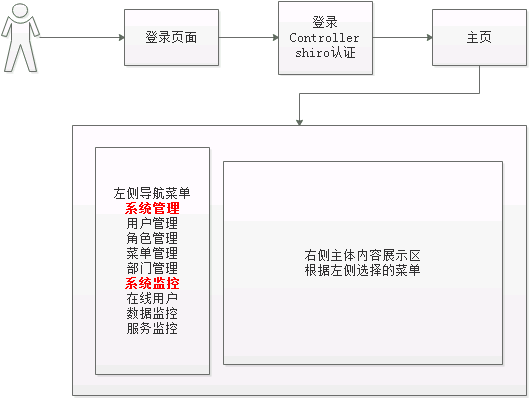
对应下拉菜单，去除删除项并更新



## 4、架构分析

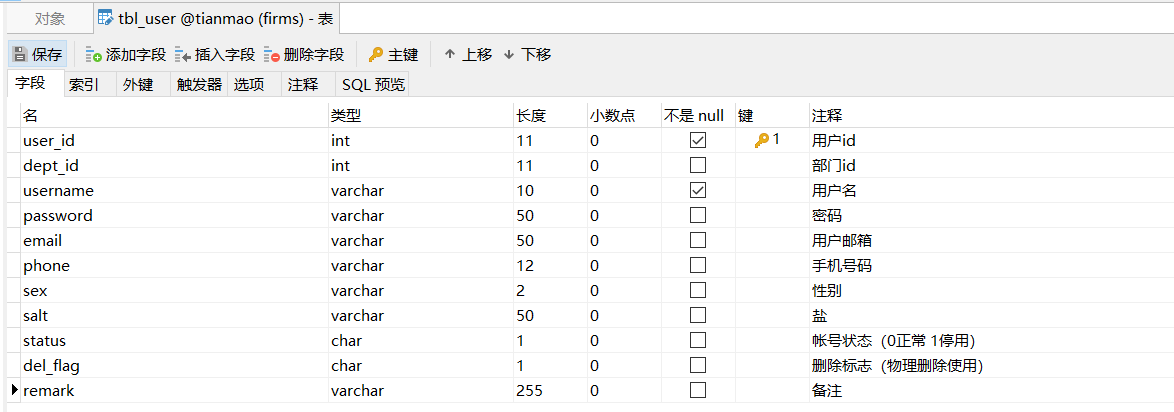


### 4.1、整体架构设计

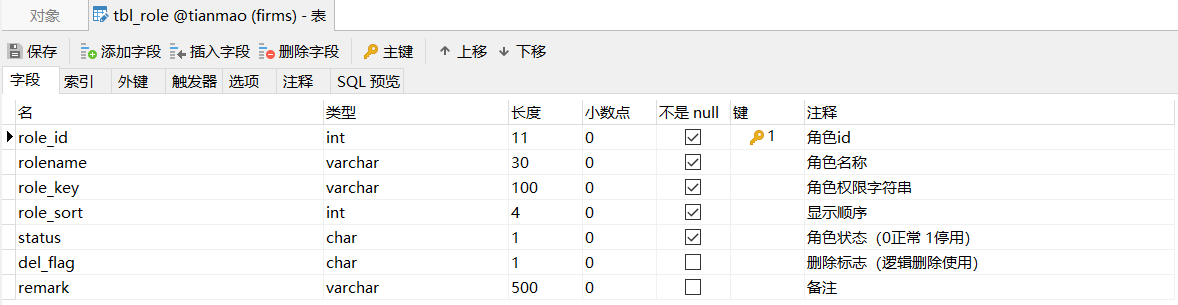


### 5、数据库设计

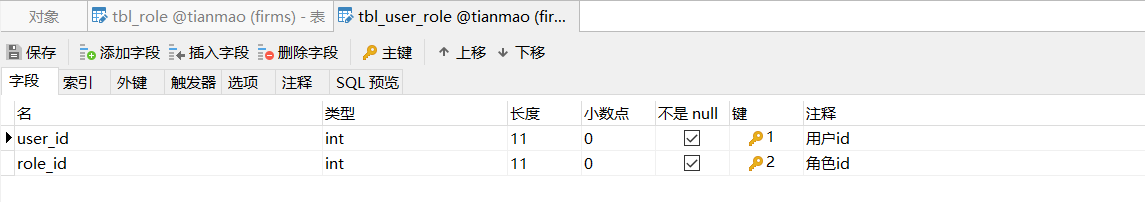
#### 5.1、用户表



#### 5.2、角色表



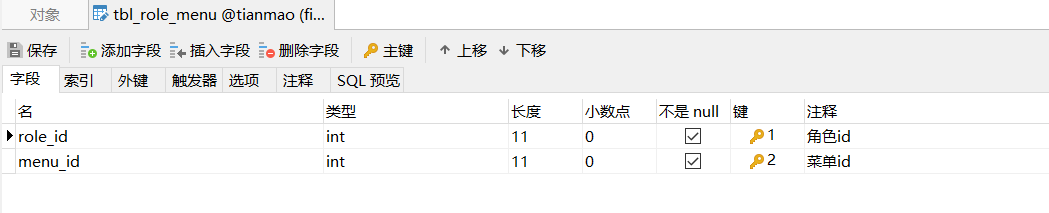
#### 5.3、用户角色表（中间表）



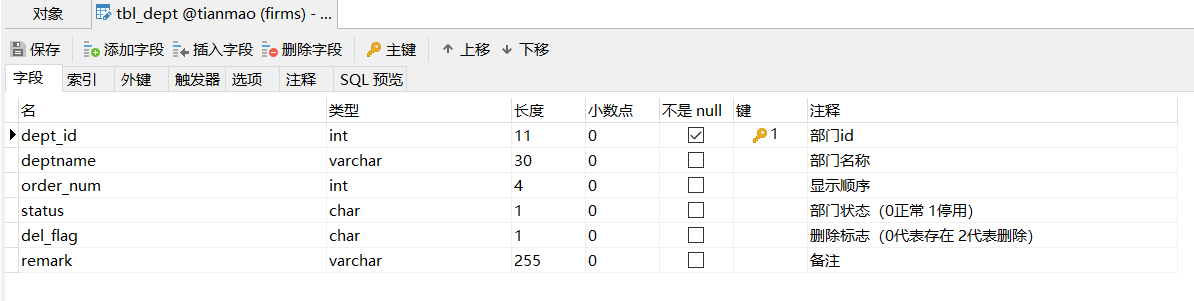
#### 5.4、菜单表



#### 5.5、角色菜单表

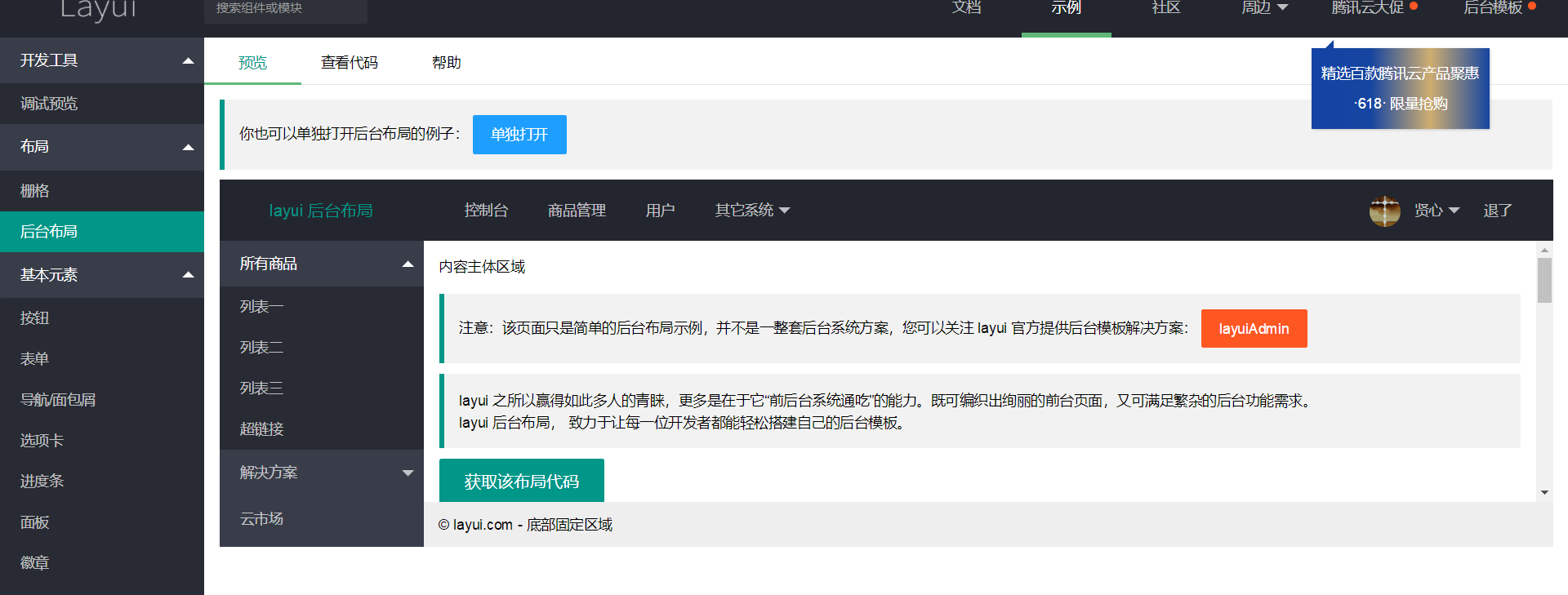


#### 5.6、部门表



## 代码开发参考

1. Layui后台布局



2、Layui树形组件



3、树形设计涉及代码：

步骤一：根据json字符串封装java对象（gsonformat插件）

|  |
| --- |
| public class LayUiTree {   private String title;  private int id;  private String field;  private boolean checked;  private boolean spread;  private List<LayUiTree> children;   } |

步骤二：查询所有的菜单，并且组装成LayUiTree 对象

|  |
| --- |
| package com.aaa.util;   import com.aaa.entity.LayUiTree; import com.aaa.entity.Menu;  import java.util.ArrayList; import java.util.Iterator; import java.util.List;  */\*\*  \* 权限数据处理  \*   \** ***@author*** *teacherChen  \*/* public class TreeUtils {  */\*\*  \* 根据父节点的ID获取所有子节点  \*   \** ***@param*** *list 分类表  \** ***@param*** *parentId 传入的父节点ID  \** ***@return*** *String  \*/* public static List<LayUiTree> getChildPerms(List<Menu> list, int parentId)  {  List<LayUiTree> returnList = new ArrayList<LayUiTree>();  for (Menu menu : list) {  // 一、根据传入的某个父节点ID,遍历该父节点的所有子节点  if(menu.getParentId()==parentId){  //构造tree对象  LayUiTree tree= new LayUiTree();  tree.setId(menu.getMenuId());  tree.setTitle(menu.getMenuName());  tree.setChecked(false);  //开始递归，把所有菜单和当前菜单放入  *recursionFn*(list, tree);  returnList.add(tree);  }  }  return returnList;  }   */\*\*  \* 递归列表  \*   \** ***@param*** *list  \** ***@param*** *t  \*/* private static void recursionFn(List<Menu> list, LayUiTree t)  {  // 得到t的子节点列表  List<LayUiTree> childList = *getChildList*(list, t);  t.setChildren(childList);  for (LayUiTree tChild : childList)  {  if (*hasChild*(list, tChild))  {  // 判断是否有子节点  Iterator<LayUiTree> it = childList.iterator();  while (it.hasNext())  {  LayUiTree n = it.next();  *recursionFn*(list, n);  }  }  }  }   */\*\*  \* 得到子节点列表  \*/* private static List<LayUiTree> getChildList(List<Menu> list, LayUiTree t)  {   List<LayUiTree> tlist = new ArrayList<LayUiTree>();  Iterator<Menu> it = list.iterator();  while (it.hasNext())  {  Menu n = (Menu) it.next();  if (n.getParentId() == t.getId())  {  LayUiTree tree= new LayUiTree();  tree.setId(n.getMenuId());  tree.setTitle(n.getMenuName());  tree.setChecked(false);  tlist.add(tree);  }  }  return tlist;  }   */\*\*  \* 判断是否有子节点  \*/* private static boolean hasChild(List<Menu> list, LayUiTree t)  {  int size = *getChildList*(list, t).size();  if(size>0){  return true;  }else  {  return false;  }  }  } |