尚筹网

[05-后台管理系统-管理员信息维护]

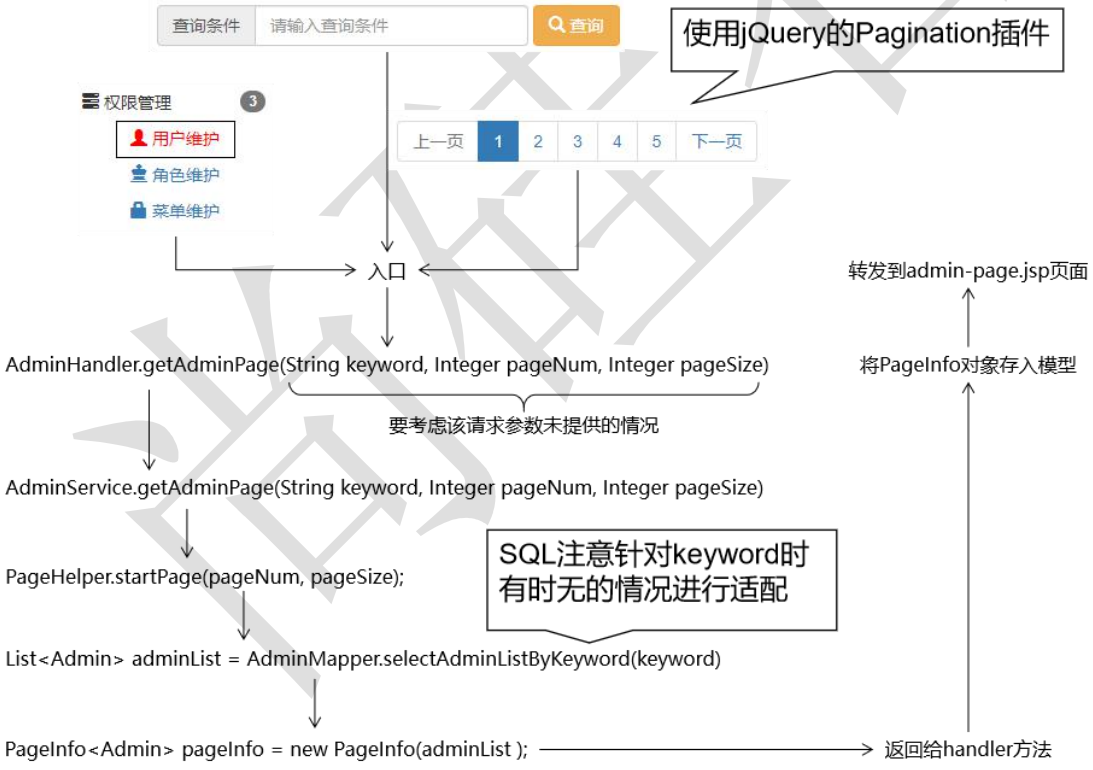
1 分页显示管理员信息部分

1.1 目标

以分页的形式把管理员信息显示到页面上。特殊需求：兼顾关键词查询，让后端代码不管有没有查询条件都能够以分页形式显示数据。

1.2 思路

1.2.1 流程图



1.2.2 技术点

- 让 SQL 语句针对 keyword 时有时无的情况进行适配
  使用 SQL 中做字符串连接的函数：CONCAT("%",#{keyword},"%")

  keyword 有值：“like %tom%”
  keyword 无值：“like %%”
- PageHelper 使用

- 引入依赖
- 在 SqlSessionFactoryBean 中配置 PageHelper
- 在 Java 代码中使用

  PageHelper.startPage(pageNum, pageSize)

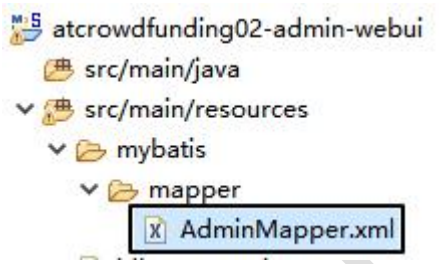
  PageInfo<Admin> pageInfo = new PageInfo(adminList);

➢ 显示页码

- 使用 jQuery 插件：Pagination

1.3 后端代码

1.3.1 查询 Admin 数据的 SQL 语句




```
<select id="selectAdminListByKeyword" resultMap="BaseResultMap">
    select
    <include refid="Base_Column_List" />
    from t_admin
    where login_acct like CONCAT("%",#{keyword},"%")
        or
        user_name like CONCAT("%",#{keyword},"%")
        or
        email like CONCAT("%",#{keyword},"%")
</select>
```

1.3.2 AdminMapper 中的抽象方法


```
List<Admin> selectAdminListByKeyword(String keyword);
```

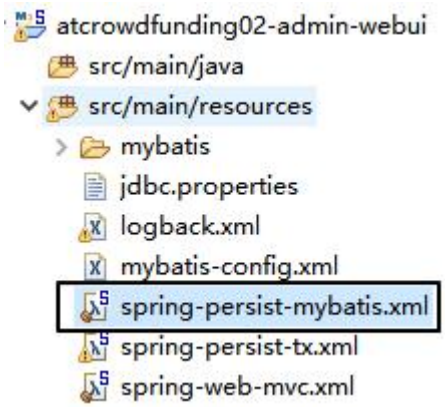
1.3.3 准备 PageHelper 环境

➢ 加入依赖

```
<!-- MyBatis 分页插件 -->
```

```
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
</dependency>
```

- 配置 SqlSessionFactoryBean




```
<!-- 配置 SqlSessionFactoryBean -->
<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 装配数据源 -->
    <property name="dataSource" ref="dataSource"/>

    <!-- 指定 MyBatis 全局配置文件位置 -->
    <property name="configLocation" value="classpath:mybatis-config.xml"/>

    <!-- 指定 Mapper 配置文件位置 -->
    <property name="mapperLocations" value="classpath:mybatis/mapper/*Mapper.xml"/>

    <!-- 配置 MyBatis 的插件 -->
    <property name="plugins">
        <array>
            <!-- 配置 PageHelper -->
            <bean class="com.github.pagehelper.PageHelper">
                <!-- 配置相关属性 -->
                <property name="properties">
                    <props>
                        <!-- 配置数据库方言，告诉 PageHelper 当前使用的具体数据库，
-->
                        <!-- 让 PageHelper 可以根据当前数据库生成对应的分页 SQL 语
句 -->
                        <prop key="dialect">mysql</prop>

                        <!-- 配置页码的合理化修正 -->
```

```
                        <!-- 让 PageHelper 自动把浏览器传来的 PageNum 修正到 0~总页数范围 -->
                        <prop key="reasonable">true</prop>
                    </props>
                </property>
            </bean>
        </array>
    </property>
</bean>
```

1.3.4 AdminService 方法

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

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

    // 2.查询 Admin 数据
    List<Admin> adminList = adminMapper.selectAdminListByKeyword(keyword);

    // ※辅助代码：打印 adminList 的全类名
    Logger logger = LoggerFactory.getLogger(AdminServiceImpl.class);

    logger.debug("adminList 的全类名是："+adminList.getClass().getName());

    // 3.为了方便页面使用将 adminList 封装为 PageInfo
    PageInfo<Admin> pageInfo = new PageInfo<>(adminList);

    return pageInfo;
}
```

1.3.5 AdminHandler 方法

```
@RequestMapping("/admin/page.html")
public String getAdminPage(

        // 注意：页面上有可能不提供关键词，要进行适配
        // 在@RequestParam 注解中设置 defaultValue 属性为空字符串表示浏览器不提供关键词时，keyword 变量赋值为空字符串
        @RequestParam(value="keyword", defaultValue="") String keyword,

        // 浏览器未提供 pageNum 时，默认前往第一页
```

```
            @RequestParam(value="pageNum", defaultValue="1") Integer pageNum,

            // 浏览器未提供 pageSize 时，默认每页显示 5 条记录
            @RequestParam(value="pageSize", defaultValue="5") Integer pageSize,

            ModelMap modelMap

        ) {

    // 查询得到分页数据
    PageInfo<Admin> pageInfo = adminService.getAdminPage(keyword, pageNum, pageSize);

    // 将分页数据存入模型
    modelMap.addAttribute(CrowdConstant.ATTR_NAME_PAGE_INFO, pageInfo);

    return "admin-page";
}
```

1.4 前端代码

1.4.1 抽取页面公共部分

- head 标签部分

  ```
  <%@ include file="/WEB-INF/include-header.jsp" %>
  ```

- nav 标签部分

  ```
  <%@ include file="/WEB-INF/include-nav.jsp" %>
  ```

- sidebar 部分

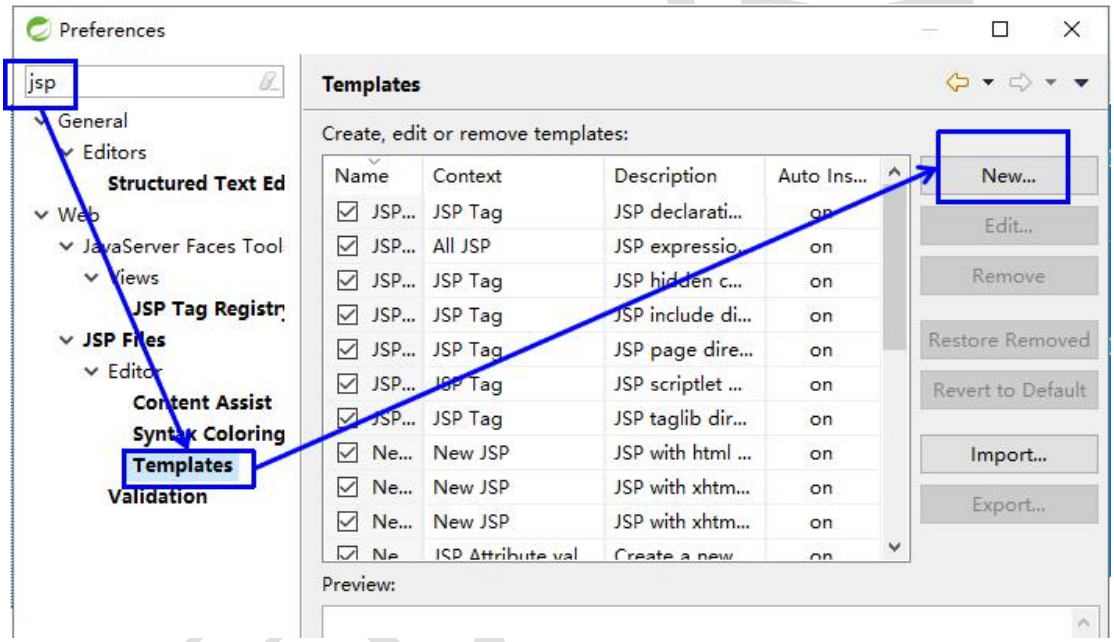
  ```
  <%@ include file="/WEB-INF/include-sidebar.jsp" %>
  ```

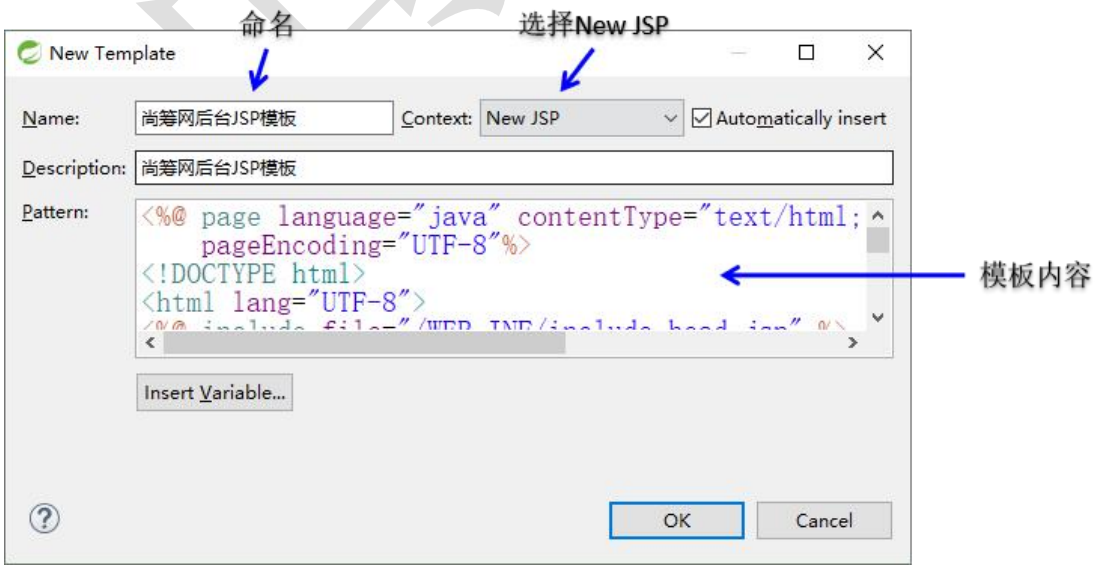
1.4.2 创建 JSP 模板

模板内容是：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="zh-CN">
<head>
<%@include file="/WEB-INF/include-header.jsp" %>
</head>
<body>

    <%@include file="/WEB-INF/include-nav.jsp" %>
```

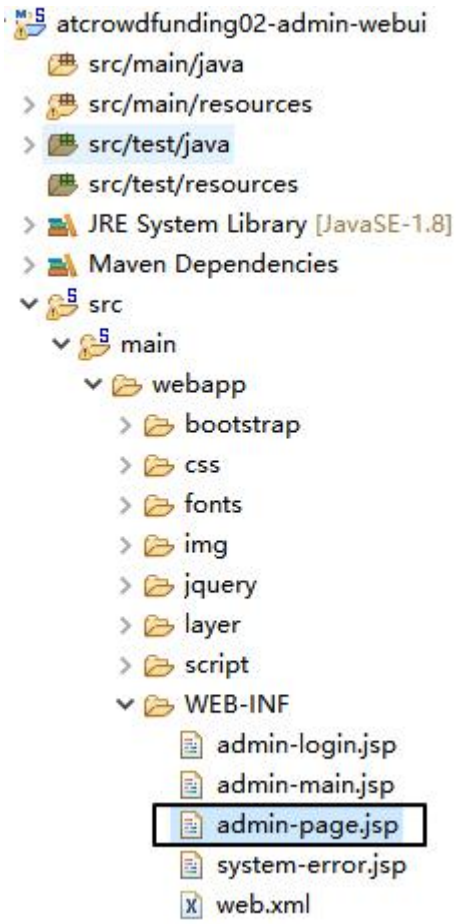
```
    <div class="container-fluid">
        <div class="row">
            <%@include file="/WEB-INF/include-sidebar.jsp" %>
            <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2 main">

            </div>
        </div>
    </div>
</body>
</html>
```





模板创建完成后，第一次创建 JSP 页面时需要选择一下我们创建的模板，以后就默认使用这个模板。

1.4.3 准备 admin-page.jsp



1.4.4 在 admin-page 页面显示真实数据

加入 JSTL 依赖

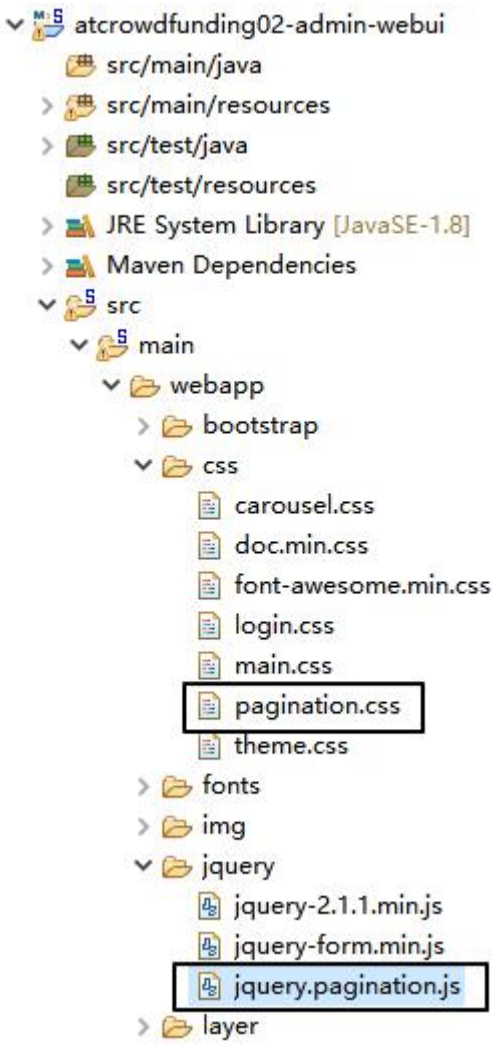
```
<!-- JSTL 标签库 -->
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

```
<tbody>
    <c:if test="${empty requestScope.pageInfo.list }">
        <tr>
            <td colspan="2">抱歉！ 没有查询到相关的数据！ </td>
        </tr>
    </c:if>
    <c:if test="${!empty requestScope.pageInfo.list }">
        <c:forEach items="${requestScope.pageInfo.list }" var="admin" varStatus="myStatus">
```

```
                <tr>
                    <td>${myStatus.count }</td>
                    <td><input type="checkbox"></td>
                    <td>${admin.loginAcct }</td>
                    <td>${admin.userName }</td>
                    <td>${admin.email }</td>
                    <td>
                        <button type="button" class="btn btn-success btn-xs">
                            <i class=" glyphicon glyphicon-check"></i>
                        </button>
                        <button type="button" class="btn btn-primary btn-xs">
                            <i class=" glyphicon glyphicon-pencil"></i>
                        </button>
                        <button type="button" class="btn btn-danger btn-xs">
                            <i class=" glyphicon glyphicon-remove"></i>
                        </button>
                    </td>
                </tr>
            </c:forEach>
        </c:if>
</tbody>
```

1.4.5 加入 Pagination 插件环境

- 解压 pagination_zh.zip 包
- 把相关文件加入到工程



- 在页面上引入样式文件和 JavaScript 文件

```
<link rel="stylesheet" href="css/pagination.css">
<script type="text/javascript" src="jquery/jquery.pagination.js"></script>
```

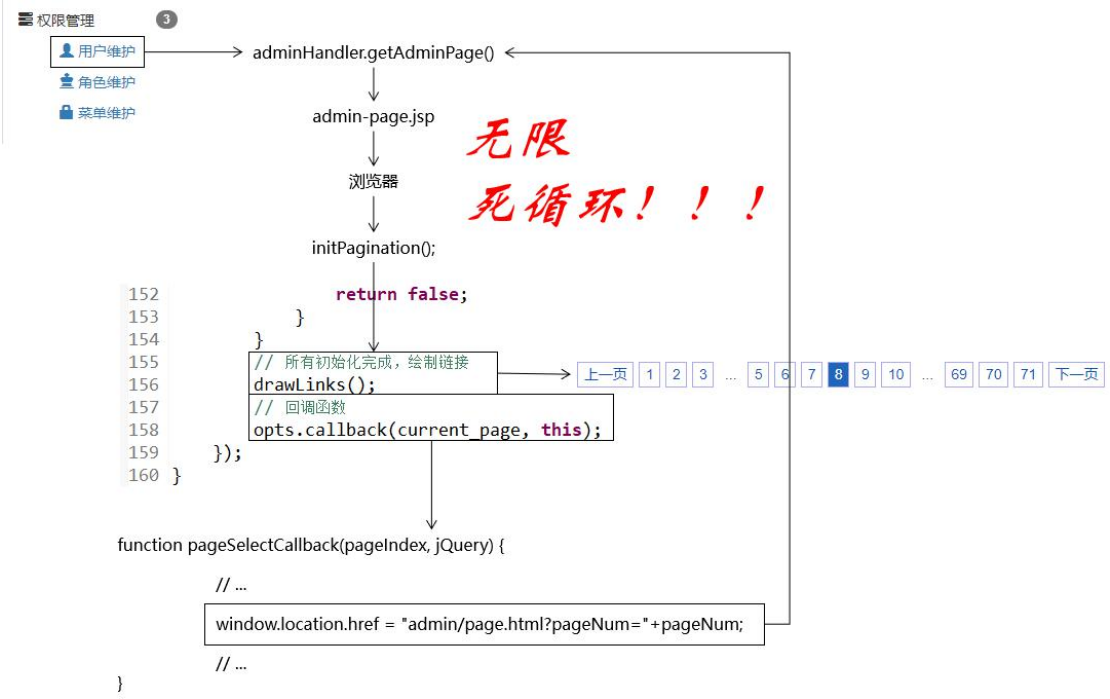
1.4.6 编写 Pagination 代码

```
$(function(){

    // 调用专门的函数初始化分页导航条
    initPagination();

});

// 声明一个函数用于初始化 Pagination
function initPagination() {
```

```
    // 获取分页数据中的总记录数
    var totalRecord = ${requestScope.pageInfo.total};

    // 声明 Pagination 设置属性的 JSON 对象
    var properties = {
        num_edge_entries: 3,                                    // 边缘页数
        num_display_entries: 5,                                 // 主体页数
        callback: pageSelectCallback,                           // 用户点击“翻页”按钮之后
执行翻页操作的回调函数
        current_page: ${requestScope.pageInfo.pageNum-1}, // 当前页，pageNum 从 1 开始，
必须-1 后才可以赋值
        prev_text: "上一页",
        next_text: "下一页",
        items_per_page:${requestScope.pageInfo.pageSize}  // 每页显示 1 项
    };

    // 调用分页导航条对应的 jQuery 对象的 pagination()方法生成导航条
    $("#Pagination").pagination(totalRecord, properties);

}

// 翻页过程中执行的回调函数
// 点击“上一页”、“下一页”或“数字页码”都会触发翻页动作，从而导致当前函数被调
用
// pageIndex 是用户在页面上点击的页码数值
function pageSelectCallback(pageIndex, jQuery) {
    // pageIndex 是当前页页码的索引，相对于 pageNum 来说，pageIndex 比 pageNum 小 1
    var pageNum = pageIndex + 1;

    // 执行页面跳转也就是实现“翻页”
    window.location.href = "admin/page.html?pageNum="+pageNum;

    // 取消当前超链接的默认行为
    return false;
}
```

1.4.7 修改 jquery.pagination.js 文件源码

问题分析如下图所示：



问题的解决：不让 Pagination 在页码导航条初始化完成时就调用回调函数！

```
            else {
                return false;
            }
        }
        // 所有初始化完成，绘制链接
        drawLinks();
        // 回调函数
        // opts.callback(current_page, this);
    });
}
```

1.4.8 准备测试数据

```
@Test
public void testSaveAdminMulti() {
    for(int i = 0; i < 352; i++) {
        adminMapper.insert(new Admin(null, "loginAcct"+i, "userPswd"+i, "userName"+i,
"email"+i+"@qq.com", null));
```

```
    }
}
```

2 关键词查询

2.1 页面调整待提交的表单

/atcrowdfunding02-admin-webui/src/main/webapp/WEB-INF/admin-page.jsp 文件中

调整前：

```
<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 class="form-control has-success" type="text"
                placeholder="请输入查询条件">
        </div>
    </div>
    <button type="button" class="btn btn-warning">
        <i class="glyphicon glyphicon-search"></i> 查询
    </button>
</form>
```

调整后：

```
<form action="admin/page.html" method="post" 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 name="keyword" class="form-control has-success" type="text"
                placeholder="请输入查询条件">
        </div>
    </div>
    <button type="submit" class="btn btn-warning">
        <i class="glyphicon glyphicon-search"></i> 查询
    </button>
</form>
```

2.2 翻页时保持 keyword 值

```
window.location.href =
"admin/page.html?pageNum="+pageNum+"&keyword=${param.keyword}";
```

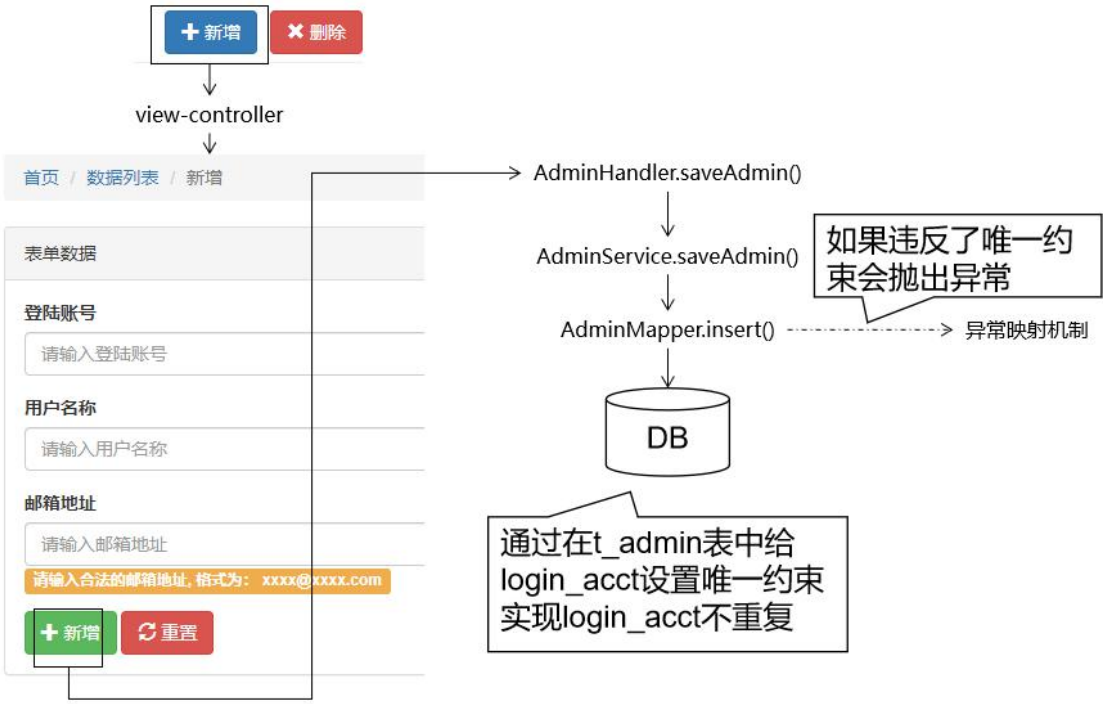
EL 表达式中的 param 也是一个隐含对象，可以用来获取请求参数！

3 新增管理员信息

3.1 目标

创建新的管理员信息，通过表单提交给后端程序并保存到数据库。

3.2 思路

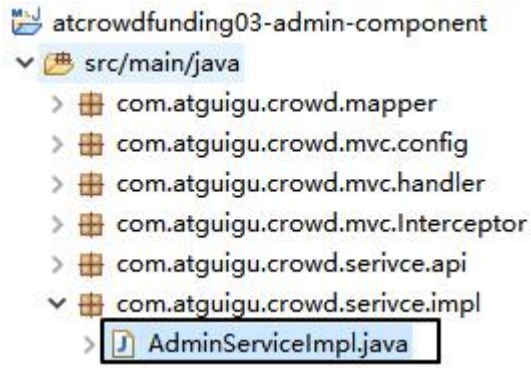


3.3 代码

3.3.1 设置唯一约束

```
ALTER TABLE `t_admin`
ADD UNIQUE INDEX (`login_acct`)
```

3.3.2 业务逻辑层




```
@Override
public void saveAdmin(Admin admin) {

    // 生成当前系统时间
    Date date = new Date();
    SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    String createTime = format.format(date);

    admin.setCreateTime(createTime);

    // 针对登录密码进行加密
    String source = admin.getUserPswd();
    String encoded = CrowdUtil.md5(source);

    admin.setUserPswd(encoded);

    // 执行保存，如果账户被占用会抛出异常
    try {
        adminMapper.insert(admin);
    } catch (Exception e) {
        e.printStackTrace();

        // 检测当前捕获的异常对象，如果是 DuplicateKeyException 类型说明是账号重复导致的
        if(e instanceof DuplicateKeyException) {

            // 抛出自定义的 LoginAcctAlreadyInUseException 异常
            throw new LoginAcctAlreadyInUseException(CrowdConstant.MESSAGE_LOGIN_ACCT_ALREADY_IN_USE);

        }
```

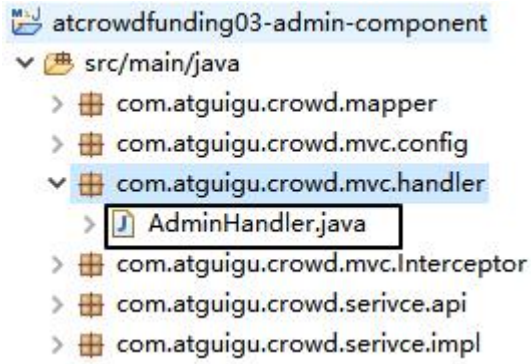
```
            // 为了不掩盖问题，如果当前捕获到的不是 DuplicateKeyException 类型的异常，则把当前捕获到的异常对象继续向上抛出
            throw e;
        }

}
```

3.3.3 异常映射处理器类


```
@ExceptionHandler(value = LoginAcctAlreadyInUseException.class)
public ModelAndView resolveLoginAcctAlreadyInUseException(
            LoginAcctAlreadyInUseException exception,
            HttpServletRequest request,
            HttpServletResponse response
        ) throws IOException {

    // 只是指定当前异常对应的页面即可
    String viewName = "admin-add";

    return commonResolveException(exception, request, response, viewName);
}
```

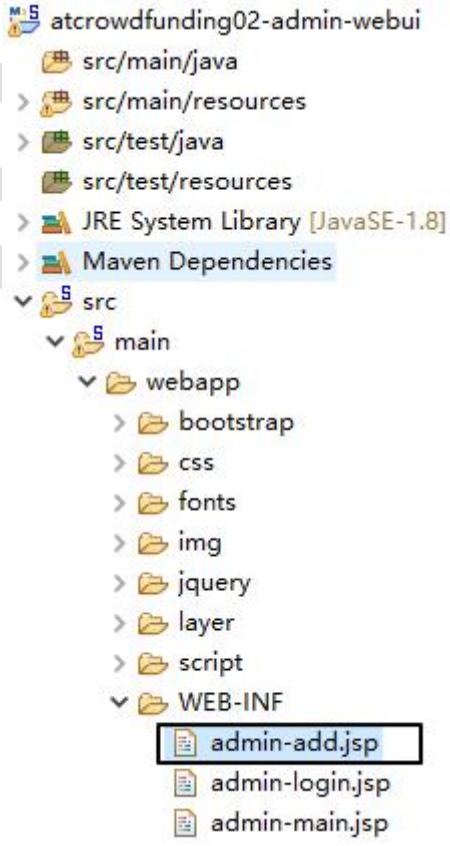
3.3.4 自定义异常

3.3.5 控制层



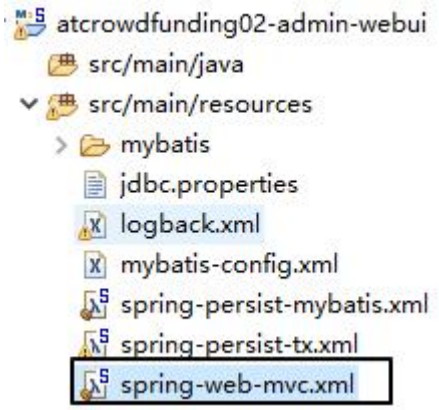

```
@RequestMapping("/save/admin.html")
public String saveAdmin(Admin admin) {

    // 执行保存
    adminService.saveAdmin(admin);

    // 重定向到分页页面，使用重定向是为了避免刷新浏览器重复提交表单
    return "redirect:/admin/page.html?pageNum="+Integer.MAX_VALUE;
}
```

3.3.6 准备表单页面

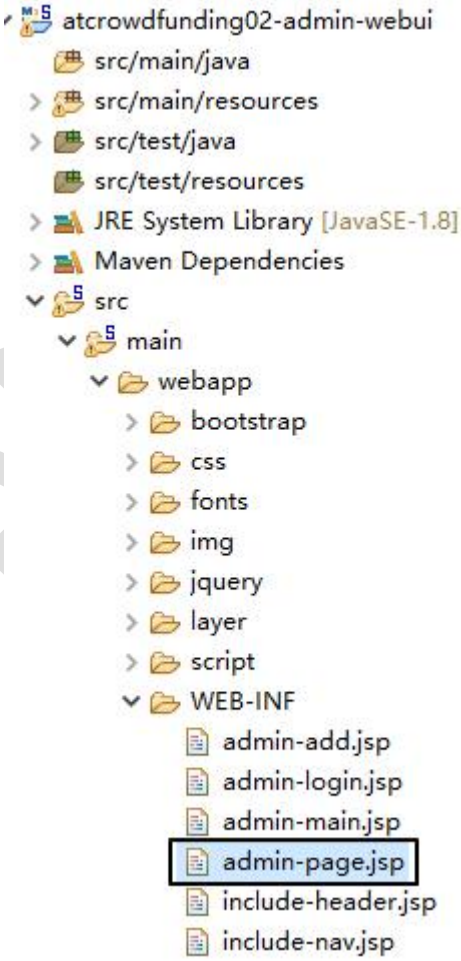


细节请参考代码中的这个文件！

3.3.7 跳转到这个页面




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



修改前：

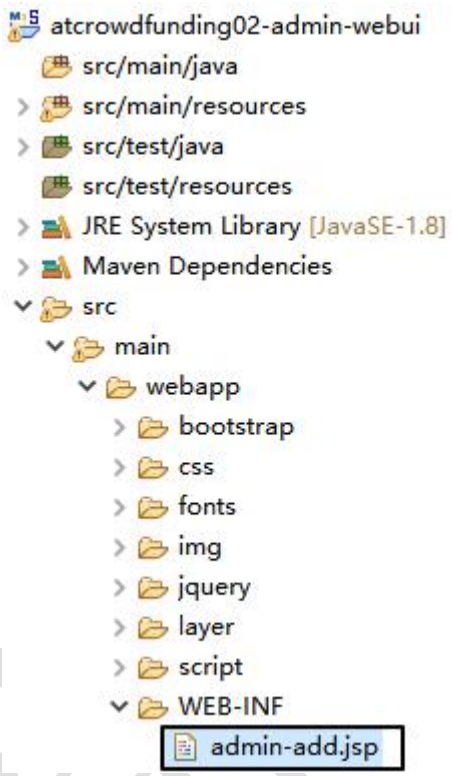
```
<button type="button" class="btn btn-primary"
      style="float: right;" onclick="window.location.href='add.html'">
```

```
    <i class="glyphicon glyphicon-plus"></i> 新增
</button>
```

修改后：

```
<a href="admin/to/add/page.html" class="btn btn-primary" style="float: right;"><i class="glyphicon glyphicon-plus"></i> 新增</a>
```

3.3.8 页面上修改表单



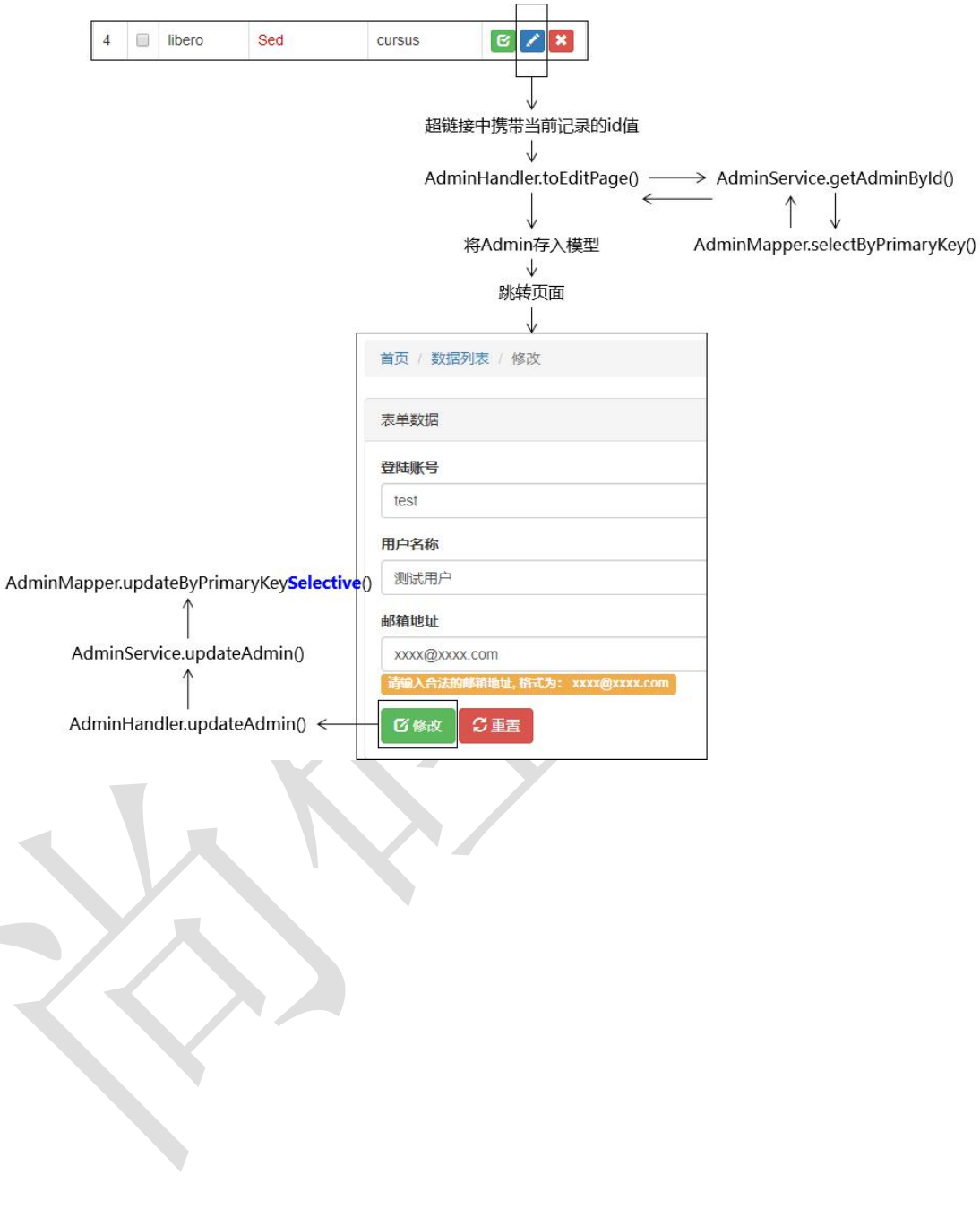
细节参考代码中的对应文件!!!

4 更新管理员信息

4.1 目标

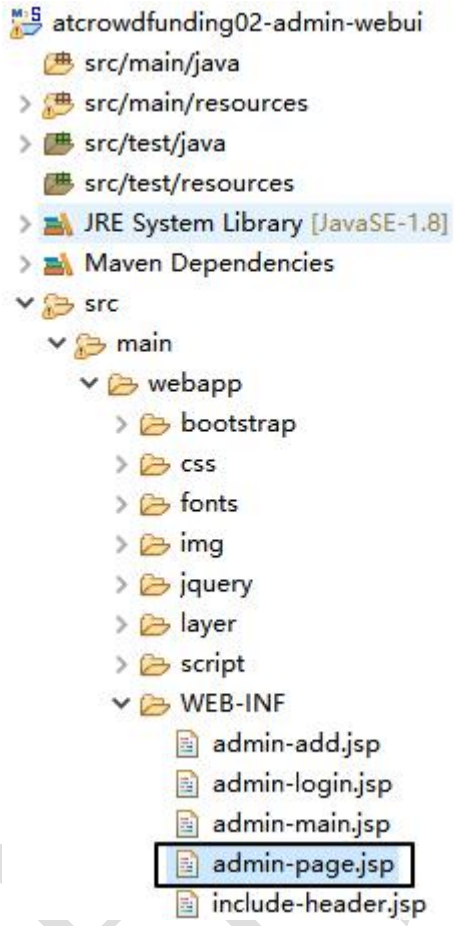
通过提交页面上的表单修改某个 Admin 的数据。

4.2 思路



4.3 代码：前往更新页面

4.3.1 准备“更新”超链接



修改前：

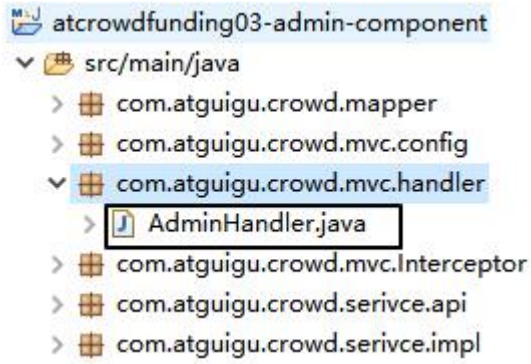
```
<button type="button" class="btn btn-primary btn-xs">
    <i class=" glyphicon glyphicon-pencil"></i>
</button>
```

修改后：

```
<a href="admin/to/edit/page.html?adminId=${admin.id }" class="btn btn-primary btn-xs"><i class=" glyphicon glyphicon-pencil"></i></a>
```

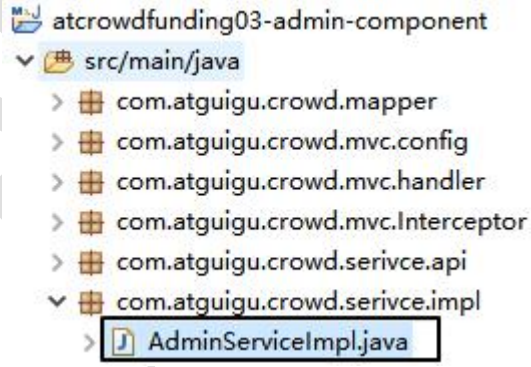
注意：按钮代码所在的位置是 c:forEach 标签中！

4.3.2 handler 代码




```
@RequestMapping("/admin/to/edit/page.html")
public String toEditPage(@RequestParam("adminId") Integer adminId, ModelMap modelMap) {

    // 1.根据 id（主键）查询待更新的 Admin 对象
    Admin admin = adminService.getAdminById(adminId);

    // 2.将 Admin 对象存入模型
    modelMap.addAttribute("admin", admin);

    return "admin-edit";
}
```

4.3.3 service 代码




```
@Override
public Admin getAdminById(Integer adminId) {
    return adminMapper.selectByPrimaryKey(adminId);
}
```

4.3.4

5 删除管理员信息