尚筹网

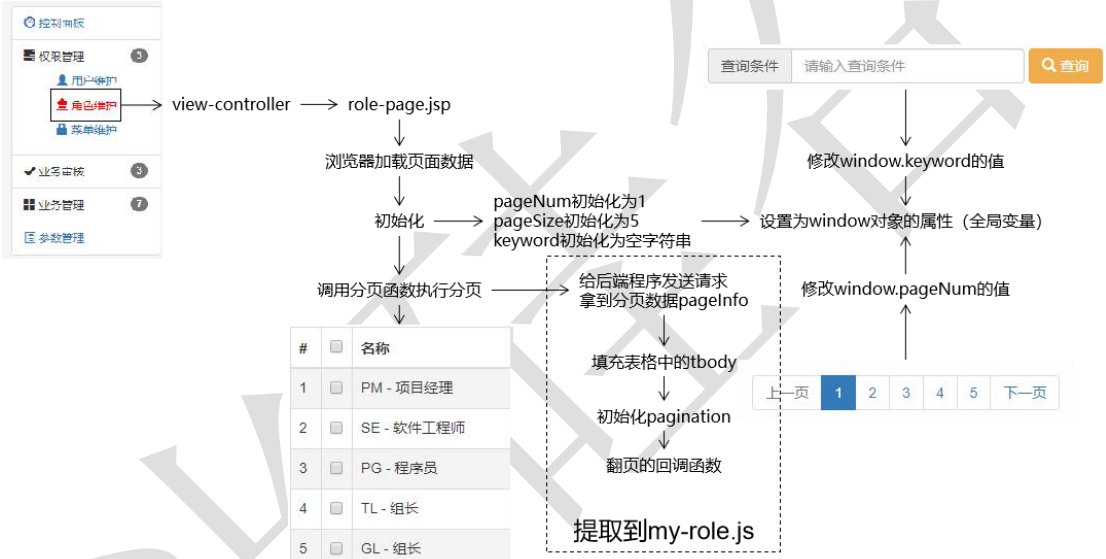
[]

1 角色分页操作

1.1 目标

将角色数据进行分页显示

1.2 思路

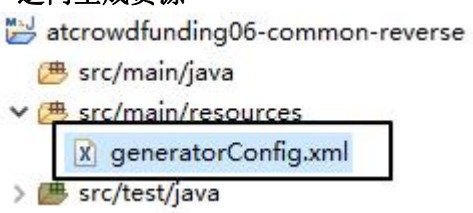


1.3 代码：后端

1.3.1 创建数据库表

```
CREATE TABLE `project_crowd`.`t_role` ( `id` INT NOT NULL, `name` CHAR(100), PRIMARY KEY (`id`) );
ALTER TABLE `project_crowd`.`t_role` CHANGE `id` `id` INT(11) NOT NULL AUTO_INCREMENT;
```

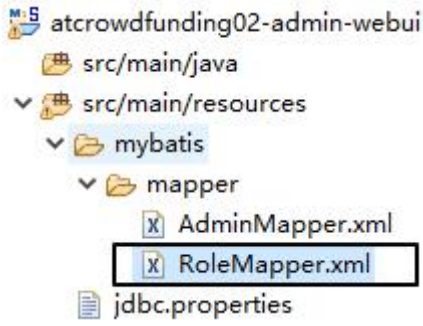
1.3.2 逆向生成资源



```
<table tableName="t_role" domainObjectName="Role" />
```

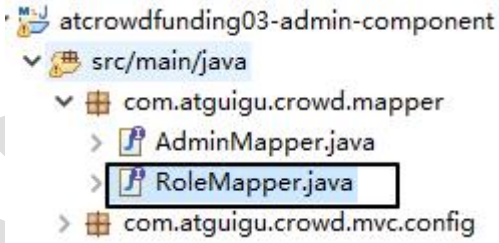
生成资源后各归其位！

1.3.3 SQL 语句



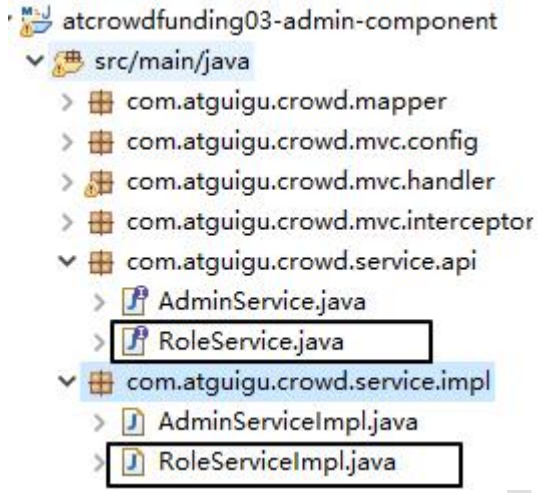

```
<select id="selectRoleByKeyword" resultMap="BaseResultMap">
    select id, name from t_role
    where name like concat("%",#{keyword},"%")
</select>
```

1.3.4 RoleMapper 接口



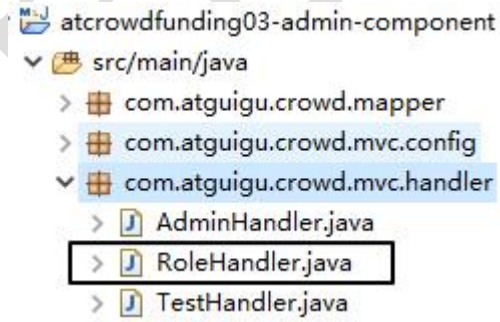

```
List<Role> selectRoleByKeyword(String keyword);
```

1.3.5 RoleService 接口和实现




```
@Override
public PageInfo<Role> getPageInfo(Integer pageNum, Integer pageSize, String keyword) {

    // 1.开启分页功能
    PageHelper.startPage(pageNum, pageSize);

    // 2.执行查询
    List<Role> roleList = roleMapper.selectRoleByKeyword(keyword);

    // 3.封装为 PageInfo 对象返回
    return new PageInfo<>(roleList);
}
```

1.3.6 RoleHandler

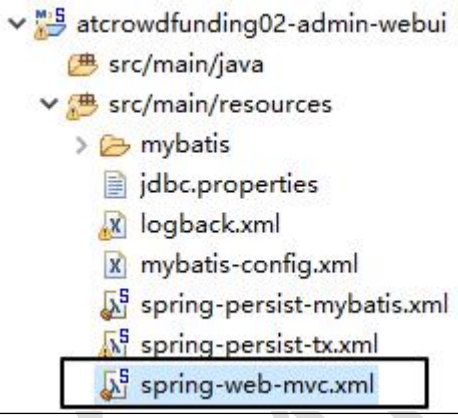



```
@ResponseBody
@RequestMapping("/role/get/page/info.json")
public ResultEntity<PageInfo<Role>> getPageInfo(
                @RequestParam(value="pageNum", defaultValue="1") Integer pageNum,
                @RequestParam(value="pageSize", defaultValue="5") Integer pageSize,
                @RequestParam(value="keyword", defaultValue="") String keyword
```

```
        ){

    // 调用 Service 方法获取分页数据
    PageInfo<Role> pageInfo = roleService.getPageInfo(pageNum, pageSize, keyword);

    // 封装到 ResultEntity 对象中返回（如果上面的操作抛出异常，交给异常映射机制处理）
    return ResultEntity.successWithData(pageInfo);
}
```

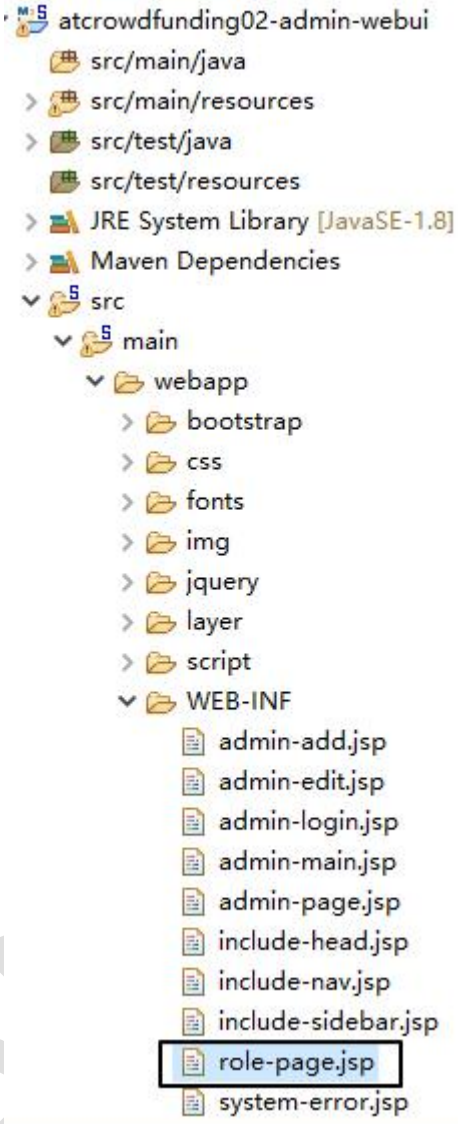
1.4 代码：过渡

1.4.1 配置 view-controller

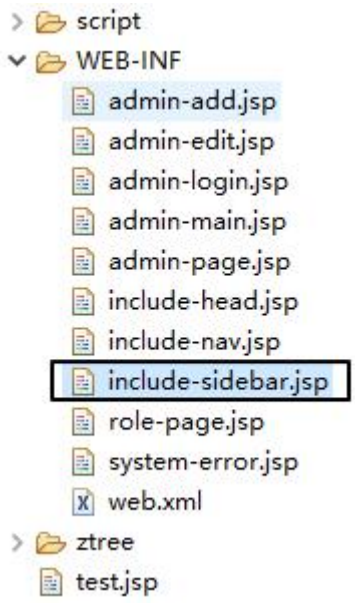



```
<mvc:view-controller path="/role/to/page.html" view-name="role-page"/>
```

1.4.2 完善 role-page.jsp



1.4.3 修改“角色维护”超链接



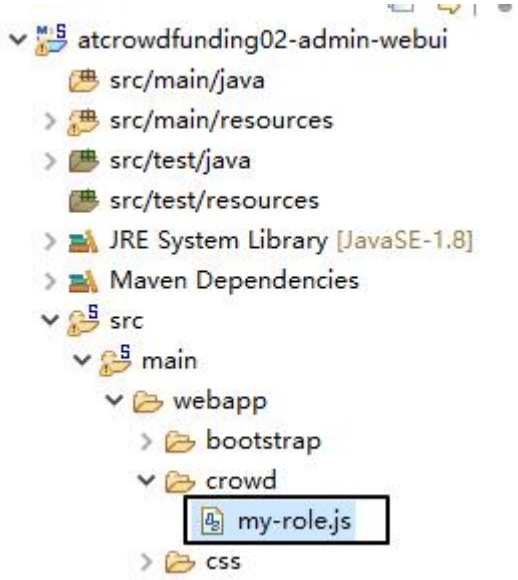

```
<li style="height: 30px;"><a href="role/to/page.html"><i class="glyphicon glyphicon-king"></i>角色维护</a></li>
```

1.5 代码：前端

1.5.1 初始化数据

```
$(function(){

    // 1.为分页操作准备初始化数据
    window.pageNum = 1;
    window.pageSize = 5;
    window.keyword = "";

    // 2.调用执行分页的函数，显示分页效果
    generatePage();

});
```

1.5.2 创建外部 JavaScript 文件



在 role-page.jsp 引入外部 JavaScript 文件 my-role.js

```
<script type="text/javascript" src="crowd/my-role.js"></script>
```

1.5.3 在 role-page.jsp 中引入 Pagination 环境

```
<script type="text/javascript" src="jquery/jquery.pagination.js"></script>
<script type="text/javascript" src="crowd/my-role.js"></script>
```

1.5.4 外部函数一：generatePage()

```
// 执行分页，生成页面效果，任何时候调用这个函数都会重新加载页面
function generatePage() {

    // 1.获取分页数据
    var pageInfo = getPageInfoRemote();

    // 2.填充表格
    fillTableBody(pageInfo);

}
```

1.5.5 外部函数二：getPageInfoRemote()

```
// 远程访问服务器端程序获取 pageInfo 数据
function getPageInfoRemote() {
```

```
// 调用$.ajax()函数发送请求并接受$.ajax()函数的返回值
var ajaxResult = $.ajax({
    "url": "role/get/page/info.json",
    "type":"post",
    "data": {
        "pageNum": window.pageNum,
        "pageSize": window.pageSize,
        "keyword": window.keyword
    },
    "async":false,
    "dataType":"json"
});

console.log(ajaxResult);

// 判断当前响应状态码是否为 200
var statusCode = ajaxResult.status;

// 如果当前响应状态码不是 200，说明发生了错误或其他意外情况，显示提示消息，让当前函数停止执行
if(statusCode != 200) {
    layer.msg("失败！响应状态码="+statusCode+" 说明信息="+ajaxResult.statusText);
    return null;
}

// 如果响应状态码是 200，说明请求处理成功，获取 pageInfo
var resultEntity = ajaxResult.responseJSON;

// 从 resultEntity 中获取 result 属性
var result = resultEntity.result;

// 判断 result 是否成功
if(result == "FAILED") {
    layer.msg(resultEntity.message);
    return null;
}

// 确认 result 为成功后获取 pageInfo
var pageInfo = resultEntity.data;

// 返回 pageInfo
```

```
    return pageInfo;
}
```

1.5.6 外部函数三：fillTableBody(pageInfo)

```
// 填充表格
function fillTableBody(pageInfo) {

    // 清除 tbody 中的旧的内容
    $("#rolePageBody").empty();

    // 这里清空是为了让没有搜索结果时不显示页码导航条
    $("#Pagination").empty();

    // 判断 pageInfo 对象是否有效
    if(pageInfo == null || pageInfo == undefined || pageInfo.list == null || pageInfo.list.length == 0) {
        $("#rolePageBody").append("<tr><td colspan='4' align='center'>抱歉！没有查询到您搜索的数据！ </td></tr>");

        return ;
    }

    // 使用 pageInfo 的 list 属性填充 tbody
    for(var i = 0; i < pageInfo.list.length; i++) {

        var role = pageInfo.list[i];

        var roleId = role.id;

        var roleName = role.name;

        var numberTd = "<td>"+(i+1)+"</td>";
        var checkboxTd = "<td><input type='checkbox'></td>";
        var roleNameTd = "<td>"+roleName+"</td>";

        var checkBtn = "<button type='button' class='btn btn-success btn-xs'><i class='glyphicon glyphicon-check'></i></button>";
        var pencilBtn = "<button type='button' class='btn btn-primary btn-xs'><i class='glyphicon glyphicon-pencil'></i></button>";
        var removeBtn = "<button type='button' class='btn btn-danger btn-xs'><i class='glyphicon glyphicon-remove'></i></button>";
```

```
            var buttonTd = "<td>"+checkBtn+" "+pencilBtn+" "+removeBtn+"</td>";

            var tr = "<tr>"+numberTd+checkboxTd+roleNameTd+buttonTd+"</tr>";

            $("#rolePageBody").append(tr);
        }

        // 生成分页导航条
        generateNavigator(pageInfo);
}
```

1.5.7 外部函数四：generateNavigator(pageInfo)

```
// 生成分页页码导航条
function generateNavigator(pageInfo) {

    // 获取总记录数
    var totalRecord = pageInfo.total;

    // 声明相关属性
    var properties = {
        "num_edge_entries": 3,
        "num_display_entries": 5,
        "callback": paginationCallBack,
        "items_per_page": pageInfo.pageSize,
        "current_page": pageInfo.pageNum - 1,
        "prev_text": "上一页",
        "next_text": "下一页"
    }

    // 调用 pagination()函数
    $("#Pagination").pagination(totalRecord, properties);
}
```

1.5.8 外部函数五：paginationCallBack(pageIndex, jQuery)

```
// 翻页时的回调函数
function paginationCallBack(pageIndex, jQuery) {

    // 修改 window 对象的 pageNum 属性
    window.pageNum = pageIndex + 1;
```

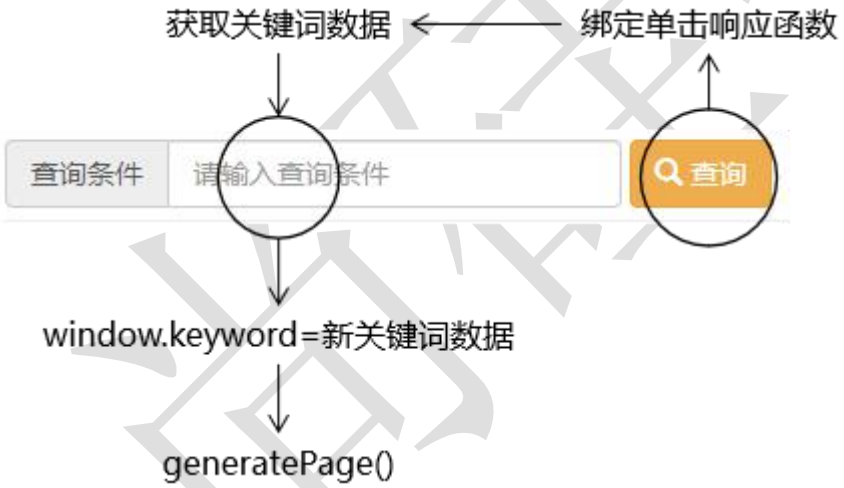
```
    // 调用分页函数
    generatePage();

    // 取消页码超链接的默认行为
    return false;

}
```

2 角色查询操作

2.1 目标

把页面上的“查询”表单和已经封装好的执行分页的函数连起来即可。

2.2 思路



2.3 代码

2.3.1 HTML 代码中标记 id

```
<form class="form-inline" role="form" style="float: left;">
    <div class="form-group has-feedback">
        <div class="input-group">
            <div class="input-group-addon">查询条件</div>
            <input id="keywordInput" class="form-control has-success" type="text"
                placeholder="请输入查询条件">
        </div>
    </div>
```

```
    <button id="searchBtn" type="button" class="btn btn-warning">
        <i class="glyphicon glyphicon-search"></i> 查询
    </button>
</form>
```

2.3.2 jQuery 代码

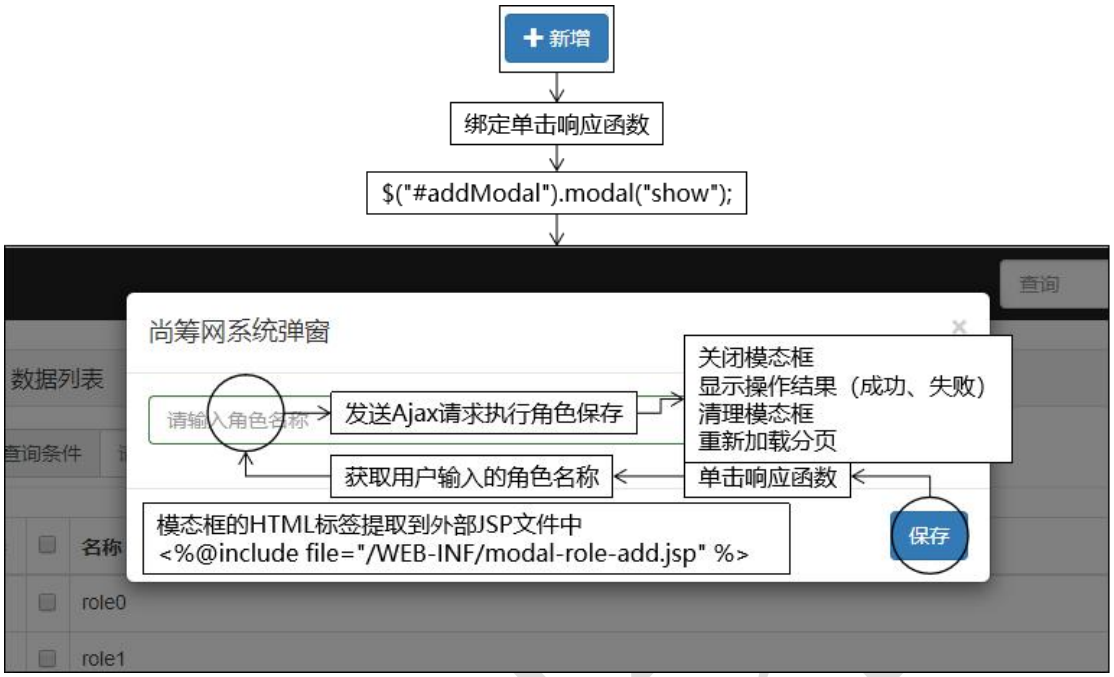
```
// 3.给查询按钮绑定单击响应函数
$("#searchBtn").click(function(){

    // ①获取关键词数据赋值给对应的全局变量
    window.keyword = $("#keywordInput").val();

    // ②调用分页函数刷新页面
    generatePage();

});
```

3 角色保存操作

3.1 目标

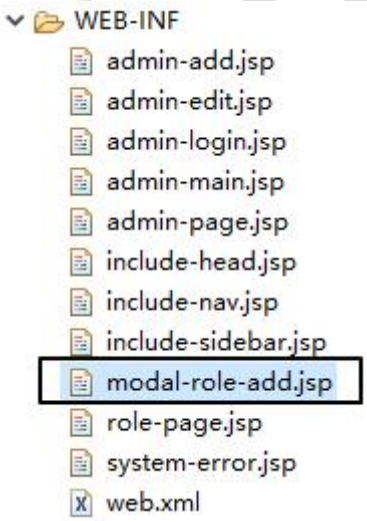
通过在打开的模态框中输入角色名称，执行对新角色的保存。

3.2 思路



3.3 代码：页面引入模态框

3.3.1 创建 JSP 文件



3.3.2 加入模态框 HTML 代码

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<div id="addModal" class="modal fade" tabindex="-1" role="dialog">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
```

```
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal"
                        aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                    <h4 class="modal-title">尚筹网系统弹窗</h4>
                </div>
                <div class="modal-body">
                    <form class="form-signin" role="form">
                        <div class="form-group has-success has-feedback">
                            <input
                                type="text" name="roleName"
                                class="form-control" id="inputSuccess4" placeholder="请输入
角色名称"
                                autofocus>
                        </div>
                    </form>
                </div>
                <div class="modal-footer">
                    <button id="saveRoleBtn" type="button" class="btn btn-primary"> 保 存
</button>
                </div>
            </div>
        </div>
</div>
```

3.3.3 在 role-page.jsp 引入上面的文件

```
<%@include file="/WEB-INF/modal-role-add.jsp" %>
```

模态框默认情况下是隐藏的，为了页面整洁，统一放在最后的位置。

3.4 代码：打开模态框

3.4.1 修改新增按钮

```
<button
    type="button"
    id="showAddModalBtn" class="btn btn-primary"
    style="float: right;">
    <i class="glyphicon glyphicon-plus"></i> 新增
</button>
```

去掉了原来的 onclick 属性

3.4.2 给新增按钮绑定单击响应函数

```
// 4.点击新增按钮打开模态框
$("#showAddModalBtn").click(function(){

    $("#addModal").modal("show");

});
```

3.5 代码：执行保存

3.5.1 前端代码

```
// 5.给新增模态框中的保存按钮绑定单击响应函数
$("#saveRoleBtn").click(function(){

    // ①获取用户在文本框中输入的角色名称
    // #addModal 表示找到整个模态框
    // 空格表示在后代元素中继续查找
    // [name=roleName]表示匹配 name 属性等于 roleName 的元素
    var roleName = $.trim($("#addModal [name=roleName]").val());

    // ②发送 Ajax 请求
    $.ajax({
        "url": "role/save.json",
        "type":"post",
        "data": {
            "name": roleName
        },
        "dataType": "json",
        "success":function(response){

            var result = response.result;

            if(result == "SUCCESS") {
                layer.msg("操作成功！");

                // 将页码定位到最后一页
                window.pageNum = 99999999;

                // 重新加载分页数据
                generatePage();
            }
```

```
                if(result == "FAILED") {
                    layer.msg("操作失败！ "+response.message);
                }

            },
            "error":function(response){
                layer.msg(response.status+" "+response.statusText);
            }
        });

        // 关闭模态框
        $("#addModal").modal("hide");

        // 清理模态框
        $("#addModal [name=roleName]").val("");
});
```

3.5.2 后端代码

```
// handler 代码
@ResponseBody
@RequestMapping("/role/save.json")
public ResultEntity<String> saveRole(Role role) {

    roleService.saveRole(role);

    return ResultEntity.successWithoutData();
}
```

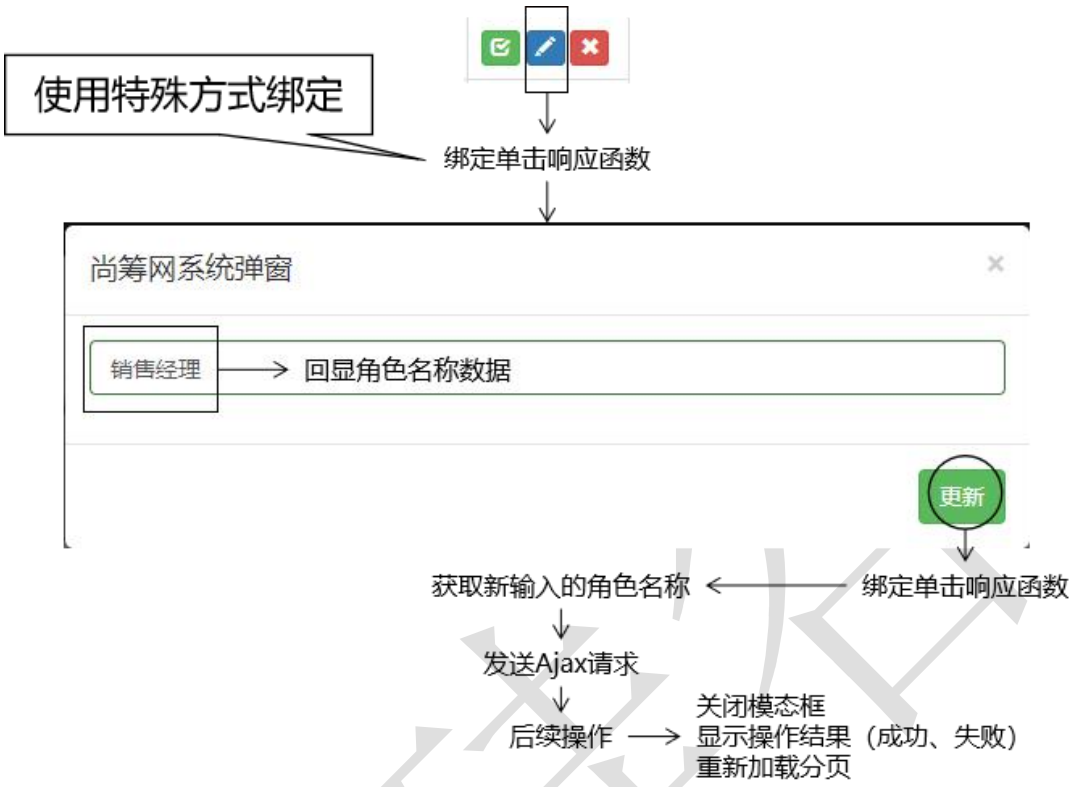
```
// service 代码
    @Override
    public void saveRole(Role role) {
        roleMapper.insert(role);
    }
```

4 角色更新操作

4.1 目标

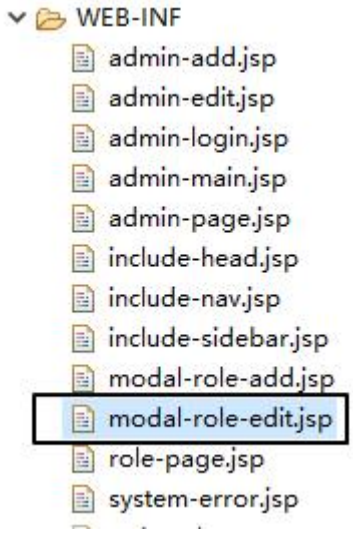
修改角色信息。

4.2 思路



4.3 代码：页面引入模态框

4.3.1 创建 JSP 文件



4.3.2 加入模态框的 HTML 代码

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<div id="editModal" class="modal fade" tabindex="-1" role="dialog">
```

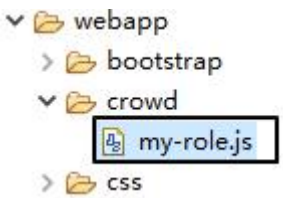
```
<div class="modal-dialog" role="document">
    <div class="modal-content">
        <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal"
                aria-label="Close">
                <span aria-hidden="true">&times;</span>
            </button>
            <h4 class="modal-title">尚筹网系统弹窗</h4>
        </div>
        <div class="modal-body">
            <form class="form-signin" role="form">
                <div class="form-group has-success has-feedback">
                    <input
                        type="text" name="roleName"
                        class="form-control" placeholder="请输入角色名称"
                        autofocus>
                </div>
            </form>
        </div>
        <div class="modal-footer">
            <button id="updateRoleBtn" type="button" class="btn btn-success">更新
</button>
        </div>
    </div>
</div>
</div>
```

4.3.3 在 role-page.jsp 引入

```
<%@include file="/WEB-INF/modal-role-edit.jsp" %>
```

4.4 代码：打开模态框（回显）

4.4.1 修改“铅笔”按钮



修改 fillTableBody()函数

```
// 通过 button 标签的 id 属性（别的属性其实也可以）把 roleId 值传递到 button 按钮的单击响应函数中，在单击响应函数中使用 this.id
var pencilBtn = "<button id='"+roleId+"' type='button' class='btn btn-primary btn-xs pencilBtn'><i class=' glyphicon glyphicon-pencil'></i></button>";
```

4.4.2 给“铅笔”按钮绑定单击响应函数

```
// 6.给页面上的“铅笔”按钮绑定单击响应函数，目的是打开模态框
// 传统的事件绑定方式只能在第一个页面有效，翻页后失效了
// $(".pencilBtn").click(function(){
//     alert("aaaa...");
// });

// 使用 jQuery 对象的 on()函数可以解决上面问题
// ①首先找到所有“动态生成”的元素所附着的“静态”元素
// ②on()函数的第一个参数是事件类型
// ③on()函数的第二个参数是找到真正要绑定事件的元素的选择器
// ③on()函数的第三个参数是事件的响应函数
$("#rolePageBody").on("click",".pencilBtn",function(){
    // 打开模态框
    $("#editModal").modal("show");

    // 获取表格中当前行中的角色名称
    var roleName = $(this).parent().prev().text();

    // 获取当前角色的 id
    // 依据是：var pencilBtn = "<button id='"+roleId+"' ……这段代码中我们把 roleId 设置到 id 属性了
    // 为了让执行更新的按钮能够获取到 roleId 的值，把它放在全局变量上
    window.roleId = this.id;

    // 使用 roleName 的值设置模态框中的文本框
    $("#editModal [name=roleName]").val(roleName);
});
```

4.5 代码：执行更新

4.4.1 前端

```
// 7.给更新模态框中的更新按钮绑定单击响应函数
$("#updateRoleBtn").click(function(){

```

```
    // ①从文本框中获取新的角色名称
    var roleName = $("#editModal [name=roleName]").val();

    // ②发送 Ajax 请求执行更新
    $.ajax({
        "url":"role/update.json",
        "type":"post",
        "data":{
            "id":window.roleId,
            "name":roleName
        },
        "dataType":"json",
        "success":function(response){

            var result = response.result;

            if(result == "SUCCESS") {
                layer.msg("操作成功！");

                // 重新加载分页数据
                generatePage();
            }

            if(result == "FAILED") {
                layer.msg("操作失败！"+response.message);
            }

        },
        "error":function(response){
            layer.msg(response.status+" "+response.statusText);
        }
    });

    // ③关闭模态框
    $("#editModal").modal("hide");
});
```

4.5.2 后端

```
// handler 代码
    @ResponseBody
    @RequestMapping("/role/update.json")
```

```
    public ResultEntity<String> updateRole(Role role) {

        roleService.updateRole(role);

        return ResultEntity.successWithoutData();
    }
```

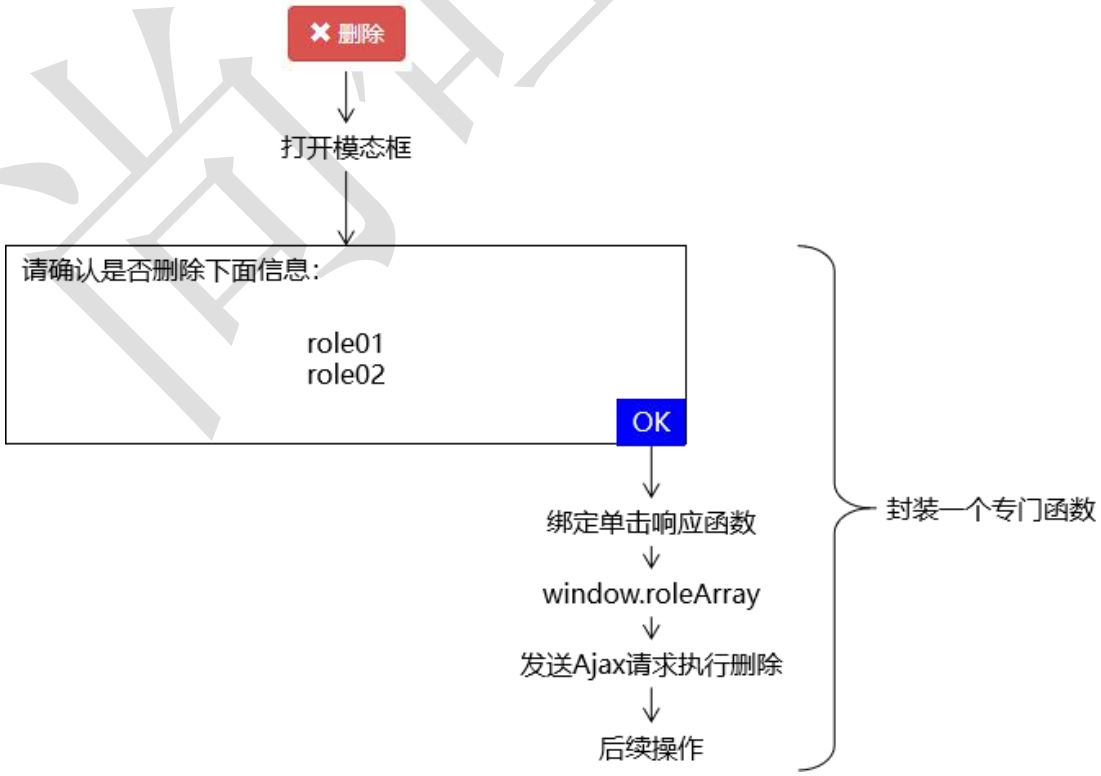
```
// service 代码
    @Override
    public void updateRole(Role role) {
        roleMapper.updateByPrimaryKey(role);
    }
```

5 角色删除操作

5.1 目标

前端的“单条删除”和“批量删除”在后端合并为同一套操作。合并的依据是：单条删除时 id 也放在数组中，后端完全根据 id 的数组进行删除。

5.2 思路



5.3