目录

[一、项目介绍 2](#_Toc536096656)

[1、项目的导入(以IDEA为例,Eclipse类似) 2](#_Toc536096657)

[2、项目结构介绍 3](#_Toc536096658)

[3、部署项目 4](#_Toc536096659)

[二、开发功能 5](#_Toc536096660)

[1、在线代码生成 5](#_Toc536096661)

[2、添加任务计划 6](#_Toc536096662)

[3、添加过滤器 7](#_Toc536096663)

[4、修改认证路径 8](#_Toc536096664)

[5、添加访问日志 8](#_Toc536096665)

[6、列表页的开发(基于bootstrap) 8](#_Toc536096666)

[7、基于异步请求方式的列表展示 9](#_Toc536096667)

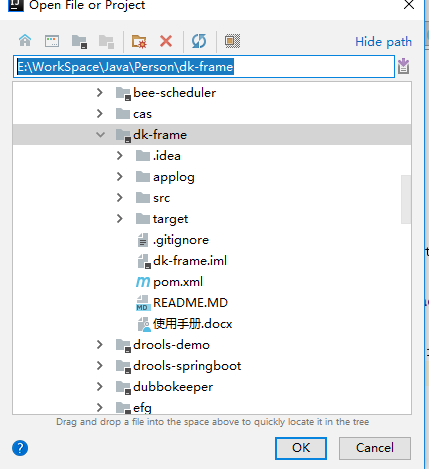
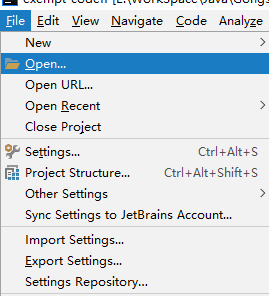
[8、给按钮添加权限 11](#_Toc536096668)

[9、列表内功能权限的控制 12](#_Toc536096669)

# 一、项目介绍

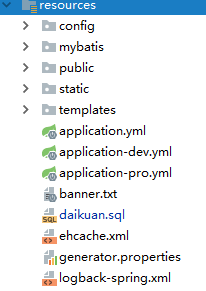
本项目以主流java技术为框架，搭建的一个简单易用的开发框架，用的技术有: Maven、JDK8、SpringBoot2.0、Shiro、Mybatis、Thymeleaf、Bootstrap、Vue、Jquery，使用的第三方中间件有: Redis、Ehcache，支持代码的一键生成…现已实现用户管理、角色管理、权限管理、部门管理、菜单管理、公告通知、数据字典、在线用户、图表展示等功能，系统方面有：系统日志、系统监控、接口测试、计划任务、SQL监控等功能，满足日常的基本管理系统的开发需求

## 1、项目的导入(以IDEA为例,Eclipse类似)



## 2、项目结构介绍





**config**: ehcach配置、quartz配置

**mybatis**: mybatis访问数据的sql，按照模块进行分目录，blog、common、oa、system

**public**: 公共资源

**static**: 静态资源

**templates**: 网页模板，同样按照模块进行分目录，blog、common、oa、system

**application.yml**：项目通用配置，端口、session、mybatis、页面模板

**application-\*.xml**：数据库、redis、日志等的配置

**dev**: 开发环境

**pro**: 生产环境

**daikuan.sql**: 默认的数据表结构及初始化数据

**ehcache.xml**: ehcache缓存的配置

**logback-spring.xml**：日志路径配置

## 3、部署项目

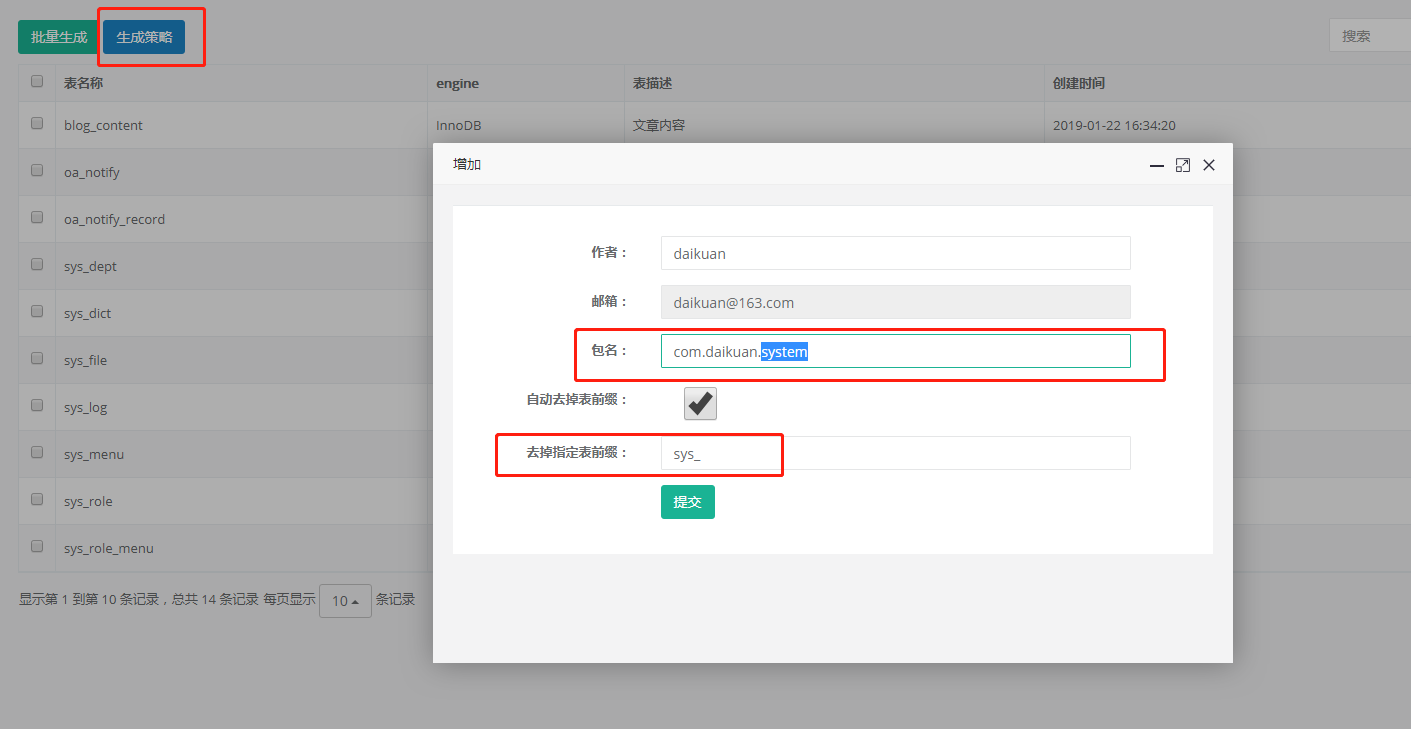
在pom.xml的同级目录下

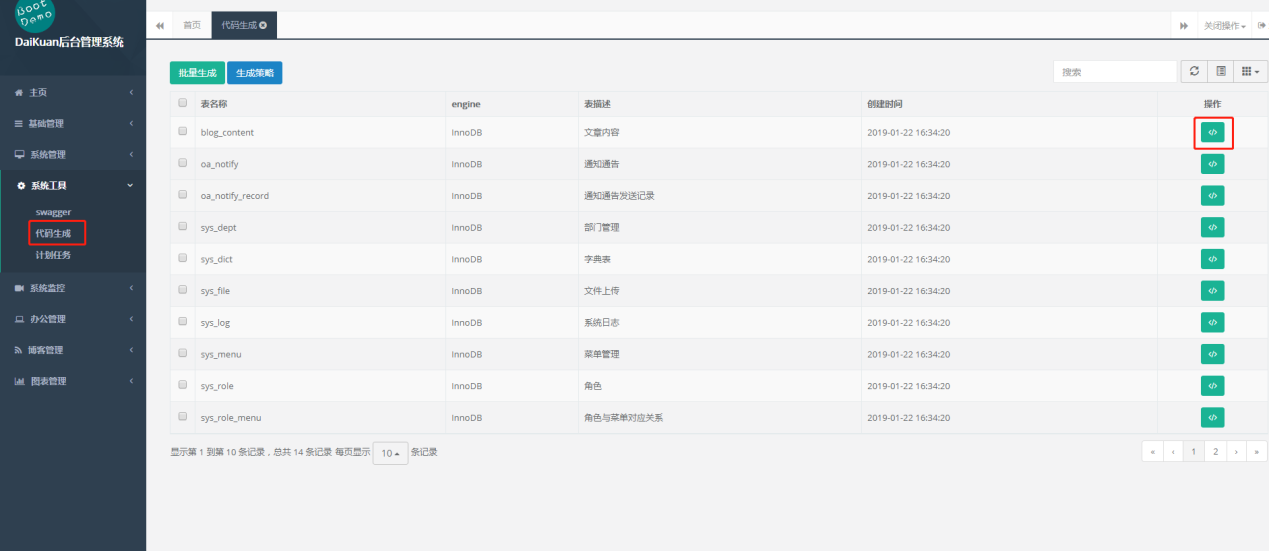
1. #执行daikuan.sql /mysql -u账号 -p密码 -D数据库名 < sql文件绝对路径
2. mysql -uroot -p123456 -Dtest<./daikuan.sql
3. ## 打包，跳过测诿/span>
4. mvn clean **package** -DskipTests=**true**
6. #启动项目 --server.port指定启动端口  --spring.profiles.active激活运行环境¿/span>
7. #nohup java -jar \*.jar --server.port=端口 --spring.profiles.active=激活环境¿gt;catalina.log &
8. nohup java -jar dk-frame-1.0.0-SNAPSHOT.jar --server.port=8080 --spring.profiles.active=dev >catalina.log &

# 二、开发功能

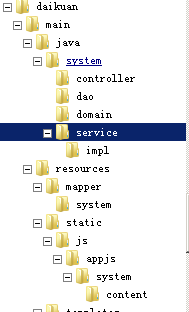
## 1、在线代码生成

首先在数据库里面创建需要的标，登录系统，找到“系统工具>代码生成”，点击生成策略，修改生成后的包名，选择刚创建的标，点击做右侧的生成按钮即可。





生成完后的代码是以jar包的形式下载，目录如下：

，按生成的目录结构导入的项目中即可。

批量生成，勾选需要生成的代码的表，点击左上角的“批量生成”，下载导入代码即可。

**注意：此功能切勿在生成环境使用，影响性能，可以把此功能的权限给屏蔽掉。**

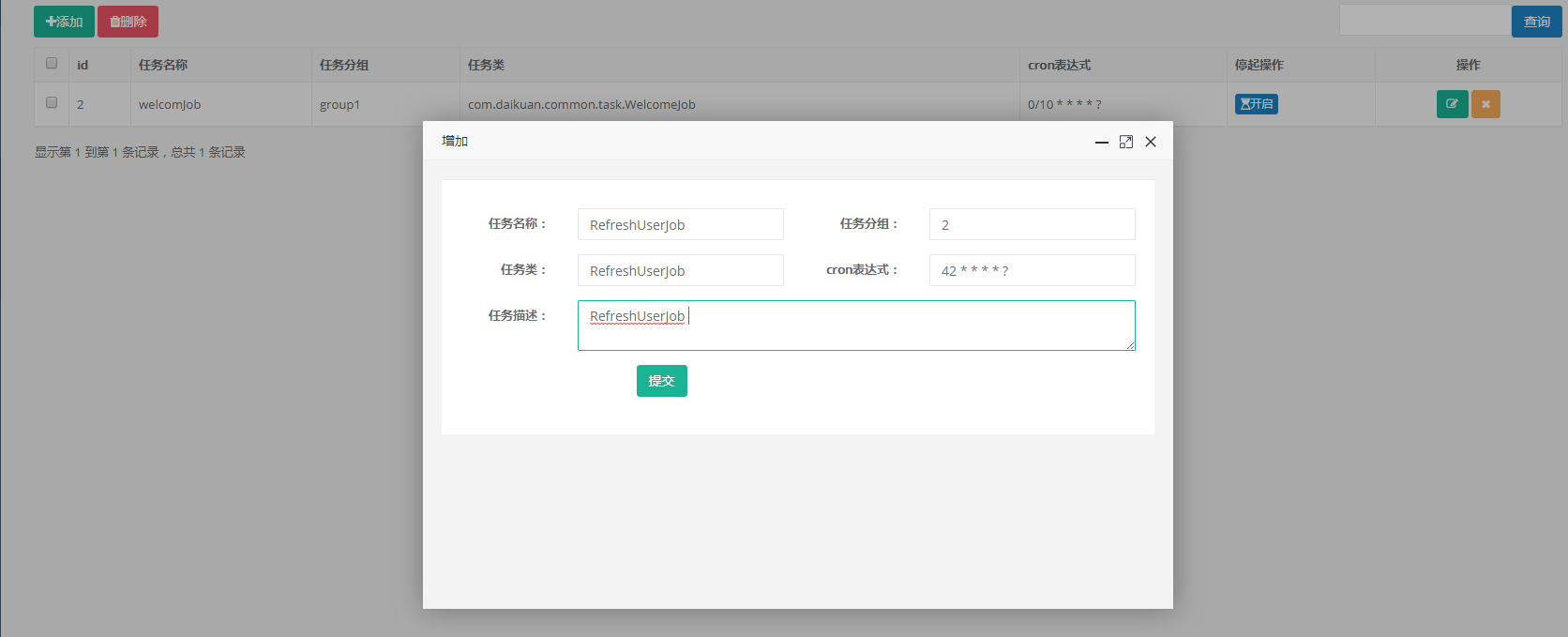
## 2、添加任务计划

任务计划，即是定时任务，在**com.daikuan.common.task**下开发

首先新的定时任务实现**org.quartz**包下**Job**接口，实现public void execute(JobExecutionContext context)方法，在实现类上添加**@Component**注解，在注入所需要的接口接口，eg:

1. @Component
2. **public** **class** RefreshUserJob **implements** Job {
4. @Autowired
5. **private** UserService userService;
7. @Override
8. **public** **void** execute(JobExecutionContext context) **throws** JobExecutionException {
9. Set<String> set = userService.listRoles(111L);
11. }
12. }

然后在控制台上添加此任务，“系统工具>计划任务”



1、任务名称，及Job的名称

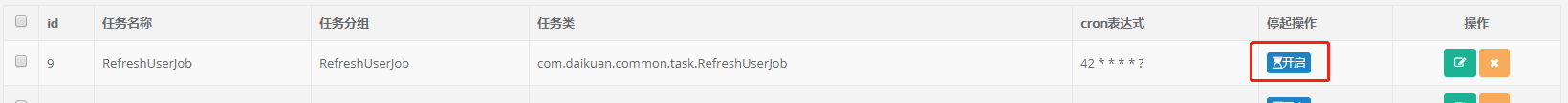
2、任务分组：任务所属的组

3、任务类：添加任务的类的全路径，新添加的那个任务为：com.daikuan.common.task.RefreshUserJob

4、cron表达式：任务的执行时点，可以在此配置生成: http://cron.qqe2.com/

5、任务描述

6、提交保存后的任务默认是关闭的，则需要手动开启



**注意：目前的部署方式不支持任务的集群，如果需要修改为集群模式，在com.daikuan.common.quartz下做相应的修改。**

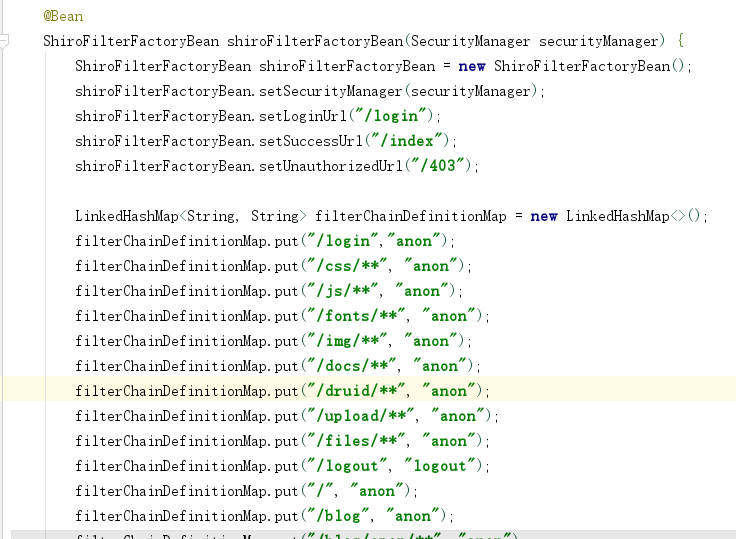
## 3、添加过滤器

首先实现**javax.servlet**包下的**Filter**，重写Filter的方法，在**doFilter**中添加需要处理的业务，然后配置新增的Filter，eg:

1. @Configuration  //这个很重要
2. **public** **class** XssConfig{
3. @Bean  //这个也能很重要
4. **public** FilterRegistrationBean xssFilterRegistrationBean() {
5. FilterRegistrationBean filterRegistrationBean = **new** FilterRegistrationBean();
6. filterRegistrationBean.setFilter(**new** XssFilter());  //配置创建的Filter
7. filterRegistrationBean.setOrder(1);
8. filterRegistrationBean.setEnabled(**false**);
9. filterRegistrationBean.addUrlPatterns("/\*");  //拦截的路径
10. Map<String, String> initParameters = Maps.newHashMap();
11. initParameters.put("excludes", "/favicon.ico,/img/\*,/js/\*,/css/\*");  //不拦截的
12. initParameters.put("isIncludeRichText", "true");
13. filterRegistrationBean.setInitParameters(initParameters);
14. **return** filterRegistrationBean;
15. }
16. }

## 4、修改认证路径

比如新增一个/news开头的请求路径，是允许不登录即可访问的，则需要在com.daikuan.system.config.ShiroConfig里面进行修改，



anon : 表示匿名可以访问，即没有登录就可以访问，/a/\*\*

authc：表示需要认证后才能访问，/a/\*\*\*

## 5、添加访问日志

在需要记录访问日志的的方法上添加**@Log(名称)** 即可

## 6、列表页的开发(基于bootstrap)

首先在相应的controller里面定义一个方法，方法有个Model的参数，在方法里查询需要展示的数据，把查询结果绑定到Model上，放回参数为，目标页面的名称，eg:

1. @GetMapping({"/index"})
2. String index(Model model) {
3. List<Tree<MenuDO>> menus = menuService.listMenuTree(getUserId());
4. model.addAttribute("menus", menus);
5. **return** "index\_v1";
6. }

在页面上使用**th:each** 进行遍历，如目录的展示,eg:

1. <li th:each="menu : ${menus}"><a href="#"><i **class**="fa fa fa-bar-chart-o" ></i>
2. <span **class**="nav-label" th:text="${menu.text}">基础信息</span><span
3. **class**="fa arrow"></span>
4. </a>
5. <ul **class**="nav nav-second-level">
6. <li th:each="cmenu : ${menu.children}">
7. <a **class**="J\_menuItem" href="graph\_echarts.html"  th:text="${cmenu.text}"
8. th:href="${cmenu.attributes.url}">系统管理</a></li>
9. </ul>
10. </li>

## 7、基于异步请求方式的列表展示

首先定义一个方法，打开这个页面，eg:

@GetMapping(**""**)  
String user(Model model) {  
**return prefix** + **"/user"**;  
}

在定义一个返回列表的方法，eg:

1. @GetMapping("/list")
2. @ResponseBody
3. PageUtils list(@RequestParam Map<String, Object> params) {
4. // 查询列表数据
5. Query query = **new** Query(params);
6. List<UserDO> sysUserList = userService.list(query);
7. **int** total = userService.count(query);
8. PageUtils pageUtil = **new** PageUtils(sysUserList, total);
9. **return** pageUtil;
10. }

一定要返回**PageUtils**对象

然后在页面上定义带id的table标签，eg:

<**table id="exampleTable" data-mobile-responsive="true"**>  
</**table**>

在这个页面对应的js上，添加请求的方法，方法中使用页面上定义好的table,eg：

1. $('#exampleTable')  //页面定义的id
2. .bootstrapTable(
3. {
4. method : 'get', // 服务器数据的请求方式 get or post
5. url : prefix + "/list", // 服务器数据的加载地址
6. iconSize : 'outline',
7. toolbar : '#exampleToolbar',
8. striped : **true**, // 设置为true会有隔行变色效果
9. dataType : "json", // 服务器返回的数据类型
10. pagination : **true**, // 设置为true会在底部显示分页板/span>
11. // //设置为limit则会发送符合RESTFull格式的参敿/span>
12. singleSelect : **false**, // 设置为true将禁止多逿/span>
13. pageSize : 10, // 如果设置了分页，每页数据条数
14. pageNumber : 1, // 如果设置了分布，首页页码
15. showColumns : **false**, // 是否显示内容下拉框（选择显示的列＿/span>
16. sidePagination : "server", // 设置在哪里进行分页，可选值为"client" 或耿/span>
17. // "server"
18. queryParams : function(params) {
19. **return** {
20. // 说明：传入后台的参数包括offset开始索引，limit步长，sort排序列，order：desc或耿以及所有列的键值对，分页查询使用的参数
21. limit : params.limit,
22. offset : params.offset,
23. name : $('#searchName').val(),
24. deptId : deptId
25. };
26. },
27. // 返回false将会终止请求
28. columns : [
29. {
30. checkbox : **true**
31. },
32. {
33. field : 'userId', // 列字段名
34. title : '序号' // 列标颿/span>
35. },
36. {
37. field : 'name',
38. title : '姓名'
39. },
40. {
41. field : 'username',
42. title : '用户吿
43. },
44. {
45. field : 'status',
46. title : '状怿,
47. align : 'center',
48. formatter : function(value, row, index) {
49. **if** (value == '0') {
50. **return** '<span class="label label-danger">禁用</span>';
51. } **else** **if** (value == '1') {
52. **return** '<span class="label label-primary">正常</span>';
53. }
54. }
55. },
56. {
57. title : '操作',
58. field : 'id',
59. align : 'center'
60. } ]
61. });

## 8、给按钮添加权限



列表页面可以通过权限管理来进行，列表是否展示的权限控制，如果是按钮的话，得需要在代码中添加相应的代码，在配合权限控制才能生效，给按钮添加权限控制，需要在页面引入**xmlns:shiro="http://www.pollix.at/thymeleaf/shiro"**配合权限码使用，eg:

添加用户按钮的权限控制：

1. <button shiro:hasPermission="sys:user:add" type="button"
2. **class**="btn  btn-primary" onclick="add()">
3. <i **class**="fa fa-plus hidden" aria-hidden="true"></i>添加
4. </button>

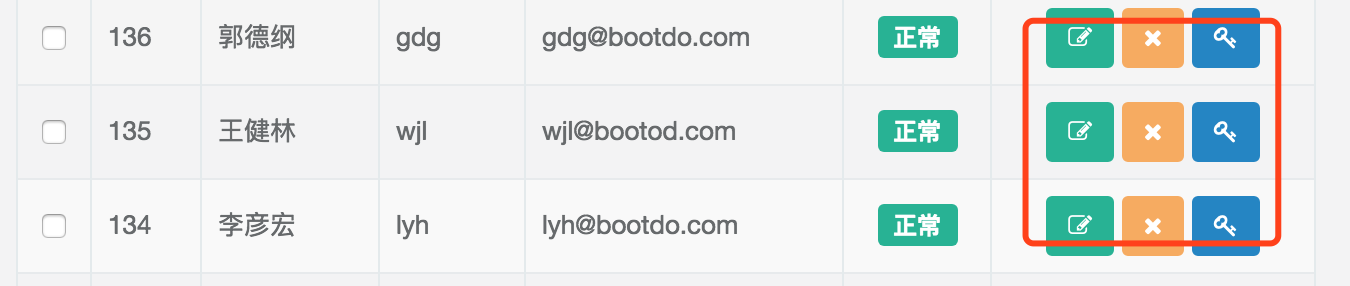
为了防止篡改页面，安全起见，需要在对应的权限功能上也添加相关的权限码

@RequiresPermissions(**"sys:user:add"**)  
@Log(**"添加用户"**)  
@GetMapping(**"/add"**)  
String add(Model model){}

**sys:user:add**称为权限码，在添加权限时输入的编码，全局唯一性

格式为：模块:角色:功能组成的，易懂，又不宜太长

## 9、列表内功能权限的控制



因为shiro没有对应的js标签，所以对表格内的按钮进行控制时稍微有些复杂，有两种方式，

第一种是在页面内渲染table，即把7中的代码写到html页面中，此时需要引入然后利用利用<**shiro:hasPermission** ></**shiro:hasPermission**>标签进行控制，受权限控制的按钮写到这个标签内eg:

部分代码

1. <shiro:hasPermission="sys:user:edit">
2. var e = '<a  class="btn btn-primary btn-sm ' + s\_edit\_h + '" href="#" mce\_href="#" title="编辑" onclick="edit(\''
3. + row.userId
4. + '\')"><i class="fa fa-edit "></i></a> ';
5. </shiro:hasPermission>

这样写的好处是，实现容易，但**js会大量出现在Html页面中，单个文件过大，不易维护**。

还有另外一种方式，在列表页面内定义一个公共变量，eg:

1. <script type="text/javascript">
2. var s\_edit\_h = 'hidden';
3. </script>

然后在根据权限给s\_edit\_b进行赋值 eg:

1. <div shiro:hasPermission="sys:user:edit">
2. <script type="text/javascript">
3. s\_edit\_h = '';
4. </script>
5. </div>

在渲染列表时，使用s\_edit\_b的值，进行控制显示或隐藏，eg:

1. formatter : function(value, row, index) {
2. var e = '<a  class="btn btn-primary btn-sm ' + s\_edit\_h + '" href="#" mce\_href="#" title="编辑" onclick="edit(\''
3. + row.userId
4. + '\')"><i class="fa fa-edit "></i></a> ';
5. **return** e;
6. }

因为如果当前用户没有权限时，就会使用默认的s\_edit\_b=hidden，对按钮进行隐藏，如果有权限，s\_edit\_b的值就为空，也就可以显示出来了。

**建议采用第二种方式**