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

页面已部署上云服务器: http://120.78.120.128:8081/login

1. 登录功能
   1. 登录页面

账号admin密码123

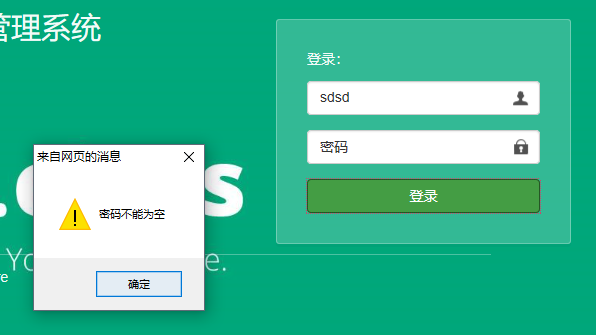
账号:jj密码:jj

均可登陆



* 1. 异常登录提示错误信息

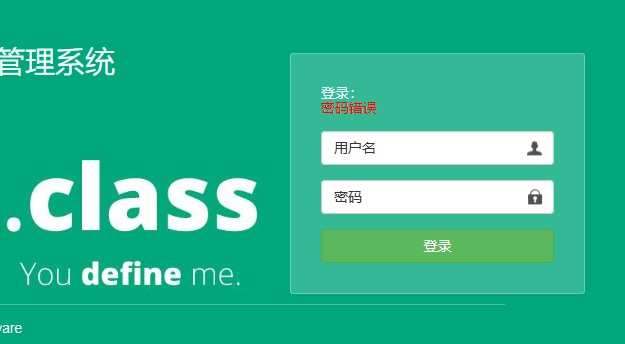
未输入密码:



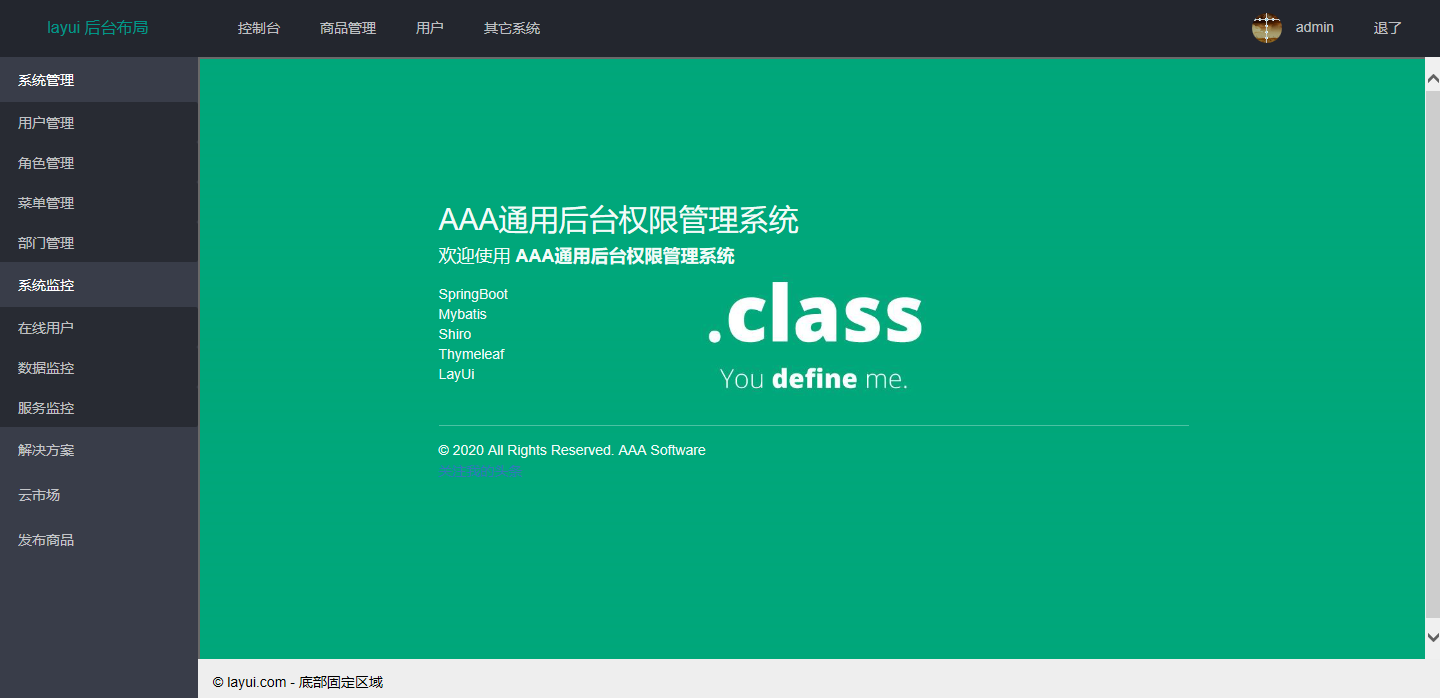
未输入用户名:



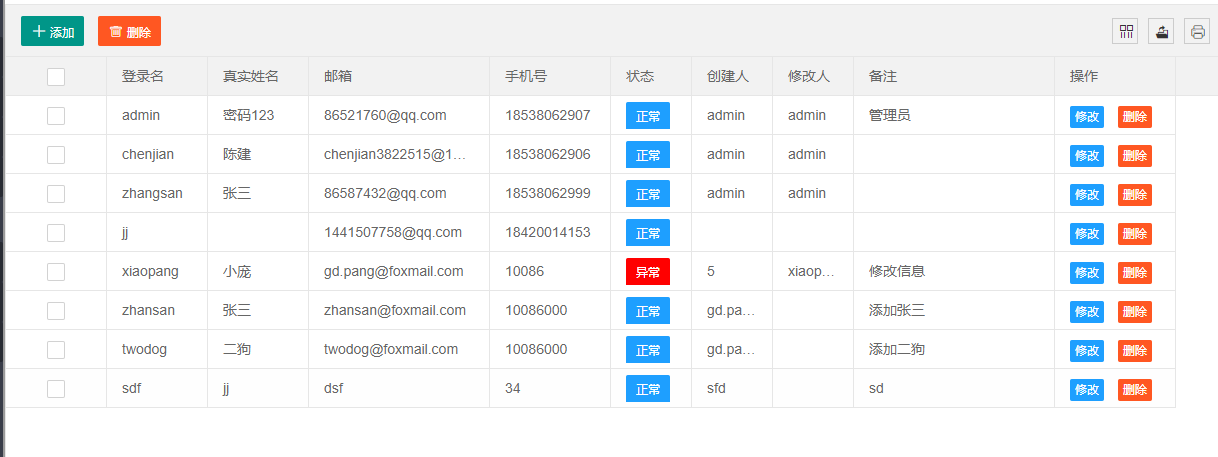
输入错误密码:



登陆成功后跳转页面:



1. 用户管理
   1. 查询用户



物理删除

delete from table where userid=1

逻辑删除（关键数据）

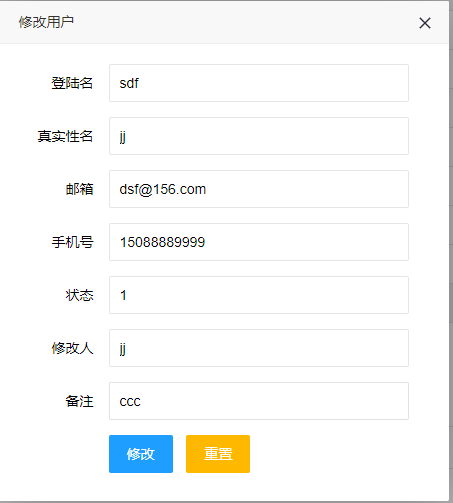
del\_flag 0 未删除 1 删除

update table set del\_flag=1 where userid=1

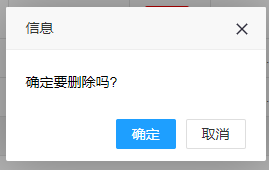
* 1. 添加用户，设置角色



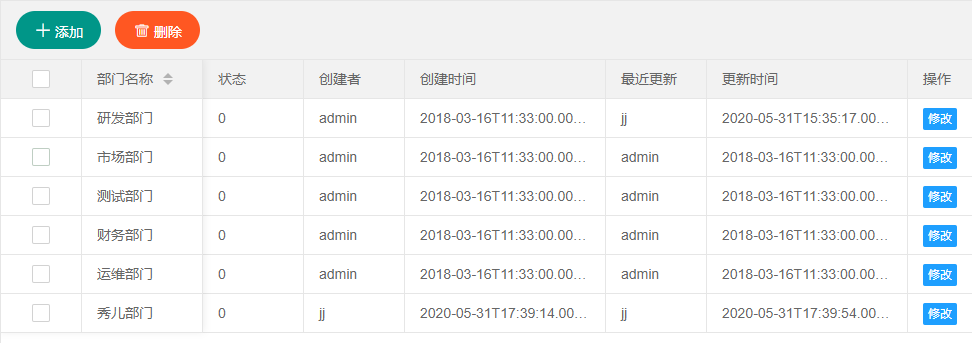
* 1. 修改用户，禁用用户



* 1. 删除用户



1. 部门管理
   1. 查询部门



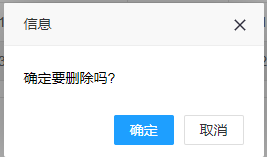
* 1. 添加部门



* 1. 修改部门，禁用部门



* 1. 删除部门



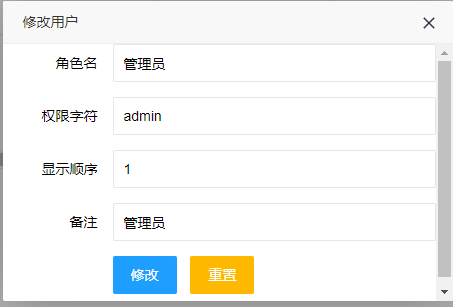
1. 角色管理
   1. 查询角色



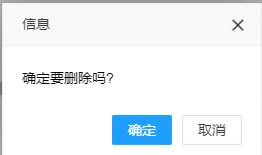
* 1. 添加角色



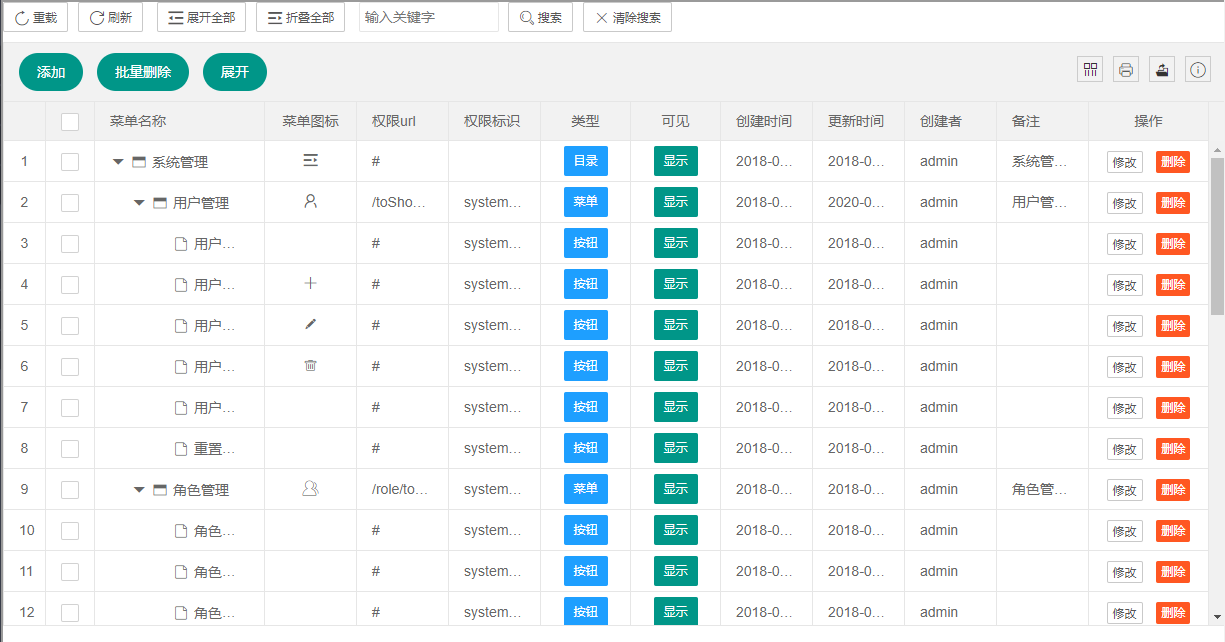
* 1. 修改角色



* 1. 删除角色



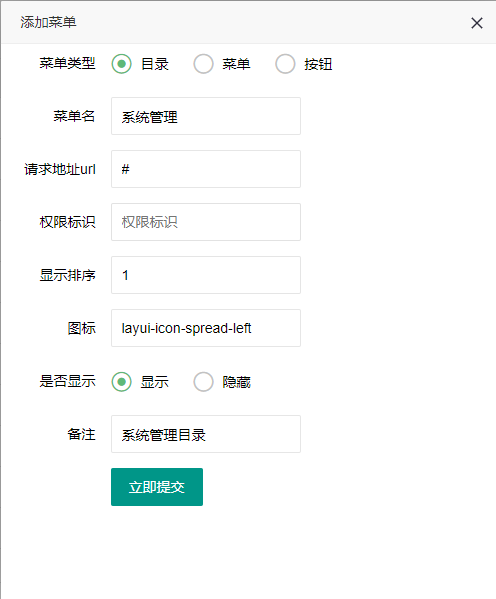
1. 菜单管理
   1. 查询菜单



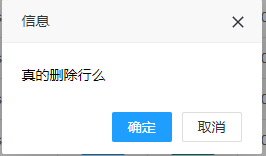
* 1. 添加菜单



* 1. 修改菜单



* 1. 删除菜单

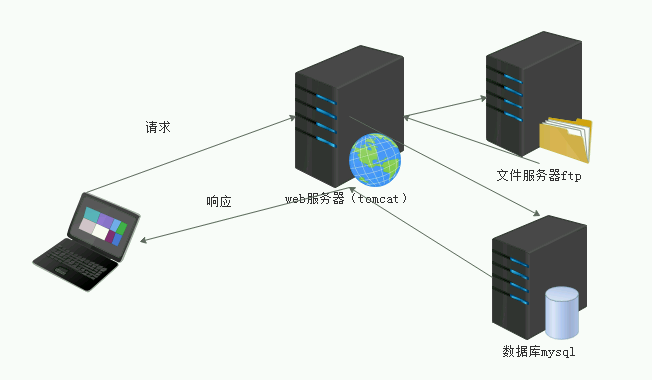


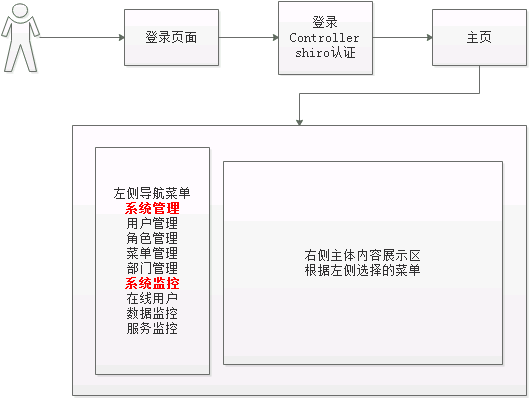
## 4、原型设计

产品经理设计产品原形

## 5、架构设计

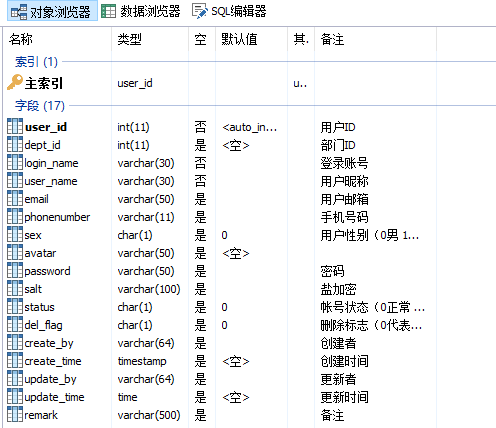
### 5.1、整体架构设计



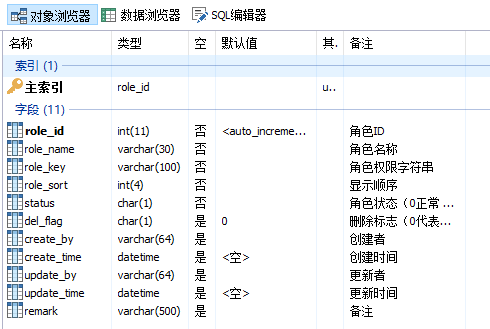


### 5.7、数据模型设计

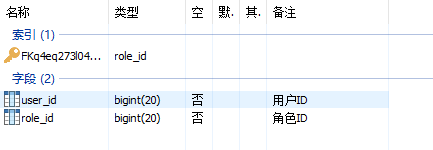
#### 5.7.1、用户表



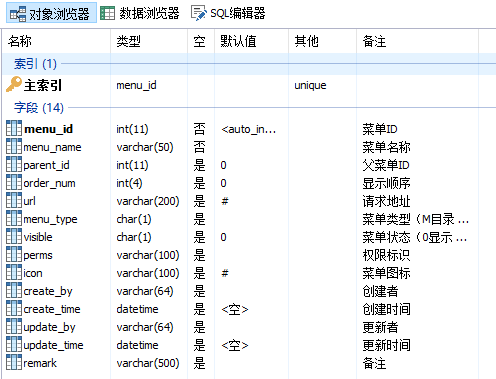
#### 5.7.2、角色表



#### 5.7.3、用户角色表（中间表）



#### 5.7.4、菜单表



#### 5.7.5、角色菜单表



#### 5.7.6、部门表



## 代码开发

开发难点：

1. 登陆界面

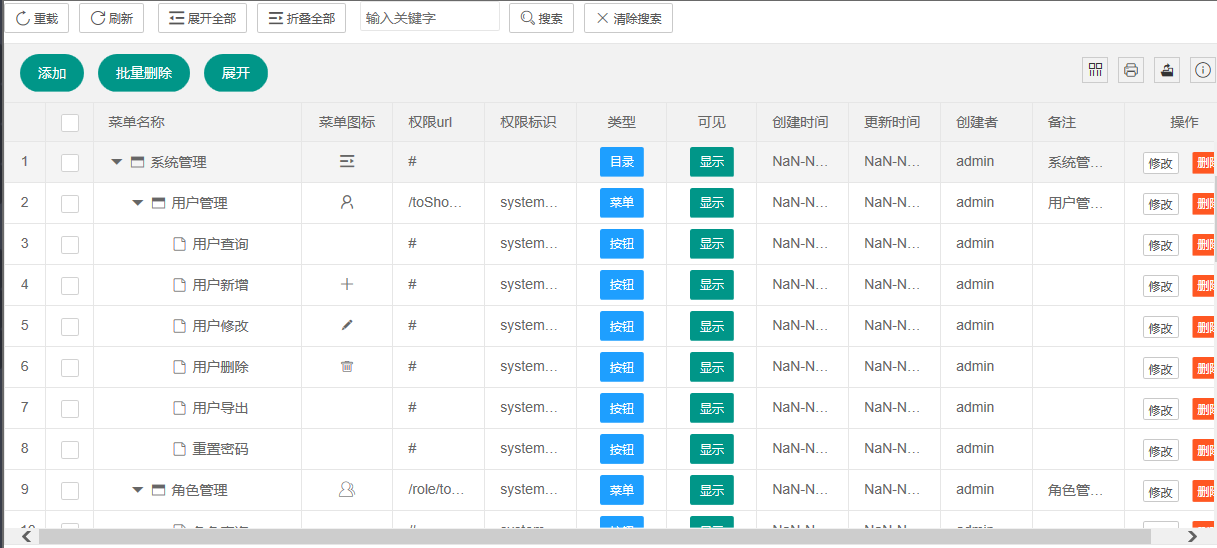


测试登陆账号为：admin 密码123

账号：jj 密码：jj

<!DOCTYPE html>  
<html lang="zh" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">  
 <title>AAA通用后台权限管理系统</title>  
 <meta name="keywords" content="AAA通用后台权限管理系统">  
 <meta name="description" content="AAA通用后台权限管理系统">  
 <link rel="shortcut icon" th:href="@{/img/favicon.ico}"/>  
 <link th:href="@{/css/bootstrap.min.css}" rel="stylesheet"/>  
 <link th:href="@{/css/login.css}" rel="stylesheet"/>  
 <link rel="stylesheet" th:href="@{/layui/css/layui.css}">  
 <!-- 全局js -->  
 <script th:src="@{/js/jquery-1.10.2.js}"></script>  
  
</head>  
<body class="signin">  
<div class="signinpanel">  
 <div class="row">  
 <div class="col-sm-7">  
 <div class="signin-info">  
 <div class="logopanel m-b">  
 <h2>AAA通用后台权限管理系统</h2>  
 </div>  
 <div class="m-b"></div>  
 <h4>欢迎使用 <strong>AAA通用后台权限管理系统</strong></h4>  
 <ul class="m-b">  
 <li><i class="fa fa-arrow-circle-o-right m-r-xs"></i> SpringBoot</li>  
 <li><i class="fa fa-arrow-circle-o-right m-r-xs"></i> Mybatis</li>  
 <li><i class="fa fa-arrow-circle-o-right m-r-xs"></i> Shiro</li>  
 <li><i class="fa fa-arrow-circle-o-right m-r-xs"></i> Thymeleaf</li>  
 <li><i class="fa fa-arrow-circle-o-right m-r-xs"></i> LayUi</li>  
 </ul>  
 </div>  
 </div>  
 <div class="col-sm-5" id="loginForm">  
 <form id="signupForm" action="login" method="post" onsubmit="**return** checkNull()">  
 <h4 class="no-margins">登录：</h4>  
 <span th:text="${message}" style="color: red"></span>  
 <p class="m-t-md"></p>  
 <input type="text" name="username" id="username" class="form-control uname" placeholder="用户名" value=""/>  
 <input type="password" name="password" id="password" class="form-control pword" placeholder="密码" value=""/>  
 <input type="submit" value="登录" class="btn btn-success btn-block" id="btnSubmit"  
 data-loading="正在验证登录，请稍后..."/>  
 </form>  
 </div>  
 </div>  
 <div class="signup-footer">  
 <div class="pull-left">  
 &copy; 2020 All Rights Reserved. AAA Software <br>  
 <a href="https://www.toutiao.com/c/user/59325412265/#mid=1589186466669581" target="\_blank" rel="nofollow">关注我的头条</a><br>  
 </div>  
 </div>  
</div>  
<script>  
 //校验用户名和密码不能为空  
 **function** checkNull() {  
 **var** username = $("#username").val();  
 **var** password = $("#password").val();  
 **if** (username == **null** || username == '') {  
 alert("用户名不能为空");  
 **return false**;  
 }  
 **if** (password == **null** || password == '') {  
 alert("密码不能为空");  
 **return false**;  
 }  
 **return true**;  
 }  
</script>  
  
</body>  
</html>

1. 菜单管理



步骤一：根据json字符串封装java对象（gsonformat插件）

LayUiTree

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

步骤二：查询所有的菜单，并且组装成layuitree对象

|  |
| --- |
| *\*/* 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){  LayUiTree tree = *fromMenuToTree*(menu);  //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 menu = (Menu) it.next();  if (menu.getParentId() == t.getId())  {  LayUiTree tree = *fromMenuToTree*(menu);  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;  }  }   */\*\*  \* 将menu对象转换成tree对象  \** ***@param*** *menu  \** ***@return*** *\*/* private static LayUiTree fromMenuToTree(Menu menu){  //构造tree对象  LayUiTree tree= new LayUiTree();  tree.setId(menu.getMenuId());  tree.setTitle(menu.getMenuName());  tree.setChecked(false);  tree.setUrl(menu.getUrl());  return tree;  }  } |

1. 用户管理



UserController

@Controller  
public class UserController {  
 @Autowired  
 private UserService userServiceImpl;  
  
 @RequestMapping("/toShowUser")  
 public String toShowUserLayui() {  
 return "user/showUser";  
 }  
  
 @RequestMapping("/showUser")  
 @ResponseBody  
 public LayUITable showUser(int page, int limit) {  
 //开始查询  
 PageInfo<User> pageInfo = userServiceImpl.selectAllUser(page, limit);  
 LayUITable layUiTable = new LayUITable();  
 layUiTable.setCode(0);  
 layUiTable.setMsg("返回消息");  
 //设置分页之后的返回值  
 layUiTable.setCount(pageInfo.getTotal());  
 layUiTable.setData(pageInfo.getList());  
 return layUiTable;  
 }  
  
 @RequestMapping("/saveUser")  
 @ResponseBody  
 public Object saveUser(User userInfo){  
 int i = userServiceImpl.insertSelective(userInfo);  
 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;  
 }  
  
 @RequestMapping("/edit")  
 @ResponseBody  
 public Object editUser(User user){  
 System.*out*.println(user);  
 int i = userServiceImpl.updateByPrimaryKeySelective(user);  
 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*);  
 }  
 System.*out*.println(map);  
 return map;  
 }  
  
 @RequestMapping("/delUser")  
 @ResponseBody  
 public Object delUser( @RequestParam(value = "ids") String ids){  
 //将json字符串转换成list对象  
 List<String> list= (List<String>) JSON.*parse*(ids);  
 System.*out*.println(list);  
 int i = userServiceImpl.delUserByID(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;  
 }  
  
 @RequestMapping("/del")  
 @ResponseBody  
 public Object delUser( @RequestParam(value = "userId") Integer userId){  
 System.*out*.println(userId);  
 int i = userServiceImpl.deleteByPrimaryKey(userId);  
 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;  
 }  
  
}

User表getset

public class User {  
 private Integer userId;  
 private Integer deptId;  
 private String loginName;  
 private String userName;   
 private String email;  
 private String phonenumber;  
 private String sex;  
 private String avatar;  
 private String password;  
 private String salt;  
 private String status;  
 private String delFlag;  
 private String createBy;  
 private Date createTime;  
 private String updateBy;  
 private Date updateTime;  
 private String remark;  
 public Integer getUserId() {  
 return userId;  
 }  
  
 public void setUserId(Integer userId) {  
 this.userId = userId;  
 }  
  
 public Integer getDeptId() {  
 return deptId;  
 }  
  
 public void setDeptId(Integer deptId) {  
 this.deptId = deptId;  
 }  
  
 public String getLoginName() {  
 return loginName;  
 }  
  
 public void setLoginName(String loginName) {  
 this.loginName = loginName == null ? null : loginName.trim();  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public void setUserName(String userName) {  
 this.userName = userName == null ? null : userName.trim();  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email == null ? null : email.trim();  
 }  
  
 public String getPhonenumber() {  
 return phonenumber;  
 }  
  
 public void setPhonenumber(String phonenumber) {  
 this.phonenumber = phonenumber == null ? null : phonenumber.trim();  
 }  
  
 public String getSex() {  
 return sex;  
 }  
  
 public void setSex(String sex) {  
 this.sex = sex == null ? null : sex.trim();  
 }  
  
 public String getAvatar() {  
 return avatar;  
 }  
  
 public void setAvatar(String avatar) {  
 this.avatar = avatar == null ? null : avatar.trim();  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password == null ? null : password.trim();  
 }  
  
 public String getSalt() {  
 return salt;  
 }  
  
 public void setSalt(String salt) {  
 this.salt = salt == null ? null : salt.trim();  
 }  
  
 public String getStatus() {  
 return status;  
 }  
  
 public void setStatus(String status) {  
 this.status = status == null ? null : status.trim();  
 }  
  
 public String getDelFlag() {  
 return delFlag;  
 }  
  
 public void setDelFlag(String delFlag) {  
 this.delFlag = delFlag == null ? null : delFlag.trim();  
 }  
  
 public String getCreateBy() {  
 return createBy;  
 }  
  
 public void setCreateBy(String createBy) {  
 this.createBy = createBy == null ? null : createBy.trim();  
 }  
  
 public Date getCreateTime() {  
 return createTime;  
 }  
  
 public void setCreateTime(Date createTime) {  
 this.createTime = createTime;  
 }  
  
 public String getUpdateBy() {  
 return updateBy;  
 }  
  
 public void setUpdateBy(String updateBy) {  
 this.updateBy = updateBy == null ? null : updateBy.trim();  
 }  
  
 public Date getUpdateTime() {  
 return updateTime;  
 }  
  
 public void setUpdateTime(Date updateTime) {  
 this.updateTime = updateTime;  
 }  
  
 public String getRemark() {  
 return remark;  
 }  
  
 public void setRemark(String remark) {  
 this.remark = remark == null ? null : remark.trim();  
 }  
}

1. 角色管理

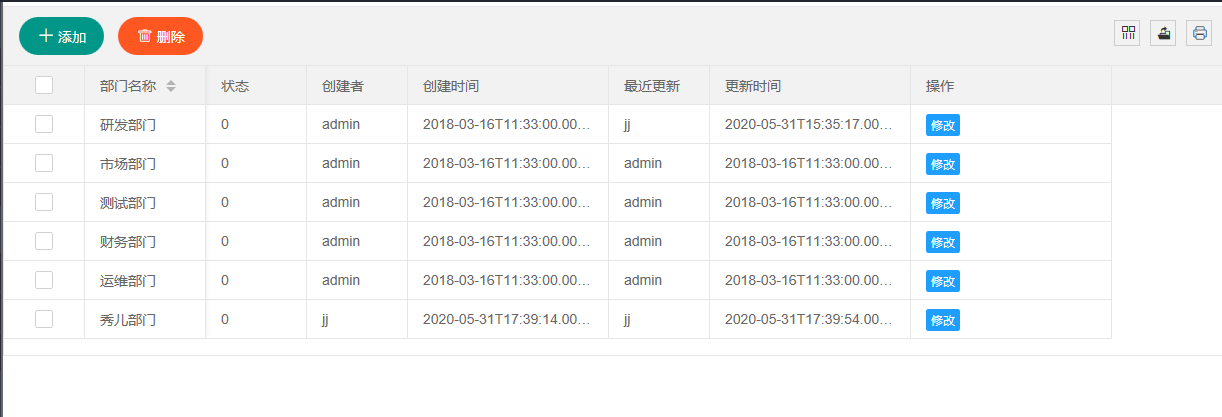


@Data  
public class Role {  
 private Integer roleId;  
  
 private String roleName;  
  
 private String roleKey;  
  
 private Integer roleSort;  
  
 private String status;  
  
 private String delFlag;  
  
 private String createBy;  
 @JsonFormat(timezone = "GMT+8",pattern = "yyyy-MM-dd HH:mm:ss")  
 private Date createTime;  
  
 private String updateBy;  
  
 @JsonFormat(timezone = "GMT+8",pattern = "yyyy-MM-dd HH:mm:ss")  
 private Date updateTime;  
  
 private String remark;  
  
 public Integer getRoleId() {  
 return roleId;  
 }  
  
 public void setRoleId(Integer roleId) {  
 this.roleId = roleId;  
 }  
  
 public String getRoleName() {  
 return roleName;  
 }  
  
 public void setRoleName(String roleName) {  
 this.roleName = roleName == null ? null : roleName.trim();  
 }  
  
 public String getRoleKey() {  
 return roleKey;  
 }  
  
 public void setRoleKey(String roleKey) {  
 this.roleKey = roleKey == null ? null : roleKey.trim();  
 }  
  
 public Integer getRoleSort() {  
 return roleSort;  
 }  
  
 public void setRoleSort(Integer roleSort) {  
 this.roleSort = roleSort;  
 }  
  
 public String getStatus() {  
 return status;  
 }  
  
 public void setStatus(String status) {  
 this.status = status == null ? null : status.trim();  
 }  
  
 public String getDelFlag() {  
 return delFlag;  
 }  
  
 public void setDelFlag(String delFlag) {  
 this.delFlag = delFlag == null ? null : delFlag.trim();  
 }  
  
 public String getCreateBy() {  
 return createBy;  
 }  
  
 public void setCreateBy(String createBy) {  
 this.createBy = createBy == null ? null : createBy.trim();  
 }  
  
 public Date getCreateTime() {  
 return createTime;  
 }  
  
 public void setCreateTime(Date createTime) {  
 this.createTime = createTime;  
 }  
  
 public String getUpdateBy() {  
 return updateBy;  
 }  
  
 public void setUpdateBy(String updateBy) {  
 this.updateBy = updateBy == null ? null : updateBy.trim();  
 }  
  
 public Date getUpdateTime() {  
 return updateTime;  
 }  
  
 public void setUpdateTime(Date updateTime) {  
 this.updateTime = updateTime;  
 }  
  
 public String getRemark() {  
 return remark;  
 }  
  
 public void setRemark(String remark) {  
 this.remark = remark == null ? null : remark.trim();  
 }

role角色控制层

@Controller  
@RequestMapping("/role")  
public class RoleController {  
 @Autowired  
 RoleService roleServiceImpl;  
  
 @RequestMapping("/toShowRole")  
 public String toShowRole(){  
 return "role/showRole";  
 }  
 @RequestMapping("/showRole")  
 @ResponseBody  
 public LayUITable showRole(){  
 List<Role> roleList = roleServiceImpl.selectAllRole();  
 LayUITable layUITable=new LayUITable();  
 layUITable.setCode(0);  
 layUITable.setMsg("返回消息");  
 layUITable.setData(roleList);  
 return layUITable;  
 }  
 @RequestMapping("/saveRole")  
 @ResponseBody  
 public Object saveRole(String roleName,String roleKey,Integer roleSort,String status,String remark){  
  
 User user = (User) SecurityUtils.*getSubject*().getPrincipal();  
 Role role=new Role();  
 role.setRoleName(roleName);  
 role.setRoleKey(roleKey);  
 role.setRoleSort(roleSort);  
 role.setRemark(remark);  
 role.setCreateBy(user.getLoginName());  
 role.setCreateTime(new Date());  
 if (status==null)  
 status = "0";  
 role.setStatus(status);  
 user=null;  
 int i = roleServiceImpl.insertSelective(role);  
 Map map = new LinkedHashMap();  
 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;  
  
 }  
 @RequestMapping("/delRole")  
 @ResponseBody  
 public Object delRole(@RequestParam(value = "ids")String ids){  
 List<Integer> list = (List<Integer>) JSON.*parse*(ids);  
 int i = roleServiceImpl.deleteById(list);  
 Map map = new LinkedHashMap();  
 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("/editRole")  
 @ResponseBody  
 public Object editRole(int roleId,String roleName,String roleKey,Integer roleSort,String status,String remark){  
  
 User user = (User) SecurityUtils.*getSubject*().getPrincipal();  
 Role role=new Role();  
 role.setRoleId(roleId);  
 role.setRoleName(roleName);  
 role.setRoleKey(roleKey);  
 role.setRoleSort(roleSort);  
 role.setRemark(remark);  
 role.setUpdateBy(user.getLoginName());  
 role.setUpdateTime(new Date());  
 if (status==null)  
 status = "0";  
 role.setStatus(status);  
 user=null;  
 int i = roleServiceImpl.updateByPrimaryKeySelective(role);  
 Map map = new LinkedHashMap();  
 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;  
  
 }  
}

1. 部门管理



Dept

public class Dept {  
 private Integer deptId;  
  
 private String deptName;  
  
 private Integer orderNum;  
  
 private String status;  
  
 private String delFlag;  
  
 private String createBy;  
  
 private Date createTime;  
  
 private String updateBy;  
  
 private Date updateTime;  
  
 public Integer getDeptId() {  
 return deptId;  
 }  
  
 public void setDeptId(Integer deptId) {  
 this.deptId = deptId;  
 }  
  
 public String getDeptName() {  
 return deptName;  
 }  
  
 public void setDeptName(String deptName) {  
 this.deptName = deptName == null ? null : deptName.trim();  
 }  
  
 public Integer getOrderNum() {  
 return orderNum;  
 }  
  
 public void setOrderNum(Integer orderNum) {  
 this.orderNum = orderNum;  
 }  
  
 public String getStatus() {  
 return status;  
 }  
  
 public void setStatus(String status) {  
 this.status = status == null ? null : status.trim();  
 }  
  
 public String getDelFlag() {  
 return delFlag;  
 }  
  
 public void setDelFlag(String delFlag) {  
 this.delFlag = delFlag == null ? null : delFlag.trim();  
 }  
  
 public String getCreateBy() {  
 return createBy;  
 }  
  
 public void setCreateBy(String createBy) {  
 this.createBy = createBy == null ? null : createBy.trim();  
 }  
  
 public Date getCreateTime() {  
 return createTime;  
 }  
  
 public void setCreateTime(Date createTime) {  
 this.createTime = createTime;  
 }  
  
 public String getUpdateBy() {  
 return updateBy;  
 }  
  
 public void setUpdateBy(String updateBy) {  
 this.updateBy = updateBy == null ? null : updateBy.trim();  
 }  
  
 public Date getUpdateTime() {  
 return updateTime;  
 }  
  
 public void setUpdateTime(Date updateTime) {  
 this.updateTime = updateTime;  
 }  
}

Dept部门控制层

@Controller  
@RequestMapping("/dept")  
public class DeptController {  
 @Autowired  
 DeptService deptService;  
  
 @RequestMapping("/toShowDept")  
 public String toShowDept(){  
 return "dept/showDept";  
 }  
  
 @RequestMapping("/showDept")  
 @ResponseBody  
 public LayUITable showDept(){  
 List<Dept> DeptList = deptService.selectAllDept();  
 LayUITable layUITable=new LayUITable();  
 layUITable.setCode(0);  
 layUITable.setMsg("返回消息");  
 layUITable.setData(DeptList);  
 return layUITable;  
 }  
  
 @RequestMapping("/saveDept")  
 @ResponseBody  
 public Object savedept(Integer deptId,String deptName,Integer orderNum,String status,String deptFlag){  
  
 User user = (User) SecurityUtils.*getSubject*().getPrincipal();  
 Dept dept=new Dept();  
 dept.setDeptId(deptId);  
 dept.setDeptName(deptName);  
 dept.setOrderNum(orderNum);  
 dept.setStatus(status);  
 dept.setDelFlag(deptFlag);  
 dept.setCreateBy(user.getLoginName());  
 dept.setCreateTime(new Date());  
 if (status==null)  
 status = "0";  
 dept.setStatus(status);  
 user=null;  
 int i = deptService.insertSelective(dept);  
 Map map = new LinkedHashMap();  
 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;  
  
 }  
 @RequestMapping("/delDept")  
 @ResponseBody  
 public Object delRole(@RequestParam(value = "ids")String ids){  
 List<Integer> list = (List<Integer>) JSON.*parse*(ids);  
 int i = deptService.deleteById(list);  
 Map map = new LinkedHashMap();  
 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("/editDept")  
 @ResponseBody  
 public Object editdept(Integer deptId,String deptName,Integer orderNum,String status,String deptFlag){  
  
 User user = (User) SecurityUtils.*getSubject*().getPrincipal();  
 Dept dept=new Dept();  
 dept.setDeptId(deptId);  
 dept.setDeptName(deptName);  
 dept.setOrderNum(orderNum);  
 dept.setStatus(status);  
 dept.setDelFlag(deptFlag);  
 dept.setUpdateBy(user.getLoginName());  
 dept.setUpdateTime(new Date());  
 if (status==null)  
 status = "0";  
 dept.setStatus(status);  
 user=null;  
 int i = deptService.updateByPrimaryKeySelective(dept);  
 Map map = new LinkedHashMap();  
 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;  
  
 }  
}