# 需求规格说明书

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。

## 3、功能分析

1. 登录功能
   1. 登录页面



* 1. 异常登录提示错误信息









1. 用户管理
   1. 查询用户



使用逻辑删除（关键数据），即当del\_flag置为0则该用户可登陆系统；若为1，该用户数据库中的数据仍在，但无法使用该系统。

* 1. 添加用户，设置角色





校验登录用户名是否存在

* 1. 修改用户，禁用用户



* 1. 删除用户



2.5重置用户密码



2.6查询功能

2.6.1根据电话号码查询



2.6.1根据登录名查询



1. 部门管理
   1. 查询部门



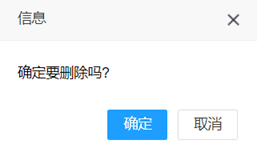
* 1. 添加部门



* 1. 修改部门



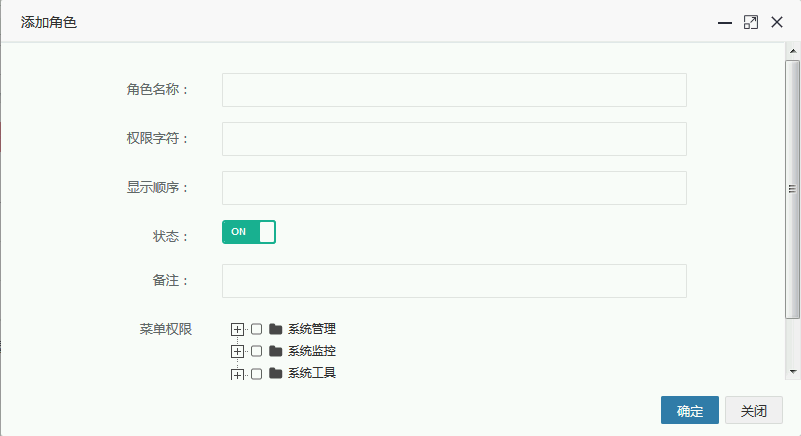
* 1. 删除部门



1. 角色管理
   1. 查询角色



* 1. 添加角色

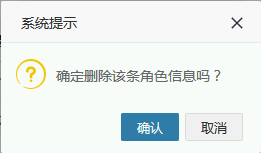


* 1. 修改角色

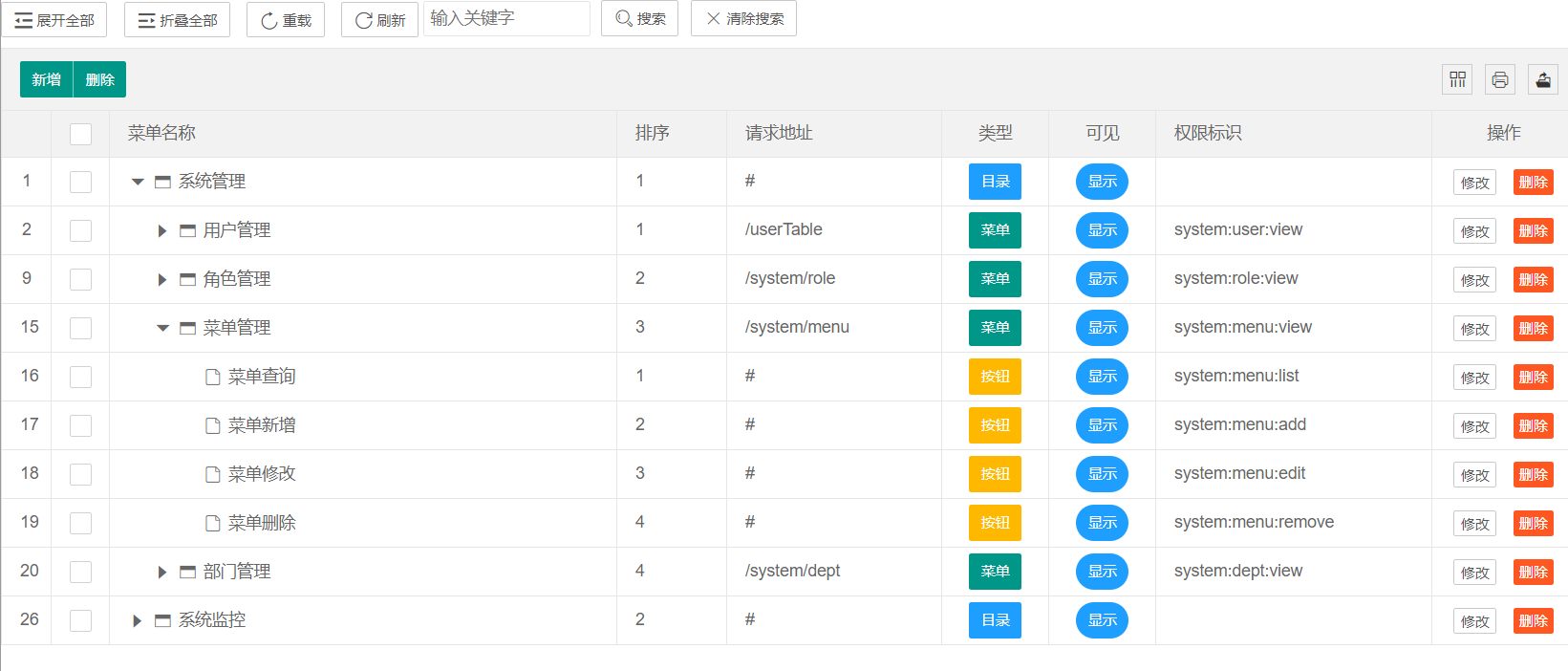




* 1. 删除角色



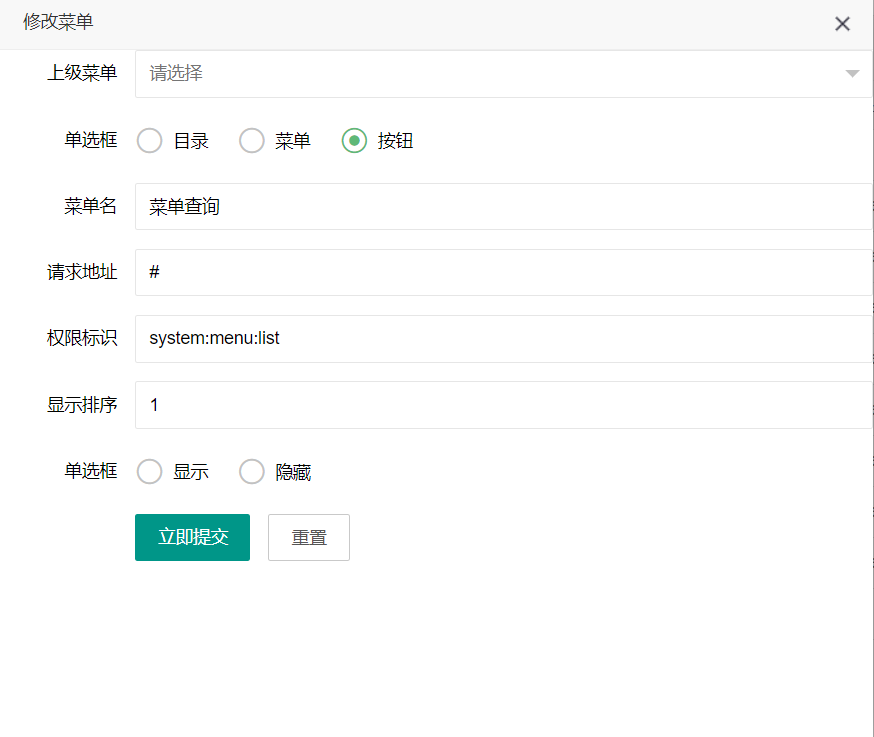
1. 菜单管理
   1. 查询菜单



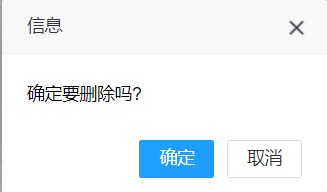
* 1. 添加菜单



* 1. 修改菜单



* 1. 删除菜单



5.5 搜索菜单

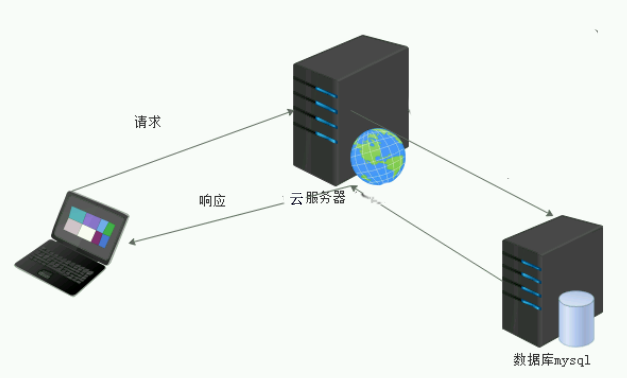


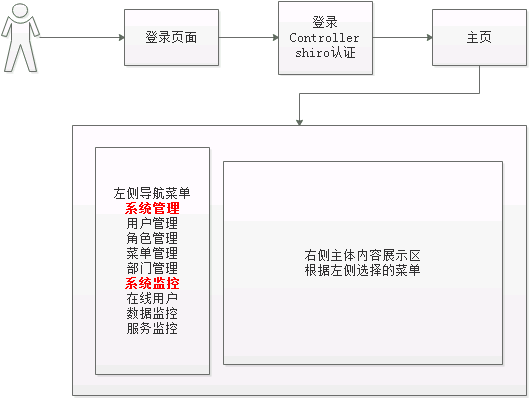
## 4、原型设计

产品经理设计产品原形

## 5、架构设计

### 5.1、整体架构设计





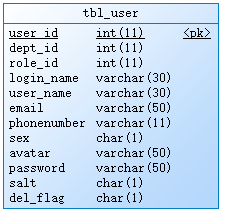
### 5.2、数据模型设计

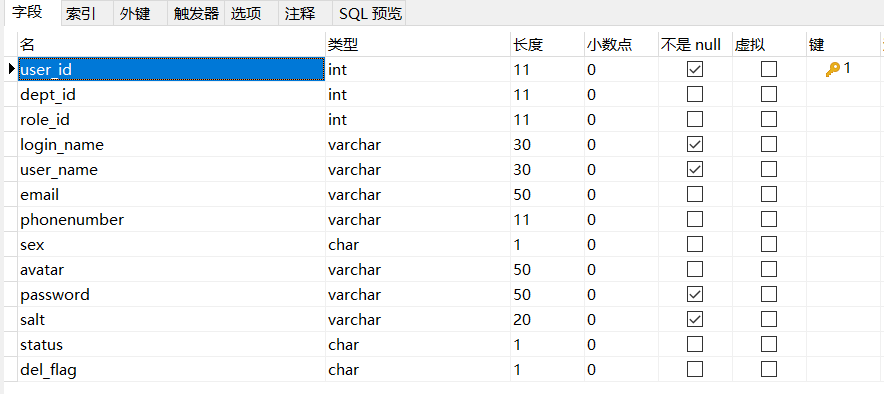
##### 模型 management09的卡片

|  |  |
| --- | --- |
| 名称 | management09 |
| 代码 | management09 |
| 注释 |  |
| DBMS | MySQL 5.0 |
| 数据库 | <None> |

#### 5.2.1、用户表

#### 5.2.1.1用户表设计：





#### 5.2.1.2表的清单

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

#### 5.2.1.3表栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| user\_id | user\_id |
| dept\_id | dept\_id |
| role\_id | role\_id |
| login\_name | login\_name |
| user\_name | user\_name |
| email | email |
| phonenumber | phonenumber |
| sex | sex |
| avatar | avatar |
| password | password |
| salt | salt |
| del\_flag | del\_flag |

#### 5.2.1.4表键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 表格 |
| Key\_1 | Key\_1 | tbl\_user |

##### 5.2.1.5表tbl\_user的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_user |
| 代码 | tbl\_user |
| DBMS | MySQL 5.0 |

##### 5.2.1.6表的代码预览tbl\_user

drop table if exists tbl\_user;

/\*==============================================================\*/

/\* Table: tbl\_user \*/

/\*==============================================================\*/

create table tbl\_user

(

user\_id int(11) not null,

dept\_id int(11),

role\_id int(11),

login\_name varchar(30) not null,

user\_name varchar(30) not null,

email varchar(50),

phonenumber varchar(11),

sex char(1),

avatar varchar(50),

password varchar(50) not null,

salt char(1) not null,

del\_flag char(1),

primary key (user\_id)

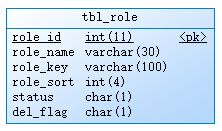
);

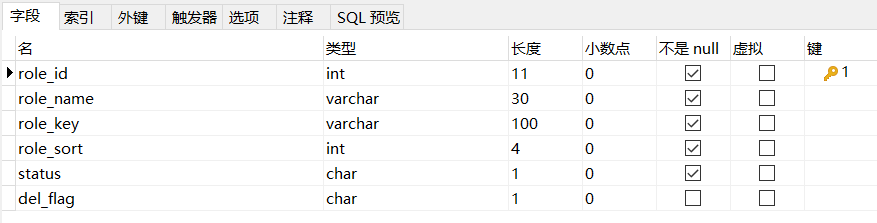
##### 5.2.1.7表tbl\_user的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| user\_id | user\_id |
| dept\_id | dept\_id |
| role\_id | role\_id |
| login\_name | login\_name |
| user\_name | user\_name |
| email | email |
| phonenumber | phonenumber |
| sex | sex |
| avatar | avatar |

#### 5.2.2、角色表

#### 5.2.2.1角色表设计：





#### 5.2.2.2：表的清单

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

#### 5.2.2.3表栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| role\_id | role\_id |
| role\_name | role\_name |
| role\_key | role\_key |
| role\_sort | role\_sort |
| status | status |
| del\_flag | del\_flag |

#### 5.2.2.4表键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 表格 |
| Key\_1 | Key\_1 | tbl\_role |

##### 5.2.2.5表的代码预览tbl\_role

drop table if exists tbl\_role;

/\*==============================================================\*/

/\* Table: tbl\_role \*/

/\*==============================================================\*/

create table tbl\_role

(

role\_id int(11) not null,

role\_name varchar(30) not null,

role\_key varchar(100) not null,

role\_sort int(4) not null,

status char(1) not null,

del\_flag char(1),

primary key (role\_id)

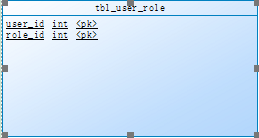
);

##### 5.2.2.6表tbl\_role的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| role\_id | role\_id |
| role\_name | role\_name |
| role\_key | role\_key |
| role\_sort | role\_sort |
| status | status |
| del\_flag | del\_flag |

#### 5.2.3、用户角色表（中间表）

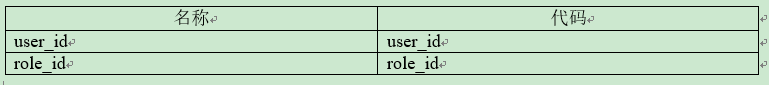
##### 5.2.3.1图tbl\_user\_role



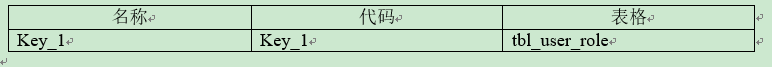
##### 5.2.3.2表的清单



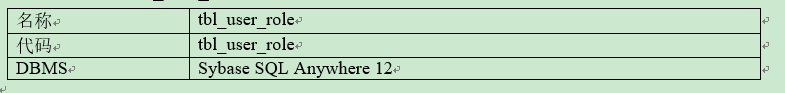
##### 5.2.3.3表栏的清单



##### 5.2.3.4表键的清单

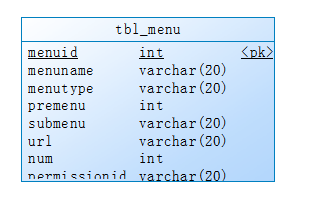


##### 5.2.3.5表的卡片



#### 5.2.4、菜单表

##### 5.2.4.1图table\_menu



##### 5.2.4.2表的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| tbl\_menu | tbl\_menu |

##### 5.2.4.3表栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| menuid | menuid |
| menuname | menuname |
| menutype | menutype |
| premenu | premenu |
| submenu | submenu |
| url | url |
| num | num |
| permissionid | permissionid |
| menuico | menuico |
| state | state |

##### 5.2.4.4表键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 表格 |
| Key\_1 | Key\_1 | tbl\_menu |

##### 5.2.4.5表tbl\_menu的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_menu |
| 代码 | tbl\_menu |
| DBMS | MySQL 5.0 |

5.2.4.5表的代码预览tbl\_menu

drop table if exists tbl\_menu;

/\*==============================================================\*/

/\* Table: tbl\_menu \*/

/\*==============================================================\*/

create table tbl\_menu

(

menuid int not null auto\_increment,

menuname varchar(20) not null,

menutype varchar(20),

premenu int,

submenu varchar(20),

url varchar(20),

num int not null,

permissionid varchar(20),

menuico varchar(100),

state varchar(20),

primary key (menuid)

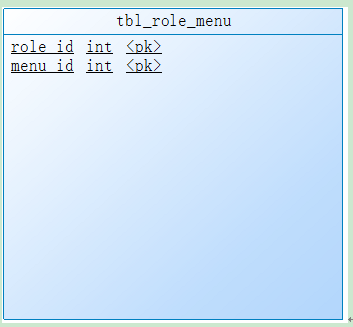
);

##### 5.2.4.6表tbl\_menu的栏的清单

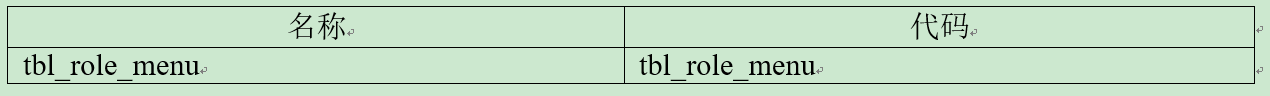
|  |  |
| --- | --- |
| 名称 | 代码 |
| menuid | menuid |
| menuname | menuname |
| menutype | menutype |
| premenu | premenu |
| submenu | submenu |
| url | url |
| num | num |
| permissionid | permissionid |
| menuico | menuico |
| state | state |

#### 5.2.5、角色菜单表

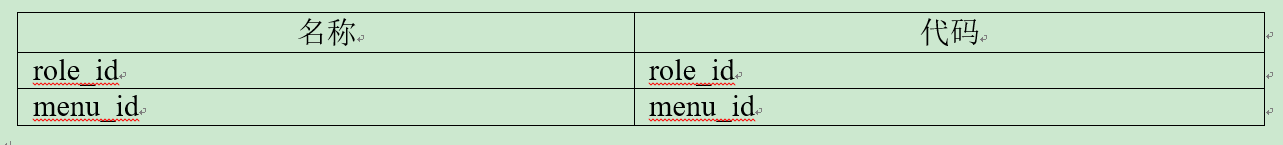
##### 5.2.5.1图tbl\_role\_menu



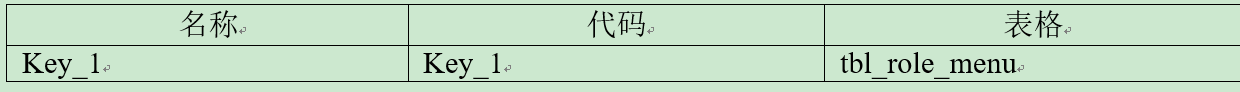
##### 5.2.5.2表的清单



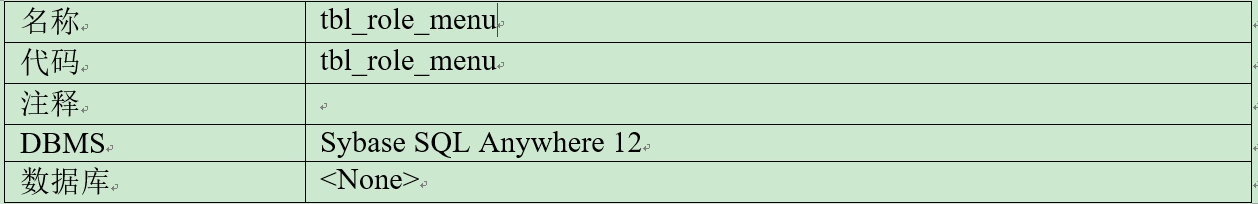
##### 5.2.5.3表栏的清单



##### 5.2.5.4表键的清单



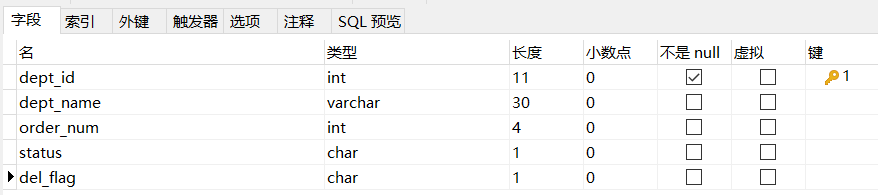
##### 5.2.5.5表的卡片



#### 5.2.6、部门表

#### 5.2.6.1部门表设计：

##### 



##### 5.2.6.2表的清单

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

#### 5.2.6.3表栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| dept\_id | dept\_id |
| dept\_name | dept\_name |
| order\_num | order\_num |
| status | status |
| del\_flag | del\_flag |

#### 5.2.6.4表键的清单

|  |  |  |
| --- | --- | --- |
| 名称 | 代码 | 表格 |
| Key\_1 | Key\_1 | tbl\_dept |

##### 5.2.6.5表tbl\_dept的卡片

|  |  |
| --- | --- |
| 名称 | tbl\_dept |
| 代码 | tbl\_dept |
| DBMS | MySQL 5.0 |

##### 5.2.6.8表的代码预览tbl\_dept

drop table if exists tbl\_dept;

/\*==============================================================\*/

/\* Table: tbl\_dept \*/

/\*==============================================================\*/

create table tbl\_dept

(

dept\_id int(11) not null,

dept\_name char(30),

order\_num int(4),

status char(1),

del\_flag char(1),

primary key (dept\_id)

);

alter table tbl\_dept comment '部门';

##### 5.2.6.9表tbl\_dept的栏的清单

|  |  |
| --- | --- |
| 名称 | 代码 |
| dept\_id | dept\_id |
| dept\_name | dept\_name |
| order\_num | order\_num |
| status | status |
| del\_flag | del\_flag |

## 代码开发

### 登录功能

#### 登录页面

#### 

图6.6.1.1 登录页面部分设计代码

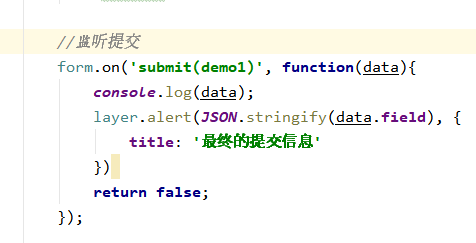


图6.6.1.2 登录页面部分JS代码

图6.6.1.2的JS代码为监听提交

#### 异常登录校验



图6.6.2.1 登录校验

### 用户管理

#### 查询用户

*//获取用户表格信息的请求，返回的是分页的表格数据* @RequestMapping(**"/userTableInfo"**)  
 @ResponseBody  
 **public** LayUITable showUserLayui(**int** page, **int** limit){  
PageInfo<TableInfo> pageInfo = **userService**.selectUserTableInfos(page,limit);  
 LayuiTableUtil layuiTableUtil = **new** LayuiTableUtil(0, **"返回消息"**, pageInfo.getTotal(), pageInfo.getList());  
 **return** layuiTableUtil.getLayUITable();}

#### 添加用户，设置角色

*//添加用户请求*@RequestMapping(**"/insertUser"**)  
@ResponseBody  
**public** Object insert(String loginName, String userName, String password, String roleName, String deptName,String status, String sex, String phoneNumber, String email){  
 Map map = **new** HashMap();  
 MyUserInfo user = **userService**.selectAllUserByName(loginName);  
 String msg;  
 **if** (user != **null**){  
 msg = **"插入失败，该用户名已存在！"**;  
 map.put(**"message"**,msg);  
 map.put(**"code"**,0);  
 **return** map;  
 }  
 **char** statusChar = **'0'**;  
 **if** (status.equalsIgnoreCase(**"off"**)){  
 statusChar = **'1'**;  
 }  
 **char** sexChar = sex.charAt(0);  
 MyUserInfo realUser = **new** MyUserInfo();  
 *//根据角色名查询角色* RoleInfo role = **roleService**.selectAllRoleByName(roleName);  
 DeptInfo dept = **deptService**.selectDeptByName(deptName);  
 String salt = loginName;  
 *//给用户属性赋值* realUser.setLoginName(loginName);  
 realUser.setUserName(userName);  
 realUser.setPassword(password);  
 realUser.setPhoneNumber(phoneNumber);  
 realUser.setEmail(email);  
 realUser.setSex(sexChar);  
 realUser.setStatus(statusChar);  
 realUser.setSalt(salt);  
 realUser.setRoleId(role.getRoleId());  
 realUser.setDeptId(dept.getDeptId());  
 realUser.setDelFlag(**'0'**);  
 *//System.out.println(realUser);* UserRoleInfo userRoleInfo = **new** UserRoleInfo();  
 *//System.out.println(realUser.getSalt());  
 //插入用户信息进用户表* **userService**.insertSelective(realUser);  
 *//接着查询该用户的userId* **int** realUserId = **userService**.selectIdByName(loginName);  
 *//给userRoleInfo对象赋值* userRoleInfo.setRoleId(role.getRoleId());  
 userRoleInfo.setUserId(realUserId);  
 **userRoleService**.insertUserRole(userRoleInfo);  
 msg = **"插入成功"**;  
 map.put(**"message"**,msg);  
 map.put(**"code"**,1);  
 **return** map;  
}

#### 修改用户，禁用用户

@RequestMapping(**"/updateUser"**)  
@ResponseBody  
**public** Object updateUser(Integer userId, String loginName, String userName, String password, String roleName, String deptName,String status, String sex, String phoneNumber, String email){  
  
 Map map = **new** HashMap();  
 MyUserInfo userWithId = **userService**.selectUserById(userId);  
 MyUserInfo userWithName = **userService**.selectAllUserByName(loginName);  
 **if** (userWithId == **null**){  
 map.put(**"message"**,**"修改失败，该用户ID已不存在！"**);  
 map.put(**"code"**,0);  
 **return** map;  
 }  
 **if** (userWithName != **null** && userWithName.getUserId() != userId){  
 map.put(**"message"**,**"修改失败，该用户名已存在！"**);  
 map.put(**"code"**,0);  
 **return** map;  
 }  
 userWithId.setUserName(userName);  
 userWithId.setLoginName(loginName);  
 userWithId.setPhoneNumber(phoneNumber);  
 userWithId.setEmail(email);  
 **char** statusChar;  
 **if** (status.equalsIgnoreCase(**"on"**)){  
 statusChar = **'0'**;  
 }  
 **else** {  
 statusChar = **'1'**;  
 }  
 **char** sexChar = sex.charAt(0);  
 userWithId.setSex(sexChar);  
 userWithId.setStatus(statusChar);  
 RoleInfo role = **roleService**.selectAllRoleByName(roleName);  
 **if** (role == **null**){  
 Integer fakeRoleId = ***NULL***;  
 userWithId.setRoleId(fakeRoleId);  
 }  
 **else** {  
 userWithId.setRoleId(role.getRoleId());  
 }  
 DeptInfo dept = **deptService**.selectDeptByName(deptName);  
 **if** (dept == **null**){  
 Integer fakeDeptId = ***NULL***;  
 userWithId.setRoleId(fakeDeptId);  
 }  
 **else** {  
 userWithId.setDeptId(dept.getDeptId());  
 }  
 **int** a = **userService**.updateByPrimaryKeySelective(userWithId);  
 UserRoleInfo userRoleInfo = **new** UserRoleInfo();  
 userRoleInfo.setUserId(userWithId.getUserId());  
 userRoleInfo.setRoleId(userWithId.getRoleId());  
 **int** b =**userRoleService**.updateUserRole(userRoleInfo);  
 **if** (a <= 0 || b <= 0){  
 map.put(**"message"**,**"数据库异常，修改失败！"**);  
 map.put(**"code"**,0);  
 **return** map;  
 }  
 map.put(**"message"**,**"修改成功！"**);  
 map.put(**"code"**,1);  
 **return** map;  
}

#### 删除用户

*//用户表格行工具栏中删除按钮点击事件的请求*@RequestMapping(**"/toolDelete"**)  
@ResponseBody  
**public** String toolDelete(Integer userId){  
 MyUserInfo user = **userService**.selectUserById(userId);  
 **if** (user == **null**){  
 **return "删除失败，该用户ID不存在！"**;  
 }  
 *//userService.deleteById(id);* **userService**.updateDelFlagById(userId);  
 **return "删除成功！"**;  
}

*//用户表头删除按钮点击事件的请求*@RequestMapping(**"/toolbarDelete"**)  
@ResponseBody  
**public** String toolbarDelete(@RequestParam(value = **"list"**)String list){  
 List<String> ids = (List<String>) JSON.*parse*(list);  
 *//int i = userService.deleteByIds(ids);* **int** i = **userService**.updateDelFlagByIds(ids);  
 **if** (i <= 0){  
 **return "删除失败"**;  
 }  
 **return "删除成功！"**;  
}

### 部门管理

**8.1编写controller中部门管理增删改查的方法**

@Controller  
@RequestMapping("/system")  
public class DeptController {  
 @Autowired  
 private DeptService deptService;

//返回一个路径，通过该路径调用页面文件  
 @RequestMapping("/dept")  
 public String toShowDept(){  
 return "dept/showDept";  
 }  
 //动态加载部门下拉框数据的请求，用于用户添加时获取部门信息  
 @RequestMapping("/deptSelect")  
 @ResponseBody  
 public Object selectAllDept(){  
 return deptService.selectAllDept();  
 }

//查询，获取说有部门信息  
 @RequestMapping("/showDept")  
 @ResponseBody  
 public Object showDept() {  
 //开始查询  
 List<DeptInfo> deptList = deptService.selectAllDept();  
 LayUITable layUiTable = new LayUITable();  
 layUiTable.setCode(0);  
 layUiTable.setMsg("返回消息");  
 layUiTable.setData(deptList);  
 return layUiTable;  
 }

//保存，在数据库添加获取的部门信息  
 @RequestMapping("/saveDept")  
 @ResponseBody  
 public Object saveDept(DeptInfo dept){  
 int i = deptService.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;  
 }

//编辑，编辑数据库中已有的信息@RequestMapping("/editDept")  
 @ResponseBody  
 public Object editDept(DeptInfo dept){  
 int i = deptService.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;  
 }

//删除，删除数据库中已有的信息  
 @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 = deptService.delDeptByID(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;  
 }  
}

**8.2部门管理页面增删改查接口代码：**

<script>  
 **var** form=layui.form;  
 **var** table=layui.table;  
 form.render();//渲染表单  
 //第一个实例  
 **var** myTable= table.render({  
 elem: '#demo'  
 ,height: 350  
 ,url: 'showDept' //数据接口  
 ,toolbar: '#mytoolbar' //开启工具栏，此处显示默认图标，可以自定义模板，详见文档  
 ,cols: [  
 [ //表头  
 {field: 'deptId', title: '部门编号', width:80, sort: **true**, fixed: 'left',checkbox: **true**}  
 ,{field: 'deptName', title: '部门名称', width:120, sort: **true**, fixed: 'left'}  
 ,{field: 'orderNum', title: '显示顺序', width:180}  
 ,{field: 'status', title: '状态', width:50}  
 /\*,{field: 'createTime', title: '创建时间', width:200}\*/  
 ,{ title: '操作', width:200, toolbar: '#barDemo'}  
 ]]  
 });  
 //监听头工具栏事件  
 table.on('toolbar(test)', **function**(obj){  
 **var** checkStatus = table.checkStatus(obj.config.id)  
 ,data = checkStatus.data; //获取选中的数据  
 **var** ids=[];  
 **switch**(obj.event){  
 **case** 'saveDept':  
 layer.open(  
 {  
 type: 1,  
 content: $("#DeptSaveForm"),  
 icon: 1,  
 title:"添加部门",  
 area: ['450px', '300px']  
 }  
 )  
 **break**;  
 **case** 'update':  
 **if**(data.length === 0){  
 layer.msg('请选择一行');  
 } **else if**(data.length > 1){  
 layer.msg('只能同时编辑一个');  
 } **else** {  
 layer.alert('编辑 [id]：'+ checkStatus.data[0].id);  
 }  
 **break**;  
 **case** 'delDept':  
 **if**(data.length === 0){  
 layer.msg('请选择一行');  
 } **else** {  
 console.log(data)  
 layer.confirm('确定要删除吗?', **function**(index){  
 //获取所有需要删除的userid，保存在ids的数组中  
 **for**(**var** i=0;i<data.length;i++){  
 ids.push(data[i].deptId);  
 }  
 $.ajax({  
 url: 'delDept',//地址  
 dataType: 'json',//数据类型  
 type: 'POST',//类型  
 data:{ids:JSON.stringify(ids)},//将数组ids转换成json字符串  
 //请求成功  
 success: **function** (result) {  
 **if**(result.code==0){  
 //关闭弹出层  
 layer.close(index);  
 //显示删除成功消息  
 layer.msg(result.message,{icon:1})  
 //重新加载数据表格  
 myTable.reload();  
 }**else** {  
 //关闭弹出层  
 layer.close(index);  
 layer.msg(result.message,{icon:5})  
 }  
 }  
 });  
 });  
 }  
 **break**;  
 };  
 });  
 //监听提交  
 form.on('submit(saveDeptForm)', **function**(data){  
 //数据的提交，ajax方式  
 console.log(data.field)  
 $.ajax({  
 url: 'saveDept',//地址  
 dataType: 'json',//数据类型  
 type: 'POST',//类型  
 data:data.field,  
 //请求成功  
 success: **function** (result) {  
 **if**(result.code==0){  
 //关闭弹出层  
 layer.closeAll();  
 //显示添加成功消息  
 layer.msg(result.message,{icon:1})  
 //重新加载数据表格  
 myTable.reload();  
 }**else** {  
 //关闭弹出层  
 layer.closeAll();  
 layer.msg(result.message,{icon:5})  
 }  
 }  
 });  
 **return false**;  
 });  
 form.on('submit(editDeptForm)', **function**(data){  
 //数据的提交，ajax方式  
 console.log(data.field)  
 $.ajax({  
 url: 'editDept',//地址  
 dataType: 'json',//数据类型  
 type: 'POST',//类型  
 data:data.field,  
 //请求成功  
 success: **function** (result) {  
 **if**(result.code==0){  
 //关闭弹出层  
 layer.closeAll();  
 //显示添加成功消息  
 layer.msg(result.message,{icon:1})  
 //重新加载数据表格  
 myTable.reload();  
 }**else** {  
 //关闭弹出层  
 layer.closeAll();  
 layer.msg(result.message,{icon:5})  
 }  
 }  
 });  
 **return false**;  
 });  
 //监听行工具事件  
 table.on('tool(test)', **function**(obj){ //注：tool 是工具条事件名，test 是 table 原始容器的属性 lay-filter="对应的值"  
 **var** data = obj.data //获得当前行数据  
 ,layEvent = obj.event; //获得 lay-event 对应的值  
 **if**(layEvent === 'detail'){  
 layer.msg('查看操作');   
 } **else if**(layEvent === 'del'){  
 layer.confirm('真的删除行么', **function**(index){  
 obj.del(); //删除对应行（tr）的DOM结构  
 layer.close(index);  
 //向服务端发送删除指令  
 });  
 } **else if**(layEvent === 'edit'){  
 layer.msg('编辑操作');  
 //填充修改用户的表单项  
 console.log(data);  
 $("#deptName").val(data.deptName);  
 $("#deptId").val(data.deptId);  
 layer.open(  
 {  
 type: 1,  
 content: $("#DeptEditForm"),  
 icon: 1,  
 title:"修改部门",  
 area: ['450px', '200px']  
 }  
 )  
 }  
 });  
</script>

### 角色管理

#### 查询角色

@RequestMapping(**"/roleTableInfo"**)  
@ResponseBody  
**public** LayUITable roleTableInfo(){  
 List<RoleInfo> roles = **roleService**.selectAllRole();  
 LayuiTableUtil layuiTableUtil = **new** LayuiTableUtil(0,**"角色表格返回消息"**, roles.size(),roles);  
 **return** layuiTableUtil.getLayUITable();  
}

#### 添加角色

@RequestMapping(**"/insertRole"**)  
 @ResponseBody  
 **public** Object insertRole(String roleName, String roleKey, Integer sort, String status){  
 RoleInfo role = **roleService**.selectAllRoleByName(roleName);  
*// System.out.println(status);* Map map = **new** HashMap();  
 **if** (role != **null**){  
 map.put(**"code"**,0);  
 map.put(**"message"**,**"添加失败，该角色已存在！"**);  
 **return** map;  
 }  
 **char** statusChar;  
 **if** (!status.equals(**"on"**)){  
 statusChar = **'1'**;  
 }  
 **else**{  
 statusChar = **'0'**;  
 }

#### 修改角色

@RequestMapping(**"/updateRole"**)  
 @ResponseBody  
 **public** Object updateRole(Integer roleId, String roleName, String roleKey, Integer roleSort, String status){  
 Map map = **new** HashMap();  
 RoleInfo role = **roleService**.selectRoleById(roleId);  
 **if** (role == **null**){  
 map.put(**"code"**,0);  
 map.put(**"message"**,**"编辑失败，该角色已失效！"**);  
 **return** map;  
 }  
 RoleInfo roleWithName = **roleService**.selectAllRoleByName(roleName);  
 **if** (roleWithName != **null** && roleWithName.getRoleId() != roleId){  
 map.put(**"code"**,0);  
 map.put(**"message"**,**"编辑失败，该角色存在！"**);  
 **return** map;  
 }  
 **char** statusChar;  
 **if** (!status.equals(**"on"**)){  
 statusChar = **'1'**;  
 }  
 **else**{  
 statusChar = **'0'**;  
 }  
*// RoleInfo roleWithId = new RoleInfo();* role.setRoleId(roleId);  
 role.setStatus(statusChar);  
 role.setRoleName(roleName);  
 role.setRoleSort(roleSort);  
 role.setRoleKey(roleKey);  
 **int** i = **roleService**.updateRoleSelective(role);  
 **if** (i <= 0){  
 map.put(**"code"**,0);  
 map.put(**"message"**,**"编辑失败，数据库异常！ "**);  
 **return** map;  
 }  
 map.put(**"code"**,1);  
 map.put(**"message"**,**"编辑成功！"**);  
 **return** map;  
 }

#### 删除角色

*//用户表头删除按钮点击事件的请求*@RequestMapping(**"/toolbarDeleteRole"**)  
@ResponseBody  
**public** String toolbarDelete(@RequestParam(value = **"list"**)String list){  
 List<String> ids = (List<String>) JSON.*parse*(list);  
 *//int i = userService.deleteByIds(ids);* **int** i = **roleService**.updateRoleDelFlagByIds(ids);  
 **if** (i <= 0){  
 **return "删除失败"**;  
 }  
 **return "删除成功！"**;  
}

### 菜单管理

#### 查询菜单

@RequestMapping(**"/showMenu"**)  
@ResponseBody  
**public** LayuitreeTable showMenu() {  
 *//开始查询* List<MenuInfo> menuInfoList = **menuBiz**.selectAllMenu();  
 LayuitreeTable layuitreeTable =**new** LayuitreeTable();  
 layuitreeTable.setCount(**menuBiz**.selectMenuNum());  
 layuitreeTable.setCode(MyConstants.***successCode***);  
 layuitreeTable.setMsg(**"查询成功"**);  
 layuitreeTable.setData(menuInfoList);  
 **return** layuitreeTable;  
}

#### 添加菜单

@RequestMapping(**"/toAddMenu"**)  
@ResponseBody  
**public** Object toAddMenu(MenuInfo menuInfo) {  
  
 Integer i = **menuBiz**.insertSelective(menuInfo);  
 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;  
}

#### 修改菜单

@RequestMapping(**"/toEditMenu"**)  
@ResponseBody  
**public** Object toEditMenu(MenuInfo menuInfo) {  
  
 Integer i = **menuBiz**.updateByPrimaryKeySelective(menuInfo);  
 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;  
}

#### 删除菜单

//单条删除

@RequestMapping(**"/toDelMenu"**)  
@ResponseBody  
**public** Object toDelMenu(MenuInfo menuInfo) {  
  
 Integer i = **menuBiz**.deleteByPrimaryKey(menuInfo.getMenuId());  
 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;  
}

//多条删除  
@RequestMapping(**"/toDelMenus"**)  
@ResponseBody  
**public** Object toDelMenus( @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 = **menuBiz**.delMenusByID(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;  
}