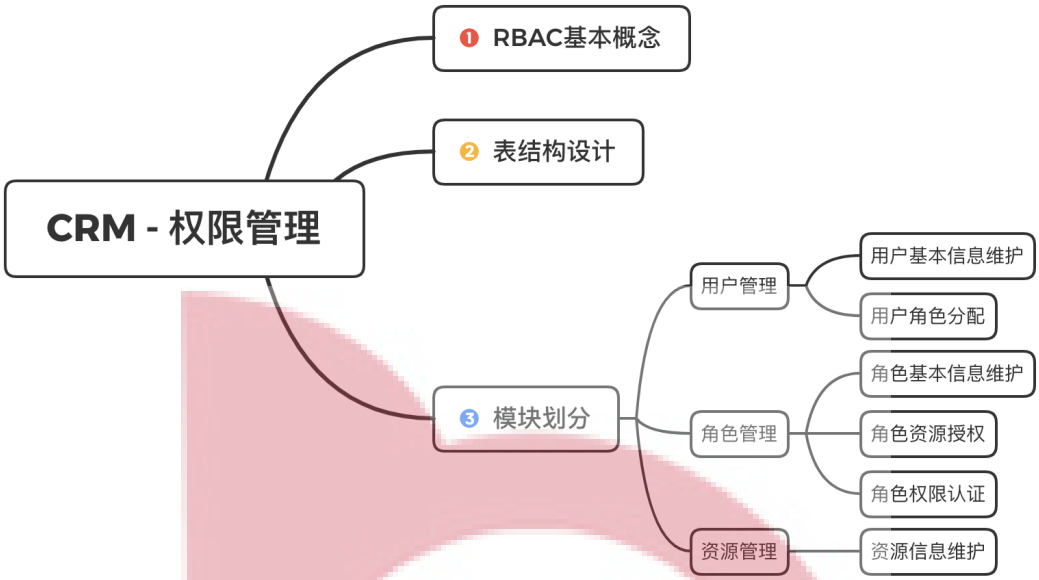
CRM - 权限管理

学习目标



权限管理RBAC基本概念

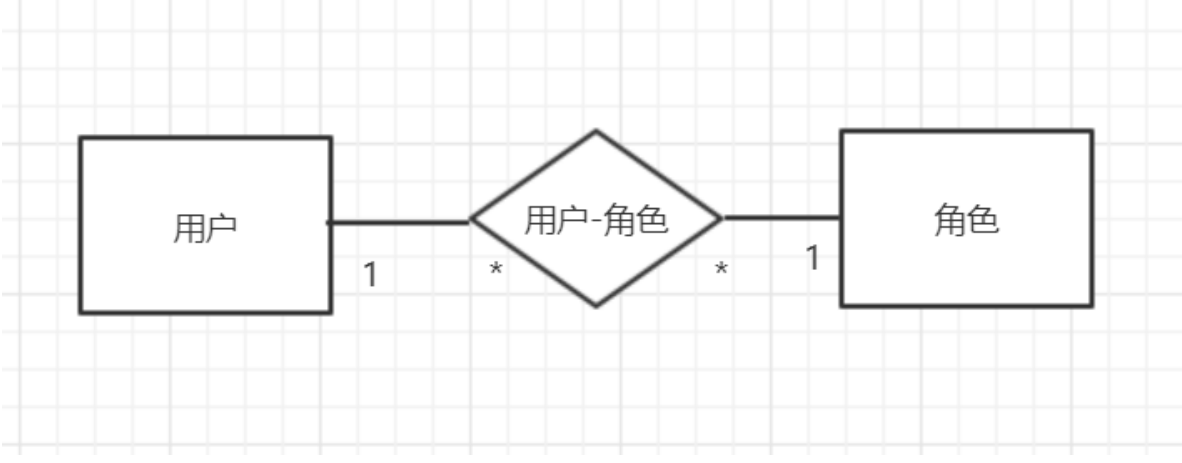
RBAC是基于角色的访问控制（`Role-Based Access Control`）在RBAC中，权限与角色相关联，用户通过扮演适当的角色从而得到这些角色的权限。这样管理都是层级相互依赖的，权限赋予给角色，角色又赋予用户，这样的权限设计很清楚，管理起来很方便。

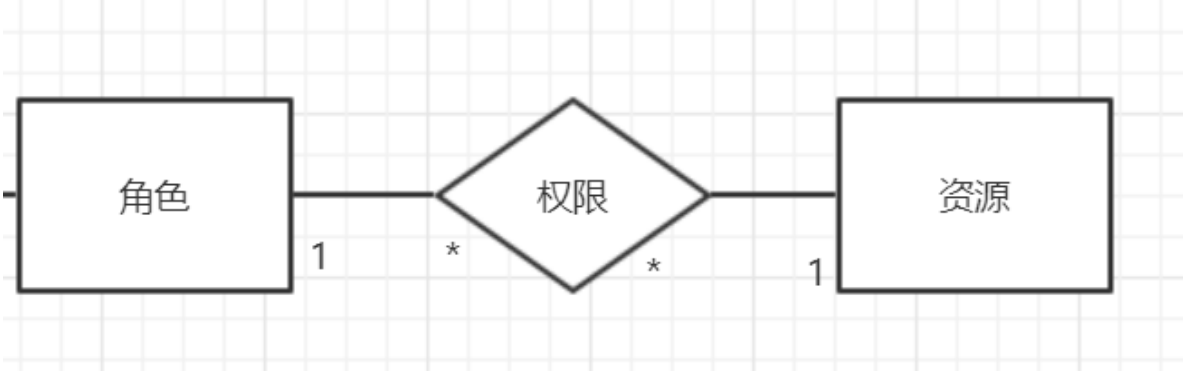
RBAC授权实际上是`Who`、`What`、`How`三元组之间的关系，也就是`Who`对`What`进行`How`的操作，简单说明就是谁对什么资源做了怎样的操作。

RBAC表结构设计

实体对应关系

用户-角色-资源实体间对应关系图分析如下

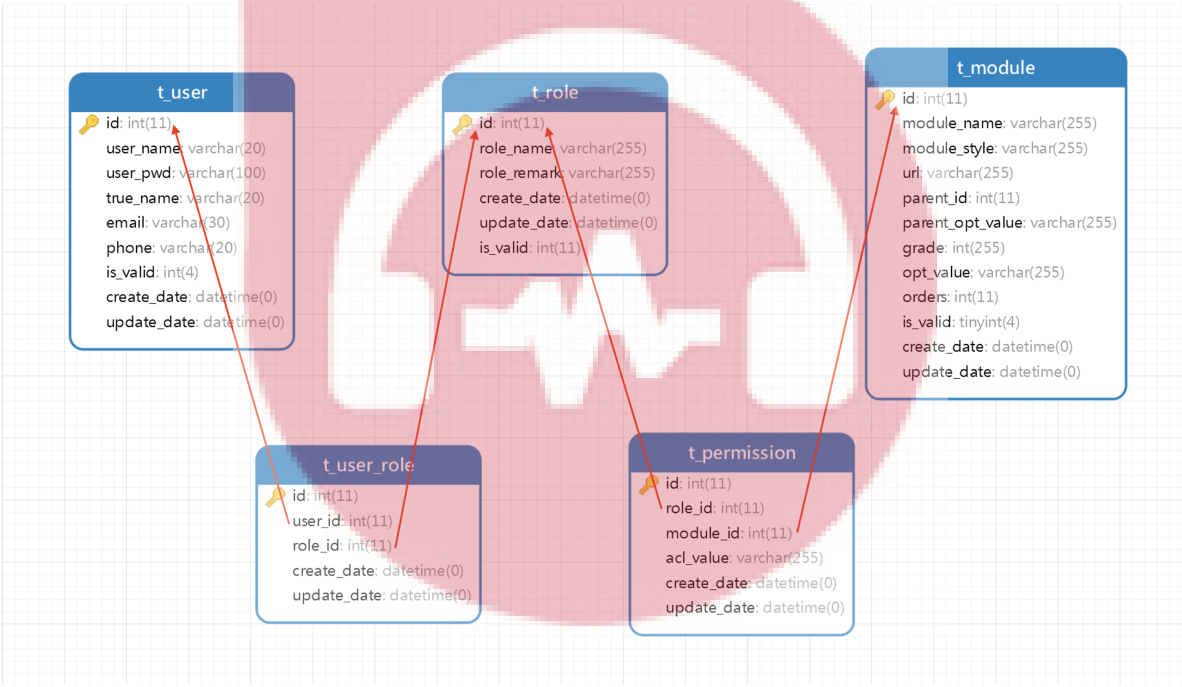




这里用户与角色实体对应关系为多对多，角色与资源对应关系同样为多对多关系，所以在实体设计上用户与角色间增加用户角色实体，将多对多的对应关系拆分为一对多，同理，角色与资源多对多对应关系拆分出中间实体对象权限实体。

表结构设计

从上面实体对应关系分析，权限表设计分为以下基本的五张表结构：用户表(t_user)、角色表(t_role)、t_user_role(用户角色表)、资源表(t_module)、权限表(t_permission)，表结构关系如下:



t_user	用户表			
	字段	字段类型	字段限制	字段描述
主键	id	int(11)	自增	主键id
	user_name	varchar(20)	非空	用户名
	user_pwd	varchar(100)	非空	用户密码
	true_name	varchar(20)	可空	真实姓名
	email	varchar(30)	可空	邮箱
	phone	varchar(20)	可空	电话
	is_valid	int(4)	可空	有效状态
	create_date	datetime	可空	创建时间
	update_date	datetime	可空	更新时间

t_role	角色表			
	字段	字段类型	字段限制	字段描述
主键	id	int(11)	自增	主键id
	role_name	varchar(20)	非空	角色名
	role_remarker	varchar(100)	可空	角色备注
	is_valid	int(4)	可空	有效状态
	create_date	datetime	可空	创建时间
	update_date	datetime	可空	更新时间

t_user_role	用户角色表			
	字段	字段类型	字段限制	字段描述
主键	id	int(11)	自增	主键id
	user_id	int(4)	非空	用户id
	role_id	int(4)	角色id	角色id
	create_date	datetime	可空	创建时间
	update_date	datetime	可空	更新时间

t_module	资源表			
	字段	字段类型	字段限制	字段描述
主键	id	int(11)	自增	资源id
	module_name	varchar(20)	可空	资源名
	module_style	varchar(100)	可空	资源样式
	url	varchar(20)	可空	资源url地址
	parent_id	int(11)	非空	上级资源id
	parent_opt_value	varchar(20)	非空	上级资源权限码
	grade	int(11)	非空	层级
	opt_value	varchar(30)	可空	权限码
	orders	int(11)	非空	排序号
	is_valid	int(4)	可空	有效状态
	create_date	datetime	可空	创建时间
	update_date	datetime	可空	更新时间

t_permission	权限表			
	字段	字段类型	字段限制	字段描述
主键	id	int(11)	自增	主键id
	role_id	int(11)	非空	角色id
	module_id	int(11)	非空	资源id
	acl_value	varchar(20)	非空	权限码
	create_date	datetime	可空	创建时间
	update_date	datetime	可空	更新时间

模块划分

从表结构设计可以看出：这里有三张主表(t_user，t_role，t_module)，功能实现上这里划分为三大模块：

用户管理

- 用户基本信息维护(t_user)
- 角色分配(t_user_role)

角色管理

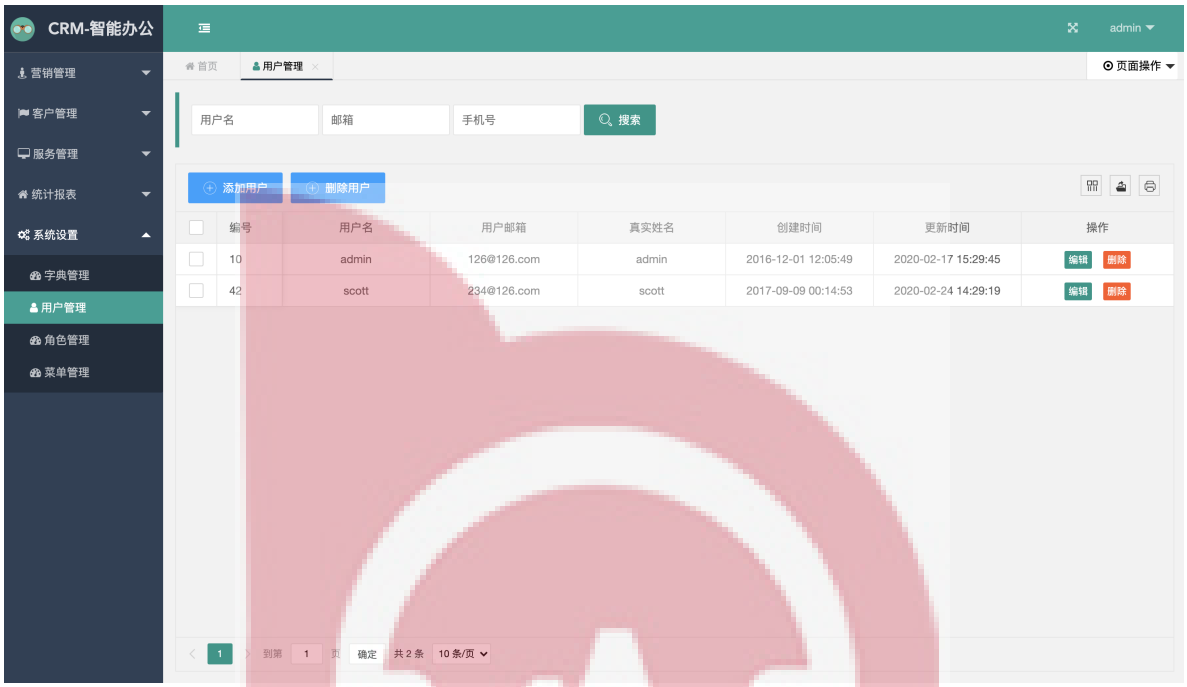
- 角色基本信息维护(t_role)
- 角色授权与认证(t_permission)

- 资源菜单信息维护(t_module)

用户管理功能实现

用户查询操作

页面效果



后端代码实现

查询条件

- UserQuery.java 查询类定义

```
public class UserQuery extends BaseQuery {
    // 用户名
    private String userName;
    // 邮箱
    private String email;
    // 电话
    private String phone;
    /*
      省略get |set 方法
    */
 }
```

设置SQL

- UserMapper.xml

```
<select id="selectByParams" parameterType="com.xxxx.crm.query.UserQuery"
resultType="com.xxxx.crm.vo.User">
```

```
select
    <include refid="Base_Column_List"/>
    from t_user
    <where>
        is_valid=1
        <if test="null !=userName and userName !=''">
            and user_name like concat('%',#{userName},'%')
        </if>
        <if test="null !=phone and phone !=''">
            and phone =#{phone}
        </if>
        <if test="null !=email and email !=''">
            and email =#{email}
        </if>
    </where>
</select>
```

Service

- UserService.java

```
/**
 * 多条件分页查询用户数据
 * @param query
 * @return
 */
public Map<String, Object> queryUserByParams (UserQuery query) {
    Map<String, Object> map = new HashMap<>();
    PageHelper.startPage(query.getPage(), query.getLimit());
    PageInfo<User>  pageInfo = new PageInfo<>(userMapper.selectByParams(query));
    map.put("code",0);
    map.put("msg", "");
    map.put("count", pageInfo.getTotal());
    map.put("data", pageInfo.getList());
    return map;
}
```

Controller

- UserController.java

列表查询方法添加

```
/**
 * 多条件查询用户数据
 * @param userQuery
 * @return
 */
@RequestMapping("user/list")
@ResponseBody
public Map<String, Object> queryUserByParams(UserQuery userQuery) {
    return userService.queryUserByParams(userQuery);
}
```

前端核心代码

- user.ftl

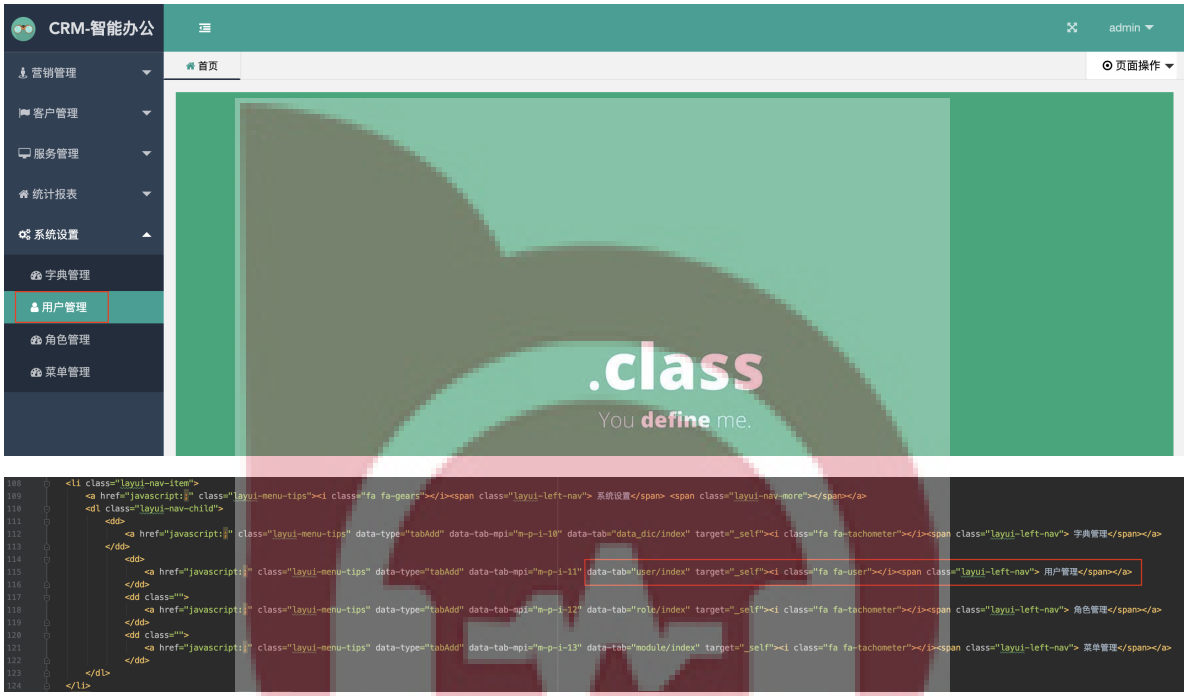
views/user 目录下添加 user.ftl 视图模板

```
<!DOCTYPE html>
<html>
    <head>
        <title>用户管理</title>
        <#include "../common.ftl">
    </head>
    <body class="childrenBody">
        <form class="layui-form" >
            <blockquote class="layui-elem-quote quoteBox">
                <form class="layui-form">
                    <div class="layui-inline">
                        <div class="layui-input-inline">
                            <input type="text" name="userName" class="layui-
input searchVal" placeholder="用户名" />
                        </div>
                        <div class="layui-input-inline">
                            <input type="text" name="email" class="layui-input
searchVal" placeholder="邮箱" />
                        </div>
                        <div class="layui-input-inline">
                            <input type="text" name="phone" class="layui-input
searchVal" placeholder="手机号" />
                        </div>
                        <a class="layui-btn search_btn" data-type="reload">
                            <i class="layui-icon">&#xe615;</i>
                            搜索
                        </a>
                    </div>
                </form>
            </blockquote>

            <table id="userList" class="layui-table"  lay-filter="users">
</table>

            <script type="text/html" id="toolbarDemo">
                <div class="layui-btn-container">
                    <a class="layui-btn layui-btn-normal addNews_btn" lay-
event="add">
                        <i class="layui-icon">&#xe608;</i>
                        添加用户
                    </a>
                    <a class="layui-btn layui-btn-normal delNews_btn" lay-
event="del">
                        <i class="layui-icon">&#xe608;</i>
                        删除用户
                    </a>
                </div>
            </script>

            <!--操作-->
            <script id="userListBar" type="text/html">
```

```
                <a class="layui-btn layui-btn-xs" id="edit" lay-event="edit">编辑
</a>
                <a class="layui-btn layui-btn-xs layui-btn-danger" lay-
event="del">删除</a>
            </script>
        </form>

    <script type="text/javascript" src="${ctx}/js/user/user.js"></script>
    </body>
</html>
```

页面入口

点击左侧的菜单，进入对应的页面



UserController 后台设置对应的接口

```
/**
  * 进入用户页面
  * @return
  */
@RequestMapping("user/index")
public String index(){
  return "user/user";
}
```

核心 JS

- user.js

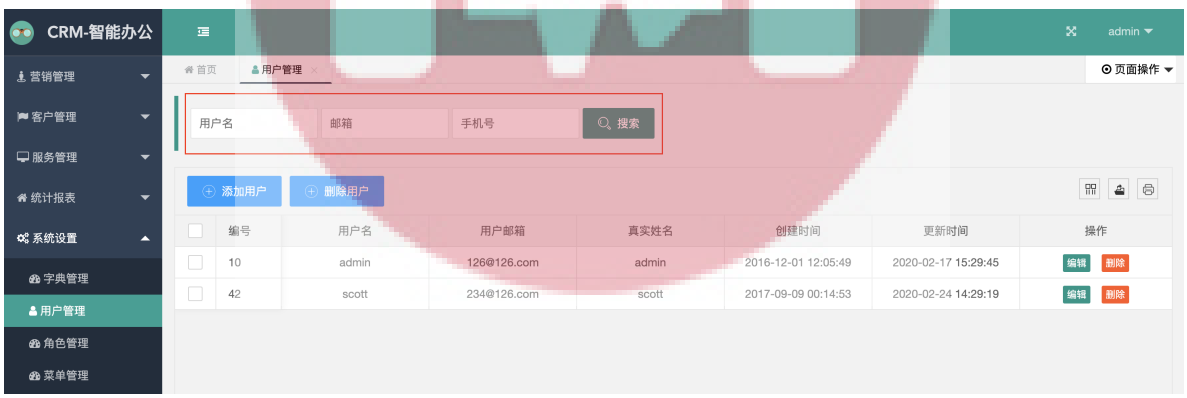
js/user 目录下添加 user.js 文件

```
layui.use(['table','layer'],function(){
        var layer = parent.layer === undefined ? layui.layer : top.layer,
         $ = layui.jquery,
         table = layui.table;
    /**
     * 用户列表展示
```

```
    var  tableIns = table.render({
        elem: '#userList',
        url : ctx+'/user/list',
        cellMinWidth : 95,
        page : true,
        height : "full-125",
        limits : [10,15,20,25],
        limit : 10,
        toolbar: "#toolbarDemo",
        id : "userListTable",
        cols : [[
            {type: "checkbox", fixed:"left", width:50},
            {field: "id", title:'编号',fixed:"true", width:80},
            {field: 'userName', title: '用户名', minWidth:50, align:"center"},
            {field: 'email', title: '用户邮箱', minWidth:100, align:'center'},
            {field: 'phone', title: '用户电话', minWidth:100, align:'center'},
            {field: 'trueName', title: '真实姓名', align:'center'},
            {field: 'createDate', title: '创建时间',
align:'center',minWidth:150},
            {field: 'updateDate', title: '更新时间',
align:'center',minWidth:150},
            {title: '操作', minWidth:150,
templet:'#userListBar',fixed:"right",align:"center"}
        ]]
    });

});
```

多条件查询

添加多条件查询点击事件



- user.ftl

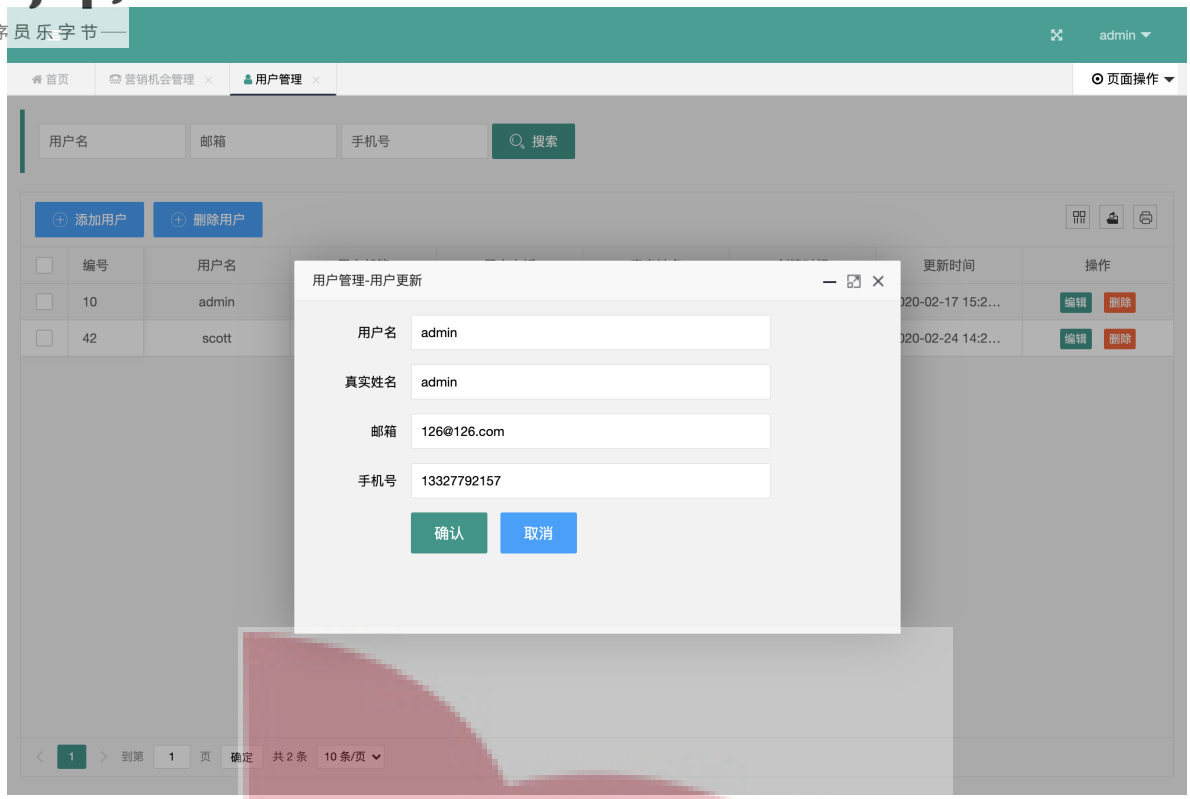
```
<form class="layui-form">
    <div class="layui-inline">
        <div class="layui-input-inline">
            <input type="text" name="userName" class="layui-input searchVal" placeholder="用户名" />
        </div>
        <div class="layui-input-inline">
            <input type="text" name="email" class="layui-input searchVal" placeholder="邮箱" />
        </div>
        <div class="layui-input-inline">
            <input type="text" name="phone" class="layui-input searchVal" placeholder="手机号" />
        </div>
        <a class="layui-btn search_btn" data-type="reload">
            <i class="layui-icon">&#xe615;</i>
            搜索
        </a>
    </div>
</form>
```

index.js 添加搜索点击事件

```
/**
 * 绑定搜索按钮的点击事件
 */
$(".search_btn").on("click",function(){
  table.reload("userListTable",{
    page: {
      curr: 1 //重新从第 1 页开始
    },
    where: {
      userName: $("input[name='userName']").val(),  //用户名
      email: $("input[name='email']").val(),  //邮箱
      phone: $("input[name='phone']").val()  //手机号
    }
  })
});
```

用户添加与更新

页面效果



用户记录添加

实现思路

```
/**
 * 添加用户
 *  1. 参数校验
 *      用户名 非空 唯一性
 *      邮箱   非空
 *      手机号 非空  格式合法
 *  2. 设置默认参数
 *      isValid 1
 *      creteDate   当前时间
 *      updateDate  当前时间
 *      userPwd 123456 -> md5加密
 *  3. 执行添加，判断结果
 */
```

核心代码

- UserService.java

```
/**
 * 添加用户
 * @param user
 */
@Transactional(propagation = Propagation.REQUIRED)
public void saveUser(User user) {
    // 1. 参数校验
    checkParams(user.getUserName(), user.getEmail(), user.getPhone());
    // 2. 设置默认参数
    user.setIsValid(1);
    user.setCreateDate(new Date());
```

```
        user.setUpdateDate(new Date());
        user.setUserPwd(Md5Util.encode("123456"));
        // 3. 执行添加，判断结果
        AssertUtil.isTrue(userMapper.insertSelective(user) == null, "用户添加失败！");
    }

    /**
     * 参数校验
     * @param userName
     * @param email
     * @param phone
     */
    private void checkParams(String userName, String email, String phone) {
        AssertUtil.isTrue(StringUtils.isBlank(userName), "用户名不能为空！");
        // 验证用户名是否存在
        User temp = userMapper.queryUserByUserName(userName);
        AssertUtil.isTrue(null != temp, "该用户已存在！");
        AssertUtil.isTrue(StringUtils.isBlank(email), "请输入邮箱地址！");
        AssertUtil.isTrue(!PhoneUtil.isMobile(phone), "手机号码格式不正确！");
    }
```

- UserController.java

```
    /**
     * 添加用户
     * @param user
     * @return
     */
    @RequestMapping("user/save")
    @ResponseBody
    public ResultInfo saveUser(User user) {
        userService.saveUser(user);
        return success("用户添加成功！");
    }
```

用户记录更新

实现思路

```
    /**
     * 更新用户
     *  1. 参数校验
     *      id  非空  记录必须存在
     *      用户名 非空  唯一性
     *      email 非空
     *      手机号 非空 格式合法
     *  2. 设置默认参数
     *      updateDate
     *  3. 执行更新，判断结果
     * @param user
     */
```

核心代码

- UserService.java

```
     * 更新用户
     * @param user
     */
    @Transactional(propagation = Propagation.REQUIRED)
    public void updateUser(User user) {
        // 1. 参数校验
        // 通过id查询用户对象
        User temp = userMapper.selectByPrimaryKey(user.getId());
        // 判断对象是否存在
        AssertUtil.isTrue(temp == null, "待更新记录不存在！");
        // 验证参数
        checkParams(user.getUserName(),user.getEmail(),user.getPhone());
        // 2. 设置默认参数
        temp.setUpdateDate(new Date());
        // 3. 执行更新，判断结果
        AssertUtil.isTrue(userMapper.updateByPrimaryKeySelective(user) < 1, "用户更新
失败！");
    }
```

- 更新参数校验方法

```
    /**
     * 参数校验
     * @param userName
     * @param email
     * @param phone
     * @param userId
     */
    private void checkParams(String userName, String email, String phone, Integer
userId) {
        AssertUtil.isTrue(StringUtils.isBlank(userName), "用户名不能为空！");
        // 验证用户名是否存在
        User temp = userMapper.queryUserByUserName(userName);
        // 如果是添加操作，数据库是没有数据的，数据库中只要查询到用户记录就表示不可用
        // 如果是修改操作，数据库是有数据的，查询到用户记录就是当前要修改的记录本身就表示可用，否则
不可用
        // 数据存在，且不是当前要修改的用户记录，则表示其他用户占用了该用户名
        AssertUtil.isTrue(null != temp && !(temp.getId().equals(userId)), "该用户已存
在！");
        AssertUtil.isTrue(StringUtils.isBlank(email), "请输入邮箱地址！");
        AssertUtil.isTrue(!PhoneUtil.isMobile(phone), "手机号码格式不正确！");
    }
```

- UserController.java

```
 * 更新用户
 * @param user
 * @return
 */
@RequestMapping("user/update")
@ResponseBody
public ResultInfo updateUser(User user) {
    userService.updateUser(user);
    return success("用户更新成功！");
}
```

工具栏与行监听事件

- user.js

```
/**
 * 头部工具栏事件
 */
table.on("toolbar(users)", function (obj) {
    var checkStatus = table.checkStatus(obj.config.id);
    switch (obj.event) {
        case "add":
            openAddOrUpdateUserDialog();
            break;
    }
});

/**
 * 行监听事件
 */
table.on("tool(users)", function(obj){
    var layEvent = obj.event;
    // 监听编辑事件
    if(layEvent === "edit") {
        openAddOrUpdateUserDialog(obj.data.id);
    }
});


/**
 * 打开用户添加或更新对话框
 */
function openAddOrUpdateUserDialog(userId) {
    var url  =  ctx + "/user/addOrUpdateUserPage";
    var title = "用户管理-用户添加";
    if(userId){
        url = url + "?id="+userId;
        title = "用户管理-用户更新";
    }
    layui.layer.open({
        title : title,
        type : 2,
        area:["650px","400px"],
        maxmin:true,
        content : url
    });
```

视图转发方法

- UserController.java

UserController 添加视图转发方法

```
/**
 * 进入用户添加或更新页面
 * @param id
 * @param model
 * @return
 */
@RequestMapping("user/addOrUpdateUserPage")
public String addUserPage(Integer id, Model model){
    if(null != id){
        model.addAttribute("user",userService.selectByPrimaryKey(id));
    }
    return "user/add_update";
}
```

页面模板

- add_update.ftl

views/user 目录下添加add_update.ftl 模板文件.

```
<!DOCTYPE html>
<html>
    <head>
        <#include "../common.ftl">
    </head>
    <body class="childrenBody">
        <form class="layui-form" style="width:80%;">
            <input name="id" type="hidden" value="${(user.id)!}"/>
            <div class="layui-form-item layui-row layui-col-xs12">
                <label class="layui-form-label">用户名</label>
                <div class="layui-input-block">
                    <input type="text" class="layui-input userName"
                           lay-verify="required" name="userName" id="userName"
 value="${(user.userName)!}" placeholder="请输入用户名">
                </div>
            </div>
            <div class="layui-form-item layui-row layui-col-xs12">
                <label class="layui-form-label">真实姓名</label>
                <div class="layui-input-block">
                    <input type="text" class="layui-input userName"
                           lay-verify="required" name="trueName" id="trueName"
value="${(user.trueName)!}" placeholder="请输入真实姓名">
                </div>
            </div>
            <div class="layui-form-item layui-row layui-col-xs12">
                <label class="layui-form-label">邮箱</label>
                <div class="layui-input-block">
                    <input type="text" class="layui-input userEmail"
```

```
                            lay-verify="email" name="email"
value="${(user.email)!}"
                            id="email"
                            placeholder="请输入邮箱">
                    </div>
                </div>

                <div class="layui-form-item layui-row layui-col-xs12">
                    <label class="layui-form-label">手机号</label>
                    <div class="layui-input-block">
                        <input type="text" class="layui-input userEmail"
                               lay-verify="phone" name="phone"
value="${(user.phone)!}" id="phone" placeholder="请输入手机号">
                    </div>
                </div>


                <br/>
                <div class="layui-form-item layui-row layui-col-xs12">
                    <div class="layui-input-block">
                        <button class="layui-btn layui-btn-lg" lay-submit=""
                                lay-filter="addOrUpdateUser">确认
                        </button>
                        <button class="layui-btn layui-btn-lg layui-btn-normal">取消
</button>
                    </div>
                </div>
            </form>

        <script type="text/javascript" src="${ctx}/js/user/add.update.js"></script>
        </body>
</html>
```

JS 文件

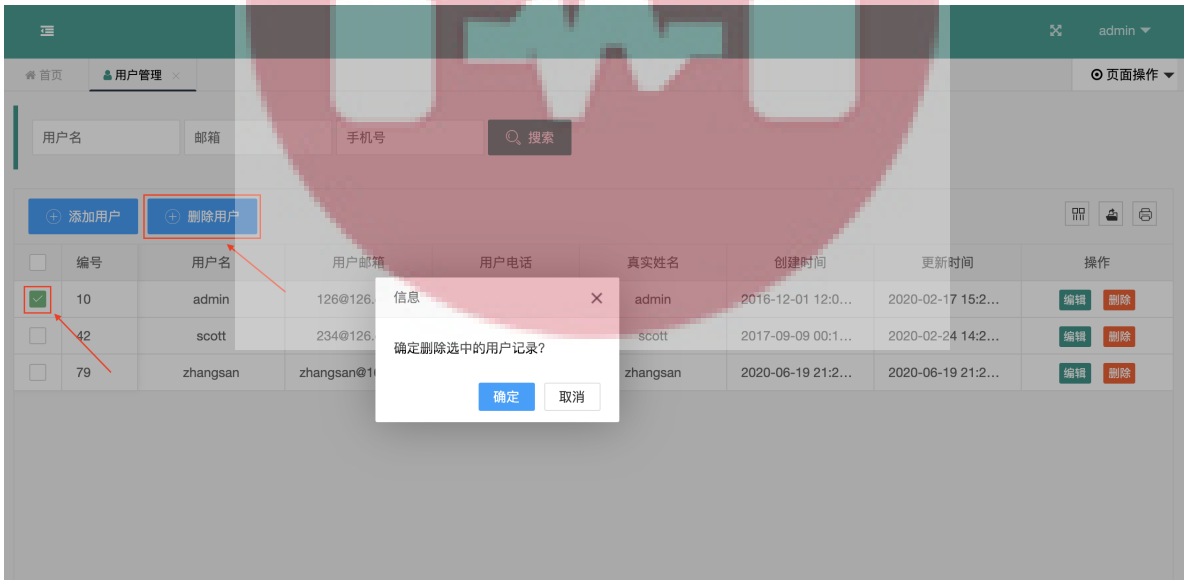
- add.update.js

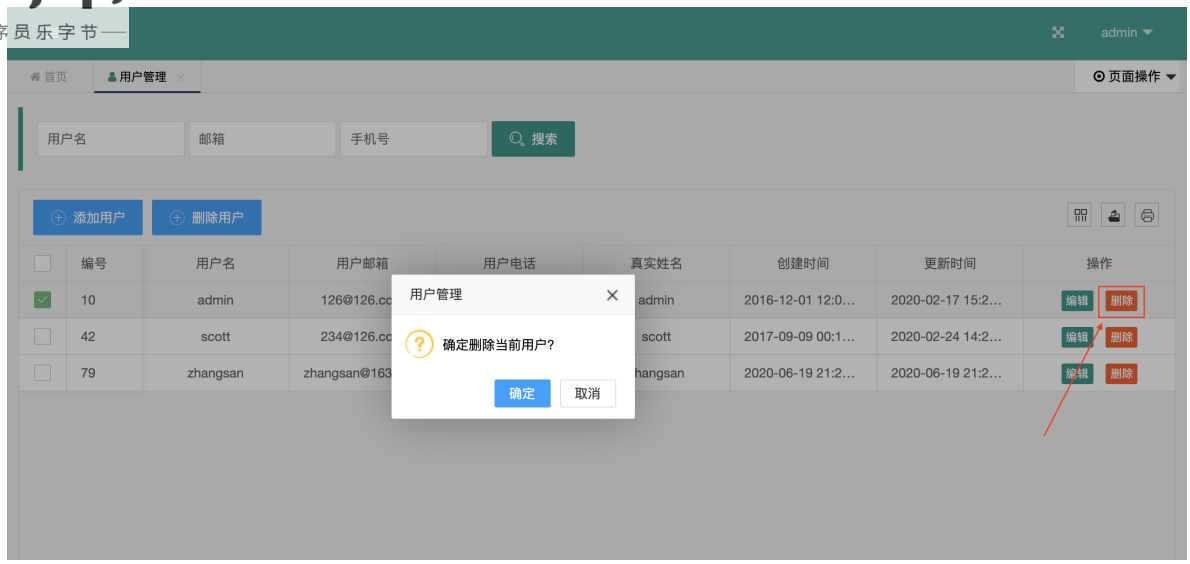
js/user 目录下添加 add.update.js 文件，实现表单数据提交操作

```
layui.use(['form', 'layer'], function () {
    var form = layui.form,
        layer = parent.layer === undefined ? layui.layer : top.layer,
        $ = layui.jquery;

    /**
     * 添加或更新用户
     */
    form.on("submit(addOrUpdateUser)", function (data) {
        // 弹出loading层
        var index = top.layer.msg('数据提交中，请稍候', {icon: 16, time: false,
shade: 0.8});
        var url = ctx + "/user/save";
        if($("input[name='id']").val()){
            url = ctx + "/user/update";
        }
        $.post(url, data.field, function (res) {
            if (res.code == 200) {
```

```
                setTimeout(function () {
                    // 关闭弹出层（返回值为index的弹出层）
                    top.layer.close(index);
                    top.layer.msg("操作成功！");
                    // 关闭所有ifream层
                    layer.closeAll("iframe");
                    // 刷新父页面
                    parent.location.reload();
                }, 500);
            } else {
                layer.msg(res.msg, {icon: 5});
            }
        });
        return false;
    });

    /**
     * 关闭弹出层
     */
    $("#closeBtn").click(function () {
        var index = parent.layer.getFrameIndex(window.name); //先得到当前iframe层
的索引
        parent.layer.close(index); //再执行关闭
    });

});
```

用户删除操作





后端代码实现

设置SQL

- UserMapper.xml

用户数据支持数据批量删除操作，后端接收前端数组参数借助 mybatis 动态标签 foreach 实现记录批量删除。

```
<!-- （批量）删除操作 -->
<update id="deleteBatch">
  update
    t_user
  set
    is_valid = 0
  where
    id
  in
    <foreach collection="array" item="id" open="(" close=")" separator=",">
      #{id}
    </foreach>
</update>
```

Service

- UserService.java

```
/**
 * 删除用户
 * @param ids
 */
@Transactional(propagation = Propagation.REQUIRED)
public void deleteUserByIds(Integer[] ids) {
    AssertUtil.isTrue(null==ids || ids.length == 0,"请选择待删除的用户记录!");
    AssertUtil.isTrue(deleteBatch(ids) != ids.length,"用户记录删除失败!");
}
```

Controller

- UserController.java

```
 * 删除用户
 * @param ids
 * @return
 */
@RequestMapping("delete")
@ResponseBody
public ResultInfo deleteUser(Integer[] ids){
    userService.deleteBatch(ids);
    return success("用户记录删除成功");
}
```

前端核心代码

修改 user.js 文件，添加删除按钮事件

```
/**
 * 头部工具栏事件
 */
table.on("toolbar(users)", function (obj) {
    var checkStatus = table.checkStatus(obj.config.id);
    switch (obj.event) {
        case "add":
            openAddOrUpdateUserDialog();
            break;
        case "del":
            deleteUser(checkStatus.data);
            break;
    }
});

/**
 * 批量删除用户
 * @param datas
 */
function deleteUser(datas) {
    if(datas.length == 0){
        layer.msg("请选择删除记录!", {icon: 5});
        return;
    }
    layer.confirm('确定删除选中的用户记录？', {
        btn: ['确定','取消'] //按钮
    }, function(index){
        layer.close(index);
        var ids= "ids=";
        for(var i=0;i<datas.length;i++){
            if(i<datas.length-1){
                ids=ids+datas[i].id+"&ids=";
            }else {
                ids=ids+datas[i].id
            }
        }
        $.ajax({
            type:"post",
            url:ctx + "/user/delete",
```

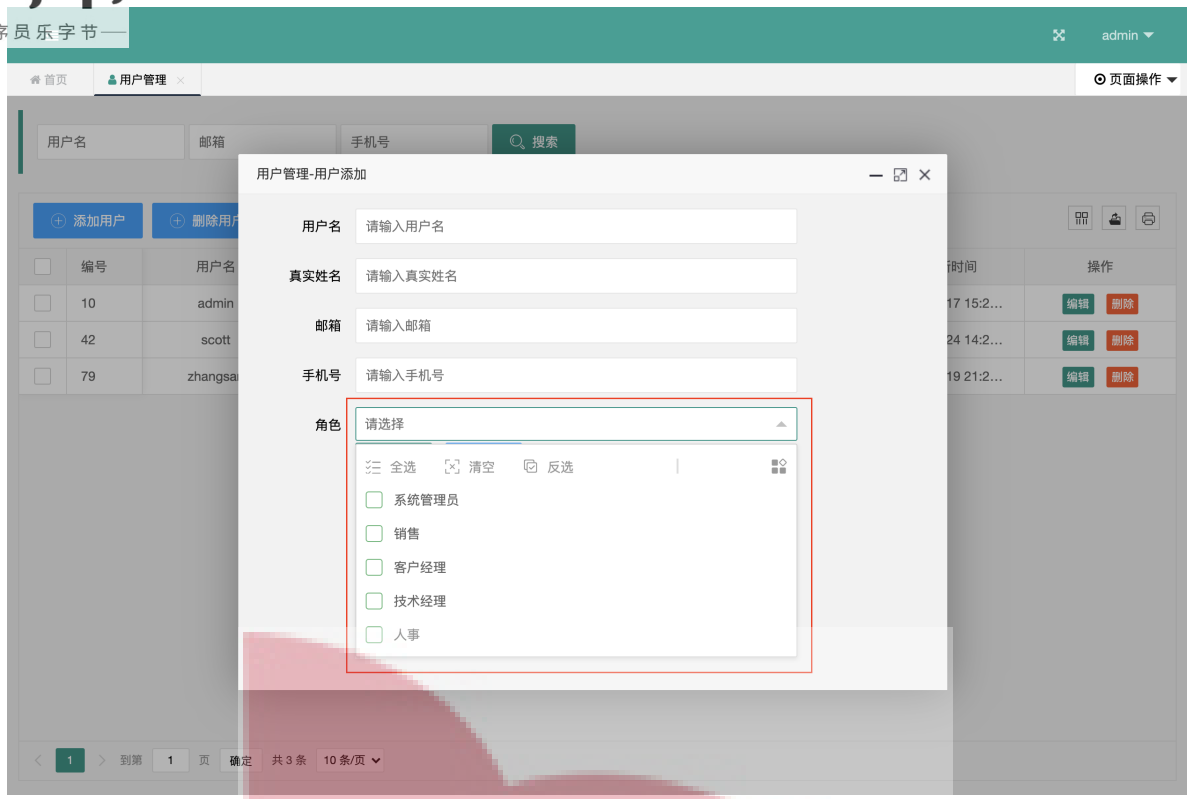
```
            data:ids,
            dataType:"json",
            success:function (data) {
                if(data.code==200){
                    tableIns.reload();
                }else{
                    layer.msg(data.msg, {icon: 5});
                }
            }
        })
    });
}

/**
 * 行监听事件
 */
table.on("tool(users)", function(obj){
    var layEvent = obj.event;
    // 监听编辑事件
    if(layEvent === "edit") {
        openAddOrUpdateUserDialog(obj.data.id);
    } else if(layEvent === "del") {
      // 监听删除事件
      layer.confirm('确定删除当前用户？', {icon: 3, title: "用户管理"}, function
(index) {
        $.post(ctx + "/user/delete",{ids:obj.data.id},function (data) {
          if(data.code==200){
            layer.msg("操作成功！");
            tableIns.reload();
          }else{
            layer.msg(data.msg, {icon: 5});
          }
        });
      });
    }
});
```

用户角色关联

基于角色的权限管理在实现用户基本信息维护的同时，这是还需考虑用户在系统中扮演的角色有哪些，然后通过角色来进一步控制用户能够操作的资源，这里为用户管理角色使用到前端多选下拉框插件 formSelects 详见官网介绍。

用户角色查询

页面效果



后端代码实现

- RoleMapper.xml

添加查询所有角色记录标签

```
<!-- 查询所有的角色 -->
<select id="queryAllRoles" resultType="java.util.Map">
  SELECT
      id, role_name AS roleName
  from
      t_role
  where
      is_valid = 1
</select>
```

- RoleMapper.java

定义查询角色信息的接口

```
public interface RoleMapper extends BaseMapper<Role, Integer> {

    // 查询角色列表
    public List<Map<String,Object>> queryAllRoles();

}
```

- RoleService.java

```
     * 查询角色列表
     * @return
     */
    public List<Map<String, Object>> queryAllRoles(){
        return roleMapper.queryAllRoles();
    }
```

- RoleController.java

提供方法返回所有角色记录

```
/**
 * 查询角色列表
 * @return
 */
@RequestMapping("queryAllRoles")
@ResponseBody
public List<Map<String,Object>> queryAllRoles(){
    return roleService.queryAllRoles();
}
```

前端核心代码

- user/add_update.ftl

在用户的添加和修改页面，设置选择用户角色的下拉框。

```
<div class="layui-form-item layui-row layui-col-xs12">
    <label class="layui-form-label">角色</label>
    <div class="layui-input-block">
        <select name="roleIds"  xm-select="selectId">
        </select>
    </div>
</div>
```

```
                    lay-verify="email" name="email" value="${(user.email)!}"
                    id="email"
                    placeholder="请输入邮箱">
            </div>
        </div>

        <div class="layui-form-item layui-row layui-col-xs12">
            <label class="layui-form-label">手机号</label>
            <div class="layui-input-block">
                <input type="text" class="layui-input userEmail"
                lay-verify="phone" name="phone" value="${(user.phone)!}" id="phone" placeholder="请输入手机号">
            </div>
        </div>

        <div class="magb15 layui-col-md4 layui-col-xs12">
            <label class="layui-form-label">角色</label>
            <div class="layui-input-block">
                <select name="roleIds"  xm-select="selectId">
                </select>
            </div>
        </div>


        <br/>
        <div class="layui-form-item layui-row layui-col-xs12">
            <div class="layui-input-block">
                <button class="layui-btn layui-btn-lg" lay-submit=""
                        lay-filter="addOrUpdateUser">确认
                </button>
                <button class="layui-btn layui-btn-lg layui-btn-normal" id="closeBtn">取消</button>
            </div>
        </div>
    </form>
```

- user/add.update.js

这里要引入 formSelects 模块实现下拉框数据填充操作。

```
layui.use(['form', 'layer','formSelects'], function () {
    var form = layui.form,
        layer = parent.layer === undefined ? layui.layer : top.layer,
        $ = layui.jquery;
          // 引入 formSelects 模块
        formSelects = layui.formSelects;

    /**
     * 加载下拉框数据
     */
    formSelects.config('selectId',{
        type:"post",
        searchUrl:ctx + "/role/queryAllRoles",
        //自定义返回数据中name的key, 默认 name
        keyName: 'roleName',
        //自定义返回数据中value的key, 默认 value
        keyVal: 'id'
    },true);
}
```

用户角色关联

```
@Transactional(propagation = Propagation.REQUIRED)
public void saveUser(User user) {
    //这里用户记录添加代码省略
    /**
         * 用户角色分配
         *    userId
```

```
         *    roleIds
         */
        relaionUserRole(user.getId(), user.getRoleIds());
    }

    @Transactional(propagation = Propagation.REQUIRED)
    public void updateUser(User user) {
        //这里用户记录更新代码省略
        /**
         * 用户角色分配
         *    userId
         *    roleIds
         */
        relaionUserRole(user.getId(), user.getRoleIds());
    }

    private void relaionUserRole(int useId, String roleIds) {
        /**
         * 用户角色分配
         *   原始角色不存在    添加新的角色记录
         *   原始角色存在      添加新的角色记录
         *   原始角色存在      清空所有角色
         *   原始角色存在      移除部分角色
         * 如何进行角色分配???
         *  如果用户原始角色存在   首先清空原始所有角色   添加新的角色记录到用户角色表
         */

        int count = userRoleMapper.countUserRoleByUserId(useId);
        if (count > 0) {
            AssertUtil.isTrue(userRoleMapper.deleteUserRoleByUserId(useId) != count,
"用户角色分配失败!");
        }
        if (StringUtils.isNotBlank(roleIds)) {
            //重新添加新的角色
            List<UserRole> userRoles = new ArrayList<UserRole>();
            for (String s : roleIds.split(",")) {
                UserRole userRole = new UserRole();
                userRole.setUserId(useId);
                userRole.setRoleId(Integer.parseInt(s));
                userRole.setCreateDate(new Date());
                userRole.setUpdateDate(new Date());
                userRoles.add(userRole);
            }
            AssertUtil.isTrue(userRoleMapper.insertBatch(userRoles) <
userRoles.size(), "用户角色分配失败!");
        }
    }


    @Transactional(propagation = Propagation.REQUIRED)
    public void deleteUser(Integer userId) {
        User user = selectByPrimaryKey(userId);
        AssertUtil.isTrue(null == userId || null == user, "待删除记录不存在!");
        // 判断用户是否绑定了角色信息
        int count = userRoleMapper.countUserRoleByUserId(userId);
       // 如果绑定了角色信息则删除对应的数据
        if (count > 0) {
```

```
        AssertUtil.isTrue(userRoleMapper.deleteUserRoleByUserId(userId) !=
count, "用户角色删除失败!");
    }
    user.setIsValid(0);
    AssertUtil.isTrue(updateByPrimaryKeySelective(user) < 1, "用户记录删除失败!");
}
```

更新SQL配置

当更新用户信息时，如果用户有设置过角色，需要默认选中。根据当前点击用户id 查询查询所有角色并标记用户已分配角色。

```
<select id="queryAllRoles"  parameterType="int"  resultType="map">
     SELECT
        r2.id,
        r2.role_name AS roleName,
        CASE
            WHEN IFNULL(t_temp.id, 0 ) = 0 THEN ""
            ELSE "selected" END
        AS "selected"
    FROM
        t_role r2
    LEFT JOIN
        ( SELECT
                r1.id
            FROM
                t_role r1
            LEFT JOIN
                t_user_role ur
            ON
                r1.id = ur.role_id
            WHERE
                ur.user_id = #{userId}
        ) t_temp
    ON
        r2.id = t_temp.id
    WHERE
        r2.is_valid =1
</select>
```

信息 | 结果 1 | 剖析 | 状态

id	roleName	selected
1	系统管理员	selected
2	销售	selected
3	客户经理	
14	技术经理	
17	人事	

修改 JS 文件

- add_update.js

查询角色记录时传入用户id

```
var userId = $("input[name='id']").val();
formSelects.config('selectId',{
    type:"post",
    searchUrl:ctx+"/role/queryAllRoles?userId="+userId,
    keyName: 'roleName',  //自定义返回数据中name的key, 默认 name
    keyVal: 'id' //自定义返回数据中value的key, 默认 value
},true);
```

显示效果

角色管理功能实现

角色查询操作

角色录查询主页面效果

角色记录查询后端实现

- RoleMapper.xml

```
<select id="selectByParams" parameterType="com.xxxx.crm.query.RoleQuery"
resultType="com.xxxx.crm.vo.Role">
      select <include refid="Base_Column_List"/>
      from t_role
      <where>
         is_valid=1
         <if test="null !=roleName and roleName !=''">
             and role_name =#{roleName}
         </if>
      </where>
  </select>
```

- RoleQuery.java 查询类定义

```
public class RoleQuery extends BaseQuery {
    // 角色名
    private String roleName;
}
```

- RoleController.java

添加角色关注主页面视图转发 &列表查询 方法

```
@RequestMapping("index")
public String index(){
    return "role/role";
}

@RequestMapping("list")
@ResponseBody
public Map<String,Object> userList(RoleQuery roleQuery){
    return roleService.queryByParamsForTable(roleQuery);
}
```

角色管理主页面视图添加 & 核心js

- 角色管理主页面管理模板

views/role 目录下创建 role.ftl 文件

```
<!DOCTYPE html>
<html>
<head>
   <title>角色管理</title>
   <#include "../common.ftl">
</head>
<body class="childrenBody">

<form class="layui-form" >
   <blockquote class="layui-elem-quote quoteBox">
      <form class="layui-form">
         <div class="layui-inline">
            <div class="layui-input-inline">
               <input type="text" name="roleName"
```

```
                    class="layui-input
                searchVal" placeholder="角色名" />
            </div>
            <a class="layui-btn search_btn" data-type="reload"><i
                    class="layui-icon">&#xe615;</i> 搜索</a>
        </div>
    </form>
</blockquote>
<table id="roleList" class="layui-table"  lay-filter="roles"></table>

<script type="text/html" id="toolbarDemo">
    <div class="layui-btn-container">
        <a class="layui-btn layui-btn-normal addNews_btn" lay-event="add">
            <i class="layui-icon">&#xe608;</i>
            添加角色
        </a>
        <a class="layui-btn layui-btn-normal delNews_btn" lay-event="grant">
            <i class="layui-icon">&#xe672;</i>
            授权
        </a>
    </div>
</script>
<!--操作-->
<script id="roleListBar" type="text/html">
    <a class="layui-btn layui-btn-xs" id="edit" lay-event="edit">编辑</a>
    <a class="layui-btn layui-btn-xs layui-btn-danger" lay-event="del">删除</a>
</script>
</form>
<script type="text/javascript" src="${ctx}/js/role/role.js"></script>
</body>
</html>
```

- role.js

js/role 目录下添加role.js 初始化角色表格数据

```
layui.use(['table','layer'],function(){
       var layer = parent.layer === undefined ? layui.layer : top.layer,
        $ = layui.jquery,
        table = layui.table;
    //角色列表展示
    var  tableIns = table.render({
        elem: '#roleList',
        url : ctx+'/role/list',
        cellMinWidth : 95,
        page : true,
        height : "full-125",
        limits : [10,15,20,25],
        limit : 10,
        toolbar: "#toolbarDemo",
        id : "roleListTable",
        cols : [[
            {type: "checkbox", fixed:"left", width:50},
            {field: "id", title:'编号',fixed:"true", width:80},
            {field: 'roleName', title: '角色名', minWidth:50, align:"center"},
            {field: 'roleRemark', title: '角色备注', minWidth:100,
align:'center'},
```

```
            {field: 'createDate', title: '创建时间',
align:'center',minWidth:150},
            {field: 'updateDate', title: '更新时间',
align:'center',minWidth:150},
            {title: '操作', minWidth:150,
templet:'#roleListBar',fixed:"right",align:"center"}
        ]]
    });

    // 多条件搜索
    $(".search_btn").on("click",function(){
        table.reload("roleListTable",{
            page: {
                curr: 1 //重新从第 1 页开始
            },
            where: {
                roleName: $("input[name='roleName']").val()
            }
        })
    });

});
```

角色添加与更新

后端代码实现

- RoleService.java

```
@Transactional(propagation = Propagation.REQUIRED)
public void saveRole(Role role){
    AssertUtil.isTrue(StringUtils.isBlank(role.getRoleName()),"请输入角色名!");
    Role temp = roleMapper.queryRoleByRoleName(role.getRoleName());
    AssertUtil.isTrue(null !=temp,"该角色已存在!");
    role.setIsValid(1);
    role.setCreateDate(new Date());
    role.setUpdateDate(new Date());
    AssertUtil.isTrue(insertSelective(role)<1,"角色记录添加失败!");
}

@Transactional(propagation = Propagation.REQUIRED)
public void  updateRole(Role role){

 AssertUtil.isTrue(null==role.getId()||null==selectByPrimaryKey(role.getId()),"
待修改的记录不存在!");
    AssertUtil.isTrue(StringUtils.isBlank(role.getRoleName()),"请输入角色名!");
    Role temp = roleMapper.queryRoleByRoleName(role.getRoleName());
    AssertUtil.isTrue(null !=temp && !(temp.getId().equals(role.getId())),"该角色
已存在!");
    role.setUpdateDate(new Date());
    AssertUtil.isTrue(updateByPrimaryKeySelective(role)<1,"角色记录更新失败!");
}
```

- RoleController.java

```
@RequestMapping("addOrUpdateRolePage")
public String addUserPage(Integer id, Model model){
    if(null !=id){
        model.addAttribute("role",roleService.selectByPrimaryKey(id));
    }
    return "role/add_update";
}

@RequestMapping("save")
@ResponseBody
public ResultInfo saveRole(Role role){
    roleService.saveRole(role);
    return success("角色记录添加成功");
}

@RequestMapping("update")
@ResponseBody
public ResultInfo updateRole(Role role){
    roleService.updateRole(role);
    return success("角色记录更新成功");
}
```

前端核心代码

添加与编辑事件

role.js

```
//头工具栏事件
table.on('toolbar(roles)', function(obj){
    var checkStatus = table.checkStatus(obj.config.id);
    switch(obj.event){
        case "add":
            openAddOrUpdateRoleDialog();
            break;
    };
});

/**
 * 行监听
 */
table.on("tool(roles)", function(obj){
    var layEvent = obj.event;
    if(layEvent === "edit") {
        openAddOrUpdateRoleDialog(obj.data.id);
    }
});

// 打开添加页面
function openAddOrUpdateRoleDialog(uid){
    var url  =  ctx+"/role/addOrUpdateRolePage";
    var title="角色管理-角色添加";
    if(uid){
        url = url+"?id="+uid;
        title="角色管理-角色更新";
    }
    layui.layer.open({
```

```
            title : title,
            type : 2,
            area:["600px","280px"],
            maxmin:true,
            content : url
        });
    }
```

模板文件

views/role 目录下添加 add_update.ftl 文件

```
<!DOCTYPE html>
<html>
<head>
    <#include "../common.ftl">
</head>
<body class="childrenBody">
<form class="layui-form" style="width:80%;">
    <input name="id" type="hidden" value="${(role.id)!}"/>
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">角色名</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input userName"
                   lay-verify="required" name="roleName" id="roleName"
 value="${(role.roleName)!}" placeholder="角色名">
        </div>
    </div>
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">角色备注</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input userName"
                   lay-verify="required" name="roleRemark" id="roleRemark"
value="${(role.roleRemark)!}" placeholder="请输入角色备注">
        </div>
    </div>

    <br/>
    <div class="layui-form-item layui-row layui-col-xs12">
        <div class="layui-input-block">
            <button class="layui-btn layui-btn-lg" lay-submit=""
                    lay-filter="addOrUpdateRole">确认
            </button>
            <button class="layui-btn layui-btn-lg layui-btn-normal">取消</button>
        </div>
    </div>
</form>
<script type="text/javascript" src="${ctx}/js/role/add.update.js"></script>
</body>
</html>
```

add.update.js

js/role 目录下添加 add_update.js 文件

```
layui.use(['form', 'layer'], function () {
    var form = layui.form,
```

```
        layer = parent.layer === undefined ? layui.layer : top.layer,
        $ = layui.jquery;
    form.on("submit(addOrUpdateRole)", function (data) {
        var index = top.layer.msg('数据提交中，请稍候', {icon: 16, time: false,
shade: 0.8});
        //弹出loading
        var url=ctx + "/role/save";
        if($("input[name='id']").val()){
            url=ctx + "/role/update";
        }
        $.post(url, data.field, function (res) {
            if (res.code == 200) {
                setTimeout(function () {
                    top.layer.close(index);
                    top.layer.msg("操作成功！");
                    layer.closeAll("iframe");
                    //刷新父页面
                    parent.location.reload();
                }, 500);
            } else {
                layer.msg(
                        res.msg, {
                            icon: 5
                        }
                    );
            }
        });
        return false;
    });
});
```

角色删除操作

后端代码实现

RoleService.java

```
public void deleteRole(Integer roleId){
    Role temp =selectByPrimaryKey(roleId);
    AssertUtil.isTrue(null==roleId||null==temp,"待删除的记录不存在!");
    temp.setIsValid(0);
    AssertUtil.isTrue(updateByPrimaryKeySelective(temp)<1,"角色记录删除失败!");
}
```

RoleController.java

```
@RequestMapping("delete")
@ResponseBody
public ResultInfo deleteRole(Integer id){
    roleService.deleteRole(id);
    return success("角色记录删除成功");
}
```

前端代码实现

修改role.js 添加删除事件实现角色记录删除操作。

```
/**
     * 行监听
     */
table.on("tool(roles)", function(obj){
    var layEvent = obj.event;
    if(layEvent === "edit") {
        openAddOrUpdateRoleDialog(obj.data.id);
    }else if(layEvent === "del") {
        layer.confirm('确定删除当前角色？', {icon: 3, title: "角色管理"}, function
(index) {
            $.post(ctx+"/role/delete",{id:obj.data.id},function (data) {
                if(data.code==200){
                    layer.msg("操作成功！");
                    tableIns.reload();
                }else{
                    layer.msg(data.msg, {icon: 5});
                }
            });
        })
    }
});
```

角色资源授权

树形数据展示

完成角色记录基本crud功能之后，接下来实现角色授权功能，这里实现角色授权首先完成待授权资源显示功能。对于资源的显示，这里使用开源的tree插件 ztree。

资源数据查询后端实现

前端ztree显示的资源数据格式参考这里。

ModuleMapper.xml

```
<select id="queryAllModules" resultType="com.xxxx.crm.dto.TreeDto">
    select
      id,
      IFNULL(parent_id,0) as pId,
    module_name AS name
    from t_module
    where is_valid=1
</select>
```

ModuleService.java

```
public List<TreeDto> queryAllModules(){
    return moduleMapper.queryAllModules();
}
```

ModuleController.java

```
@RequestMapping("queryAllModules")
@ResponseBody
public List<TreeDto> queryAllModules(){
    return moduleService.queryAllModules();
}
```

资源数据ztree显示

role.js 添加授权点击事件

```
//头工具栏事件
table.on('toolbar(roles)', function(obj){
    var checkStatus = table.checkStatus(obj.config.id);
    switch(obj.event){
        case "add":
            openAddOrUpdateRoleDialog();
            break;
        case "grant":
            openAddGrantDailog(checkStatus.data);
            break;
    };
});

function openAddGrantDailog(datas){
    if(datas.length==0){
        layer.msg("请选择待授权角色记录!", {icon: 5});
        return;
    }
    if(datas.length>1){
        layer.msg("暂不支持批量角色授权!", {icon: 5});
        return;
    }
    var url  =  ctx+"/role/toAddGrantPage?roleId="+datas[0].id;
    var title="角色管理-角色授权";
    layui.layer.open({
        title : title,
        type : 2,
        area:["600px","280px"],
```

```
            maxmin:true,
            content : url
        });
    }
```

RoleController.java 添加视图转发方法

```
@RequestMapping("toAddGrantPage")
public String toAddGrantPage(Integer roleId,Model model){
    model.addAttribute("roleId",roleId);
    return "role/grant";
}
```

准备显示资源数据模板

views/role目录下添加grant.ftl 模板文件()

```
<html>
<head>
   <link rel="stylesheet" href="${ctx}/static/js/zTree_v3-
3.5.32/css/zTreeStyle/zTreeStyle.css" type="text/css">
   <script type="text/javascript" src="${ctx}/static/lib/jquery-3.4.1/jquery-
3.4.1.min.js"></script>
   <script type="text/javascript" src="${ctx}/static/js/zTree_v3-
3.5.32/js/jquery.ztree.core.js"></script>
   <script type="text/javascript" src="${ctx}/static/js/zTree_v3-
3.5.32/js/jquery.ztree.excheck.js"></script>
</head>
<body>
<div id="test1" class="ztree"></div>
<input id="roleId" value="${roleId!}" type="hidden">
<script type="text/javascript">
   var ctx="${ctx}";
</script>
<script type="text/javascript" src="${ctx}/static/js/role/grant.js"></script>
</body>
</html>
```

添加grant.js

```
var zTreeObj;
$(function () {
    loadModuleInfo();
});
function loadModuleInfo() {
    $.ajax({
        type:"post",
        url:ctx+"/module/queryAllModules"
        dataType:"json",
        success:function (data) {
            // zTree 的参数配置，深入使用请参考 API 文档（setting 配置详解）
            var setting = {
                data: {
                    simpleData: {
                        enable: true
                    }
```

```
            },
            view:{
                showLine: false
                // showIcon: false
            },
            check: {
                enable: true,
                chkboxType: { "Y": "ps", "N": "ps" }
            }
        };
        var zNodes =data;
        zTreeObj=$.fn.zTree.init($("#test1"), setting, zNodes);
    }
  })
}
```

角色授权

权限记录添加后端实现

RoleService.java

```
public void addGrant(Integer[] mids, Integer roleId) {
        /**
         * 核心表-t_permission  t_role(校验角色存在)
         *    如果角色存在原始权限  删除角色原始权限
         *      然后添加角色新的权限 批量添加权限记录到t_permission
         */
        Role temp =selectByPrimaryKey(roleId);
        AssertUtil.isTrue(null==roleId||null==temp,"待授权的角色不存在!");
        int count = permissionMapper.countPermissionByRoleId(roleId);
        if(count>0){
            AssertUtil.isTrue(permissionMapper.deletePermissionsByRoleId(roleId)
<count,"权限分配失败!");
        }
        if(null !=mids && mids.length>0){
            List<Permission> permissions=new ArrayList<Permission>();
            for (Integer mid : mids) {
                Permission permission=new Permission();
                permission.setCreateDate(new Date());
                permission.setUpdateDate(new Date());
                permission.setModuleId(mid);
```

```
                permission.setRoleId(roleId);

            permission.setAclValue(moduleMapper.selectByPrimaryKey(mid).getOptValue());
                permissions.add(permission);
            }
            permissionMapper.insertBatch(permissions);
        }
    }
```

RoleController.java

```
@RequestMapping("addGrant")
@ResponseBody
public ResultInfo addGrant(Integer[] mids,Integer roleId){
    roleService.addGrant(mids,roleId);
    return success("权限添加成功");
}
```

权限记录添加前端核心js

修改grant.js文件 添加ztree 复选框点击回调onCheck事件。

```
var zTreeObj;
$(function () {
    loadModuleInfo();
});
function loadModuleInfo() {
    $.ajax({
        type:"post",
        url:ctx+"/module/queryAllModules",
        dataType:"json",
        success:function (data) {
            // zTree 的参数配置，深入使用请参考 API 文档（setting 配置详解）
            var setting = {
                data: {
                    simpleData: {
                        enable: true
                    }
                },
                view:{
                    showLine: false
                    // showIcon: false
                },
                check: {
                    enable: true,
                    chkboxType: { "Y": "ps", "N": "ps" }
                },
                callback: {
                    onCheck: zTreeOnCheck
                }
            };
            var zNodes =data;
            zTreeObj=$.fn.zTree.init($("#test1"), setting, zNodes);
        }
    })
}
```

```
function zTreeOnCheck(event, treeId, treeNode) {
    var nodes= zTreeObj.getCheckedNodes(true);
        var roleId=$("#roleId").val();
        var mids="mids=";
        for(var i=0;i<nodes.length;i++){
            if(i<nodes.length-1){
                mids=mids+nodes[i].id+"&mids=";
            }else{
                mids=mids+nodes[i].id;
            }
        }
    $.ajax({
            type:"post",
            url:ctx+"/role/addGrant",
            data:mids+"&roleId="+roleId,
            dataType:"json",
            success:function (data) {
                console.log(data);
            }
        })
}
```

角色已添加权限记录回显

这里要实现已添加的角色记录权限再次查看或授权时显示原始权限的功能，ztree复选框是否选择属性配置参考这里。

资源查询后端方法实现

ModuleService.java

```
public List<TreeDto> queryAllModules02(Integer roleId) {
    List<TreeDto> treeDtos=moduleMapper.queryAllModules();
    // 根据角色id 查询角色拥有的菜单id  List<Integer>
    List<Integer>
roleHasMids=permissionMapper.queryRoleHasAllModuleIdsByRoleId(roleId);
    if(null !=roleHasMids && roleHasMids.size()>0){
        treeDtos.forEach(treeDto -> {
            if(roleHasMids.contains(treeDto.getId())){
                //  说明当前角色 分配了该菜单
                treeDto.setChecked(true);
            }
        });
    }
    return  treeDtos;
}
```

角色拥有权限sql查询

```
  <select id="queryRoleHasAllModuleIdsByRoleId" parameterType="int"
resultType="java.lang.Integer">
        select module_id from t_permission where role_id=#{roleId}
  </select>
```

ModuleController.java

```
@RequestMapping("queryAllModules")
@ResponseBody
public List<TreeDto> queryAllModules(Integer roleId){
    return moduleService.queryAllModules02(roleId);
}
```

权限回显前端js

这里修改grant.js 查询资源时传入当前选择角色id

```
function loadModuleInfo() {
    $.ajax({
        type:"post",
        url:ctx+"/module/queryAllModules",
        data:{
            roleId:$("#roleId").val()
        },
        dataType:"json",
        success:function (data) {
            // zTree 的参数配置，深入使用请参考 API 文档（setting 配置详解）
            var setting = {
                data: {
                    simpleData: {
                        enable: true
                    }
                },
                view:{
                    showLine: false
                    // showIcon: false
                },
                check: {
                    enable: true,
                    chkboxType: { "Y": "ps", "N": "ps" }
                },
                callback: {
                    onCheck: zTreeOnCheck
                }
            };
            var zNodes =data;
            zTreeObj=$.fn.zTree.init($("#test1"), setting, zNodes);
        }
    })

}
```

角色权限认证

当完成角色权限添加功能后，下一步就是对角色操作的资源进行认证操作，这里对于认证包含两块:

1. 菜单级别显示控制
2. 后端方法访问控制

菜单级别访问控制实现

登录用户扮演的不同角色来对登录用户操作的菜单进行动态控制显示操作，这里显示的控制使用freemarker指令+内建函数实现，指令与内建函数操作参考这里。

登录用户角色拥有权限查询实现

IndexController.java

```
/**
 * 后端管理主页面
 * @return
 */
@RequestMapping("main")
public String main(HttpServletRequest request){
    Integer userId = LoginUserUtil.releaseUserIdFromCookie(request);
    request.setAttribute("user",userService.selectByPrimaryKey(userId));
    List<String>
permissions=permissionService.queryUserHasRolesHasPermissions(userId);
    request.getSession().setAttribute("permissions",permissions);
    return "main";
}
```

PermissionService.java

```
@Service
public class PermissionService extends BaseService<Permission,Integer> {

    @Autowired
    private PermissionMapper permissionMapper;

    public List<String> queryUserHasRolesHasPermissions(Integer userId) {
        return permissionMapper.queryUserHasRolesHasPermissions(userId);
    }
}
```

PermissionMapper.java & PermissionMapper.xml

```
public interface PermissionMapper extends BaseMapper<Permission,Integer> {
    List<String>  queryUserHasRolesHasPermissions(Integer userId);
}
```

```
<select id="queryUserHasRolesHasPermissions" parameterType="int"
resultType="java.lang.String">
    select distinct p.acl_value
    from t_user_role ur left join  t_permission p on ur.role_id = p.role_id
    where ur.user_id=#{userId}
</select>
```

系统主页面菜单显示指令控制

这里仅显示部分菜单控制。

```
<#if permissions?seq_contains("10")>
    <li class="layui-nav-item">
```

```
<a href="javascript:;" class="layui-menu-tips"><i class="fa fa-street-view"></i><span class="layui-left-nav"> 营销管理</span> <span class="layui-nav-more"></span></a>
        <dl class="layui-nav-child">
            <#if permissions?seq_contains("1010")>
                <dd>
                    <a href="javascript:;" class="layui-menu-tips" data-type="tabAdd" data-tab-mpi="m-p-i-1" data-tab="sale_chance/index" target="_self"><i class="fa fa-tty"></i><span class="layui-left-nav"> 营销机会管理</span></a>
                </dd>
                </#if>
            <#if permissions?seq_contains("1020")>
                <dd>
                    <a href="javascript:;" class="layui-menu-tips" data-type="tabAdd" data-tab-mpi="m-p-i-2" data-tab="cus_dev_plan/index" target="_self"><i class="fa fa-ellipsis-h"></i><span class="layui-left-nav"> 客户开发计划</span></a>
                </dd>
                </#if>
        </dl>
    </li>
    </#if>
```

后端方法级别访问控制

实现了菜单级别显示控制，但最终客户端有可能会通过浏览器来输入资源地址从而越过ui界面来访问后端资源，所以接下来加入控制方法级别资源的访问控制操作，这里使用aop+自定义注解实现

自定义注解@RequirePermission

```
@Target({ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface RequirePermission {
    String code() default "";
}
```

方法级别使用注解

```
@RequestMapping("list")
@ResponseBody
@RequirePermission(code = "101001")
public Map<String,Object> querySaleChancesByParams(Integer flag,HttpServletRequest request,SaleChanceQuery saleChanceQuery){
    if(null !=flag &&flag==1){
        // 查询分配给当前登录用户 营销记录

 saleChanceQuery.setAggsinMan(LoginUserUtil.releaseUserIdFromCookie(request));
    }
    return  saleChanceService.queryByParamsForTable(saleChanceQuery);
}
```

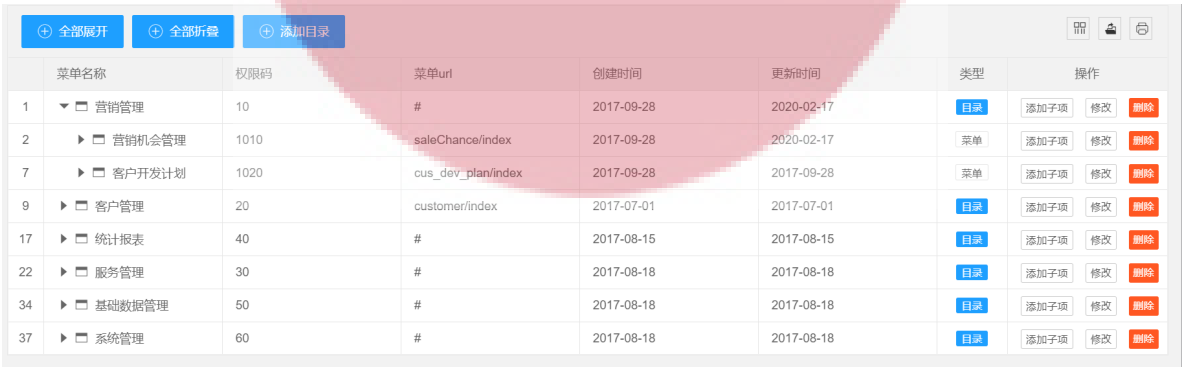
定义aop切面类 拦截指定注解标注的方法

```
@Component
@Aspect
public class PermissionProxy {

    @Autowired
    private HttpSession session;

    @Around(value = "@annotation(com.xxxx.crm.annotaions.RequirePermission)")
    public  Object around(ProceedingJoinPoint pjp) throws Throwable {
        List<String> permissions = (List<String>)
session.getAttribute("permissions");
        if(null == permissions || permissions.size()==0){
            throw  new NoLoginException();
        }
        Object result =null;
        MethodSignature methodSignature = (MethodSignature) pjp.getSignature();
        RequirePermission requirePermission =
methodSignature.getMethod().getDeclaredAnnotation(RequirePermission.class);
        if(!(permissions.contains(requirePermission.code()))){
            throw  new NoLoginException();
        }
        result= pjp.proceed();
        return result;
    }
}
```

资源管理功能实现

资源记录查询

页面效果



后端代码实现

菜单资源展示这里使用layui treeTable组件显示，treetable使用参考这里。

ModuleMapper.xml

```
<select id="queryModules"  resultType="com.xxxx.crm.vo.Module">
  SELECT
      id,
      module_name,
      module_style,
      url,
```

```
        parent_id,
        opt_value,
        orders,
        create_date,
        update_date,
        grade
    FROM
        t_module
    WHERE
        is_valid = 1
</select>
```

ModuleService.java

```
public Map<String,Object> moduleList(){
    Map<String,Object> result = new HashMap<String,Object>();
    List<Module> modules =moduleMapper.queryModules();
    result.put("count",modules.size());
    result.put("data",modules);
    result.put("code",0);
    result.put("msg","");
    return result;
}
```

ModuleController.java

```
@RequestMapping("/index")
public String index(){
    return "module/module";
}

@RequestMapping("list")
@ResponseBody
public Map<String,Object> moduleList(){
    return moduleService.moduleList();
}
```

前端代码实现

module.ftl 模板文件添加

views/module 目录下添加module.ftl 模板文件

```
<!DOCTYPE html>
<html>
<head>
    <title>资源管理</title>
    <#include "../common.ftl">
</head>
<body class="childrenBody">
    <table id="munu-table" class="layui-table" lay-filter="munu-table"></table>

    <!-- 操作列 -->
    <script type="text/html" id="auth-state">
```

```
        <a class="layui-btn layui-btn-primary layui-btn-xs" lay-event="add">添加子项</a>
        <a class="layui-btn layui-btn-primary layui-btn-xs" lay-event="edit">修改</a>
        <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除</a>
    </script>

    <script type="text/html" id="toolbarDemo">
        <div class="layui-btn-container">
            <a class="layui-btn layui-btn-normal addNews_btn" lay-event="expand">
                <i class="layui-icon">&#xe608;</i>
                全部展开
            </a>
            <a class="layui-btn layui-btn-normal addNews_btn" lay-event="fold">
                <i class="layui-icon">&#xe608;</i>
                全部折叠
            </a>
            <a class="layui-btn layui-btn-normal addNews_btn" lay-event="add">
                <i class="layui-icon">&#xe608;</i>
                添加目录
            </a>
        </div>
    </script>

     <script type="text/javascript" src="${ctx}/static/js/module/module.js">
</script>
</body>
</html>
```

module.js 文件添加

js/module 目录下添加module.js，初始化表格数据

```
layui.use(['table', 'treetable'], function () {
    var $ = layui.jquery;
    var table = layui.table;
    var treeTable = layui.treetable;
    treeTable.render({
        treeColIndex: 1,
        treeSpid: -1,
        treeIdName: 'id',
        treePidName: 'parentId',
        elem: '#munu-table',
        url: ctx+'/module/list',
        toolbar: "#toolbarDemo",
        treeDefaultClose:true,
        page: true,
        cols: [[
            {type: 'numbers'},
            {field: 'moduleName', minWidth: 100, title: '菜单名称'},
            {field: 'optValue', title: '权限码'},
            {field: 'url', title: '菜单url'},
            {field: 'createDate', title: '创建时间'},
            {field: 'updateDate', title: '更新时间'},
            {
```

```
                field: 'grade', width: 80, align: 'center', templet: function
(d) {
                    if (d.grade == 0) {
                        return '<span class="layui-badge layui-bg-blue">目录
</span>';
                    }
                    if(d.grade==1){
                        return '<span class="layui-badge-rim">菜单</span>';
                    }
                    if (d.grade == 2) {
                        return '<span class="layui-badge layui-bg-gray">按钮
</span>';
                    }
                }, title: '类型'
            },
            {templet: '#auth-state', width: 180, align: 'center', title: '操作'}
        ]],
        done: function () {
            layer.closeAll('loading');
        }
    });

    table.on('toolbar(munu-table)', function(obj){
        switch(obj.event){
            case "expand":
                treeTable.expandAll('#munu-table');
                break;
            case "fold":
                treeTable.foldAll('#munu-table');
                break;
        };
    });

});
```

资源记录添加 & 更新

资源记录添加

实现思路

```
/**
 * 1.参数校验
 *     模块名-module_name
 *         非空  同一层级下模块名唯一
 *     url
 *         二级菜单  非空  不可重复
 *     上级菜单-parent_id
 *         一级菜单   null
 *         二级|三级菜单 parent_id 非空 必须存在
 *      层级-grade
 *          非空  0|1|2
 *       权限码 optValue
 *          非空  不可重复
 * 2.参数默认值设置
```

```
is_valid  create_date update_date
 * 3.执行添加 判断结果
 */
```

后端代码实现

```
@Transactional(propagation = Propagation.REQUIRED)
public void saveModule(Module module){
    AssertUtil.isTrue(StringUtils.isBlank(module.getModuleName()),"请输入菜单
名!");
    Integer grade =module.getGrade();
    AssertUtil.isTrue(null== grade|| !(grade==0||grade==1||grade==2),"菜单层级不合
法!");
    AssertUtil.isTrue(null
!=moduleMapper.queryModuleByGradeAndModuleName(module.getGrade(),module.getModul
eName()),"该层级下菜单重复!");
    if(grade==1){
        AssertUtil.isTrue(StringUtils.isBlank(module.getUrl()),"请指定二级菜单url
值");
        AssertUtil.isTrue(null
!=moduleMapper.queryModuleByGradeAndUrl(module.getGrade(),module.getUrl()),"二级
菜单url不可重复!");
    }
    if(grade !=0){
        Integer parentId = module.getParentId();
        AssertUtil.isTrue(null==parentId ||
null==selectByPrimaryKey(parentId),"请指定上级菜单!");
    }
    AssertUtil.isTrue(StringUtils.isBlank(module.getOptValue()),"请输入权限码!");
    AssertUtil.isTrue(null
!=moduleMapper.queryModuleByOptValue(module.getOptValue()),"权限码重复!");

    module.setIsValid((byte)1);
    module.setCreateDate(new Date());
    module.setUpdateDate(new Date());
    AssertUtil.isTrue(insertSelective(module)<1,"菜单添加失败!");
}
```

资源记录更新

实现思路

```
/**
 * 1.参数校验
 *     id 非空 记录存在
 *     模块名-module_name
 *         非空  同一层级下模块名唯一
 *     url
 *         二级菜单  非空  不可重复
 *     上级菜单-parent_id
 *         二级|三级菜单 parent_id 非空 必须存在
 *      层级-grade
 *          非空  0|1|2
 *       权限码 optValue
 *          非空  不可重复
```

```
     * 2.参数默认值设置
     *       update_date
     * 3.执行更新 判断结果
     */
```

后端核心代码

```
@Transactional(propagation = Propagation.REQUIRED)
public void updateModule(Module module){
    AssertUtil.isTrue(null == module.getId() || null==
selectByPrimaryKey(module.getId()),"待更新记录不存在!");
    AssertUtil.isTrue(StringUtils.isBlank(module.getModuleName()),"请指定菜单名
称!");
    Integer grade =module.getGrade();
    AssertUtil.isTrue(null== grade|| !(grade==0||grade==1||grade==2),"菜单层级不合
法!");
    Module temp
=moduleMapper.queryModuleByGradeAndModuleName(grade,module.getModuleName());
    if(null !=temp){
        AssertUtil.isTrue(!(temp.getId().equals(module.getId())),"该层级下菜单已存
在!");
    }

    if(grade==1){
        AssertUtil.isTrue(StringUtils.isBlank(module.getUrl()),"请指定二级菜单url
值");
        temp =moduleMapper.queryModuleByGradeAndUrl(grade,module.getUrl());
        if(null !=temp){
            AssertUtil.isTrue(!(temp.getId().equals(module.getId())),"该层级下url
已存在!");
        }
    }

    if(grade !=0){
        Integer parentId = module.getParentId();
        AssertUtil.isTrue(null==parentId ||
null==selectByPrimaryKey(parentId),"请指定上级菜单!");
    }
    AssertUtil.isTrue(StringUtils.isBlank(module.getOptValue()),"请输入权限码!");

    temp =moduleMapper.queryModuleByOptValue(module.getOptValue());
    if(null !=temp){
        AssertUtil.isTrue(!(temp.getId().equals(module.getId())),"权限码已存在!");
    }
    module.setUpdateDate(new Date());
    AssertUtil.isTrue(updateByPrimaryKeySelective(module)<1,"菜单更新失败!");
}
```

控制层核心代码

```
// 添加资源页视图转发
@RequestMapping("addModulePage")
public String addModulePage(Integer grade,Integer parentId,Model model){
    model.addAttribute("grade",grade);
```

```
        model.addAttribute("parentId",parentId);
        return "module/add";
    }
    // 更新资源页视图转发
    @RequestMapping("updateModulePage")
    public String updateModulePage(Integer id,Model model){
        model.addAttribute("module",moduleService.selectByPrimaryKey(id));
        return "module/update";
    }

    @RequestMapping("save")
    @ResponseBody
    public ResultInfo saveModule(Module module){
        moduleService.saveModule(module);
        return success("菜单添加成功");
    }


    @RequestMapping("queryAllModulesByGrade")
    @ResponseBody
    public List<Map<String,Object>> queryAllModulesByGrade(Integer grade){
        return moduleService.queryAllModulesByGrade(grade);
    }


    @RequestMapping("update")
    @ResponseBody
    public ResultInfo updateModule(Module module){
        moduleService.updateModule(module);
        return success("菜单更新成功");
    }
```

前端核心代码

添加与编辑监听事件

```
    //监听工具条
    table.on('tool(munu-table)', function (obj) {
        var data = obj.data;
        var layEvent = obj.event;
        if (layEvent === 'add') {
            if(data.grade==2){
                layer.msg("暂不支持四级菜单添加操作!");
                return;
            }
            openAddModuleDialog(data.grade+1,data.id);
        } else if (layEvent === 'edit') {
            // 记录修改
            openUpdateModuleDialog(data.id);
        }
    });

    table.on('toolbar(munu-table)', function(obj){
        switch(obj.event){
            case "expand":
                treeTable.expandAll('#munu-table');
                break;
```

```
            case "fold":
                treeTable.foldAll('#munu-table');
                break;
            case "add":
                openAddModuleDialog(0,-1);
                break;
        };
    });

     // 打开添加菜单对话框
        function openAddModuleDialog(grade,parentId){
            var grade=grade;
            var url  =  ctx+"/module/addModulePage?
    grade="+grade+"&parentId="+parentId;
            var title="菜单添加";
            layui.layer.open({
                title : title,
                type : 2,
                area:["700px","450px"],
                maxmin:true,
                content : url
            });
        }

        function openUpdateModuleDialog(id){
            var url  =  ctx+"/module/updateModulePage?id="+id;
            var title="菜单更新";
            layui.layer.open({
                title : title,
                type : 2,
                area:["700px","450px"],
                maxmin:true,
                content : url
            });
        }
```

视图添加

views/module 目录下分别添加add.ftl update.ftl 文件

add.ftl 资源添加模板文件

```
<!DOCTYPE html>
<html>
<head>
    <#include "../common.ftl">
</head>
<body class="childrenBody">
<form class="layui-form" style="width:80%;">
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">菜单名</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input userName"
                   lay-verify="required" name="moduleName" id="moduleName"
placeholder="请输入菜单名">
        </div>
```

```
        </div>
        <div class="layui-form-item layui-row layui-col-xs12">
            <label class="layui-form-label">菜单样式</label>
            <div class="layui-input-block">
                <input type="text" class="layui-input userName"
                       name="moduleStyle" id="moduleStyle" placeholder="请输入菜单样
式">
            </div>
        </div>
        <div class="layui-form-item layui-row layui-col-xs12">
            <label class="layui-form-label">排序</label>
            <div class="layui-input-block">
                <input type="text" class="layui-input userName"
                       name="orders" id="orders" placeholder="请输入排序值">
            </div>
        </div>
        <div class="layui-form-item layui-row layui-col-xs12">
            <label class="layui-form-label">权限码</label>
            <div class="layui-input-block">
                <input type="text" class="layui-input userName"
                       lay-verify="required" name="optValue" id="optValue"
placeholder="请输入菜单权限码">
            </div>
        </div>
        <div class="layui-form-item layui-row layui-col-xs12">
            <label class="layui-form-label">菜单级别</label>
            <div class="layui-input-block">
                <#if grade??>
                    <select name="grade" >
                        <option value="0" <#if grade==0>selected="selected"</#if> >
一级菜单</option>
                        <option value="1" <#if grade==1>selected="selected"</#if>>二
级菜单</option>
                        <option value="2" <#if grade==2>selected="selected"</#if>>三
级菜单</option>
                    </select>
                </#if>
            </div>
        </div>

        <#if grade==1>
            <div class="layui-form-item layui-row layui-col-xs12">
                <label class="layui-form-label">菜单url</label>
                <div class="layui-input-block">
                    <input type="text" class="layui-input userName"
                           lay-verify="required" name="url" id="url" placeholder="请
输入菜单url">
                </div>
            </div>
        </#if>

        <!--
           添加根级菜单
        -->
        <input name="parentId" type="hidden" value="${parentId}"/>
        <br/>
        <div class="layui-form-item layui-row layui-col-xs12">
            <div class="layui-input-block">
```

```
            <button class="layui-btn layui-btn-lg" lay-submit=""
                    lay-filter="addModule">确认
            </button>
            <button class="layui-btn layui-btn-lg layui-btn-normal">取消</button>
        </div>
    </div>
</form>
<script type="text/javascript" src="${ctx}/static/js/module/add.js"></script>
</body>
</html>
```

add.js 文件 实现表单数据提交操作

js/module 目录下添加add.js 实现资源添加表单数据提交

```
layui.use(['form', 'layer'], function () {
    var form = layui.form,
        layer = parent.layer === undefined ? layui.layer : top.layer,
        $ = layui.jquery;
    form.on("submit(addModule)", function (data) {
        var index = top.layer.msg('数据提交中，请稍候', {icon: 16, time: false,
shade: 0.8});
        //弹出loading
        $.post(ctx+"/module/save", data.field, function (res) {
            if (res.code == 200) {
                setTimeout(function () {
                    top.layer.close(index);
                    top.layer.msg("操作成功！");
                    layer.closeAll("iframe");
                    //刷新父页面
                    parent.location.reload();
                }, 500);
            } else {
                layer.msg(
                    res.msg, {
                        icon: 5
                    }
                );
            }
        });
        return false;
    });
});
```

update.ftl 资源更新模板文件

```
<!DOCTYPE html>
<html>
<head>
    <#include "../common.ftl">
</head>
<body class="childrenBody">
<form class="layui-form" style="width:80%;">
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">菜单名</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input userName"
```

```
                lay-verify="required" name="moduleName" id="moduleName"
value="${module.moduleName}"   placeholder="请输入菜单名">
        </div>
    </div>
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">菜单样式</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input userName"
                   name="moduleStyle" id="moduleStyle"
value="${(module.moduleStyle)!""}" placeholder="请输入菜单样式">
        </div>
    </div>
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">排序</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input userName"
                   name="orders" id="orders" placeholder="请输入排序值"
value="${(module.orders)!""}">
        </div>
    </div>
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">权限码</label>
        <div class="layui-input-block">
            <input type="text" class="layui-input userName"
                   lay-verify="required" name="optValue" id="optValue"
placeholder="请输入菜单权限码" value="${module.optValue}">
        </div>
    </div>
    <div class="layui-form-item layui-row layui-col-xs12">
        <label class="layui-form-label">菜单级别</label>
        <div class="layui-input-block">
                <select name="grade" >
                    <option value="0" <#if module.grade==0>selected="selected"
</#if> >一级菜单</option>
                    <option value="1" <#if module.grade==1>selected="selected"
</#if>>二级菜单</option>
                    <option value="2" <#if module.grade==2>selected="selected"
</#if>>三级菜单</option>
                </select>
        </div>
    </div>

    <#if module.grade==1>
        <div class="layui-form-item layui-row layui-col-xs12">
            <label class="layui-form-label">菜单url</label>
            <div class="layui-input-block">
                <input type="text" class="layui-input userName"
                       lay-verify="required" name="url" id="url" placeholder="请
输入菜单url" value="${(module.url)!""}">
            </div>
        </div>
    </#if>


    <!--
       添加根级菜单
    -->
    <input name="parentId" type="hidden" value="${module.parentId}"/>
```

```
        <input name="id" type="hidden" value="${module.id}"/>
        <br/>
        <div class="layui-form-item layui-row layui-col-xs12">
            <div class="layui-input-block">
                <button class="layui-btn layui-btn-lg" lay-submit=""
                        lay-filter="updateModule">确认
                </button>
                <button class="layui-btn layui-btn-lg layui-btn-normal">取消</button>
            </div>
        </div>
    </form>
    <script type="text/javascript" src="${ctx}/static/js/module/update.js"></script>
    </body>
    </html>
```

update.js 文件 实现表单数据提交操作

js/module 目录下添加update.js 实现资源添加表单数据提交

```
layui.use(['form', 'layer'], function () {
    var form = layui.form,
        layer = parent.layer === undefined ? layui.layer : top.layer,
        $ = layui.jquery;
    form.on("submit(updateModule)", function (data) {
        var index = top.layer.msg('数据提交中，请稍候', {icon: 16, time: false,
shade: 0.8});
        //弹出loading
        $.post(ctx+"/module/update", data.field, function (res) {
            if (res.code == 200) {
                setTimeout(function () {
                    top.layer.close(index);
                    top.layer.msg("操作成功！");
                    layer.closeAll("iframe");
                    //刷新父页面
                    parent.location.reload();
                }, 500);
            } else {
                layer.msg(
                    res.msg, {
                        icon: 5
                    }
                );
            }
        });
        return false;
    });
});
```

资源记录删除

资源记录删除后端实现

ModuleService.java

```
@Transactional(propagation = Propagation.REQUIRED)
```

```
    public void deleteModuleById(Integer mid){
        Module temp =selectByPrimaryKey(mid);
        AssertUtil.isTrue(null == mid || null == temp,"待删除记录不存在!");
        /**
         * 如果存在子菜单 不允许删除
         */
        int count = moduleMapper.countSubModuleByParentId(mid);
        AssertUtil.isTrue(count>0,"存在子菜单，不支持删除操作!");

        //  权限表
        count =permissionMapper.countPermissionsByModuleId(mid);
        if(count>0){
            AssertUtil.isTrue(permissionMapper.deletePermissionsByModuleId(mid)
<count,"菜单删除失败!");
        }
        temp.setIsValid((byte) 0);
        AssertUtil.isTrue(updateByPrimaryKeySelective(temp)<1,"菜单删除失败!");

    }
```

ModuleController.java

```
@RequestMapping("delete")
@ResponseBody
public ResultInfo deleteModule(Integer id){
    moduleService.deleteModuleById(id);
    return success("菜单删除成功");
}
```

资源记录删除前端实现

修改module.js 添加删除监听事件实现资源记录删除操作。

```
//监听工具条
table.on('tool(munu-table)', function (obj) {
    var data = obj.data;
    var layEvent = obj.event;
    if (layEvent === 'add') {
        if(data.grade==2){
            layer.msg("暂不支持四级菜单添加操作!");
            return;
        }
        openAddModuleDialog(data.grade+1,data.id);
    } else if (layEvent === 'edit') {
        // 记录修改
        openUpdateModuleDialog(data.id);
    } else if (layEvent === 'del') {
        layer.confirm('确定删除当前菜单？', {icon: 3, title: "菜单管理"}, function
(index) {
            $.post(ctx+"/module/delete",{id:data.id},function (data) {
                if(data.code==200){
                    layer.msg("操作成功！");
                    window.location.reload();
                }else{
                    layer.msg(data.msg, {icon: 5});
                }
```

```
            });
        })
    }
});
```