

web综合案例

学习目标

- 目标1：完成用户与角色的绑定功能
- 目标2：完成登陆功能的快速开发
- 目标3：完成登陆用户菜单控制的功能
- 目标4：完成登陆用户权限校验的功能

1. 用户与角色

1.1 绑定用户与角色关系数据准备

(1) 在用户模块下的 `\WEB-INF\pages\system\user\list.jsp` 页面中找到 `角色` 按钮，对应的点击事件和函数的绑定都已完成，`roleList` 方法中向后台 `UserController` 发送请求，执行 `userRoleList` 方法，传递参数用户id

(2) 找到 `UserController`，添加方法

```
1  @Override
2  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
   ServletException, IOException {
3      String operation = request.getParameter("operation");
4      if("list".equals(operation)){
5          this.list(request,response);
6      }
7      //中间的省略
8      else if("userRoleList".equals(operation)){
9          this.userRoleList(request,response);
10     }
11 }
12 private void userRoleList(HttpServletRequest request, HttpServletResponse response) throws
   ServletException, IOException {
13     String userId = request.getParameter("id");
14     User user = userService.findById(userId);
15     //将数据加载到指定区域，供页面获取
16     request.setAttribute("user,user");
17     //获取所有的角色列表
18     List<Role> all = roleService.findAllRoleByUserId(userId);
19     request.setAttribute("roleList",all);
20     //跳转页面
21     request.getRequestDispatcher("/WEB-
INF/pages/system/user/role.jsp").forward(request,response);
22 }
```

同时从 `day03` 的课程资料中找到：`模块页面\user\role.jsp`，拷贝到工程 `/WEB-INF/pages/system/user/` 目录下

(3) 在 `RoleService` 接口中添加查询方法，根据用户id查询角色列表

```
1 List<Role> findAllRoleByUserId(String userId);
```

(4) 在对应的实现类中去实现该方法

```
1 @Override
2 public List<Role> findAllRoleByUserId(String userId) {
3     SqlSession sqlSession = null;
4     try{
5         //1.获取SqlSession
6         sqlSession = MapperFactory.getSqlSession();
7         //2.获取Dao
8         RoleDao roleDao = MapperFactory.getMapper(sqlSession,RoleDao.class);
9         //3.调用Dao层操作
10        return roleDao.findAllRoleByUserId(userId);
11    }catch (Exception e){
12        throw new RuntimeException(e);
13        //记录日志
14    }finally {
15        try {
16            TransactionUtil.close(sqlSession);
17        }catch (Exception e){
18            e.printStackTrace();
19        }
20    }
21 }
```

(5) 在dao接口 `RoleDao` 中添加查询方法 `findAllRoleByUserId`

```
1 List<Role> findAllRoleByUserId(String userId);
```

(6) 在映射配置文件 `RoleDao.xml` 中添加对应的查询

```
1 <!--配置根据ID查询-->
2 <select id="findAllRoleByUserId" parameterType="java.lang.String" resultMap="BaseResultMap">
3     SELECT
4         role_id,
5         NAME,
6         CASE
7             WHEN role_id IN (SELECT role_id FROM ss_role_user WHERE user_id = #{'userId'})
8             THEN 'checked'
9             ELSE ''
10        END
11        AS remark
12    FROM
13        ss_role
14 </select>
```

(7) 修改页面 `/WEB-INF/pages/system/user/role.jsp` , 添加checked

```
1 <form id="urform" action="${ctx}/system/user?operation=updateRole" method="post">
2   <input type="hidden" name="userId" value="${user.id}"/>
3   <div class="textbox" id="centerTextbox">
4     <div style="text-align:left">
5       <c:forEach items="${roleList}" var="role" varStatus="vs">
6         <span style="padding:3px;margin-right:30px;width: 160px;display: inline-
block">
7           <input type="checkbox" name="roleIds" value="${role.id}"
${role.remark}/>${role.name}
8           </span>
9         </c:forEach>
10      </div>
11    </div>
12  </form>
```

(8) 启动项目测试!

1.2 绑定用户与角色关系

现在要真正去绑定用户与角色的关系, 前台页面提交表单后会将用户的id和选择的角色的id传递到后台servlet

(1) 在 `UserServlet` 中添加新的方法

```
1 @Override
2 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
3   String operation = request.getParameter("operation");
4   if("list".equals(operation)){
5     this.list(request,response);
6   }
7   //中间的省略
8   else if("userRoleList".equals(operation)){
9     this.userRoleList(request,response);
10  }else if("updateRole".equals(operation)){
11    this.updateRole(request,response);
12  }
13 }
14 private void updateRole(HttpServletRequest request, HttpServletResponse response) throws
IOException {
15   String userId = request.getParameter("userId");
16   String[] roleIds = request.getParameterValues("roleIds");
17   userService.updateRole(userId,roleIds);
18   //跳转回到页面list
19   response.sendRedirect(request.getContextPath()+"/system/user?operation=list");
20 }
```

(2) 在 `UserService` 接口中添加一个新的方法 `updateRole`

```
1 void updateRole(String userId, String[] roleIds);
```

(3) 在对应的实现类中去实现

```
1 @Override
2 public void updateRole(String userId, String[] roleIds) {
3     SqlSession sqlSession = null;
4     try{
5         //1.获取SqlSession
6         sqlSession = MapperFactory.getSqlSession();
7         //2.获取Dao
8         UserDao userDao = MapperFactory.getMapper(sqlSession, UserDao.class);
9         userDao.deleteRole(userId);
10        for(String roleId : roleIds){
11            userDao.updateRole(userId, roleId);
12        }
13        //4.提交事务
14        TransactionUtil.commit(sqlSession);
15    }catch (Exception e){
16        TransactionUtil.rollback(sqlSession);
17        throw new RuntimeException(e);
18        //记录日志
19    }finally {
20        try {
21            TransactionUtil.close(sqlSession);
22        }catch (Exception e){
23            e.printStackTrace();
24        }
25    }
26 }
```

(4) 在dao接口 `UserDao` 中添加两个方法

```
1 void deleteRole(String userId);
2
3 void updateRole(@Param("userId") String userId, @Param("roleId")String roleId);
```

(5) 在该接口对应的映射配置文件中添加两个操作

```

1 <!--配置根据roleId删除关系表数据-->
2 <delete id="deleteRole" parameterType="java.lang.String">
3     delete from ss_role_user
4     where user_id = #{userId,jdbcType=VARCHAR}
5 </delete>
6
7 <!--配置全字段插入, 当某个字段没有值时, 插入null-->
8 <insert id="updateRole" parameterType="map">
9     insert into ss_role_user (role_id, user_id)
10    values (#{roleId,jdbcType=VARCHAR}, #{userId,jdbcType=VARCHAR})
11 </insert>

```

(6) 启动项目进行测试!

2. 登陆

2.1 登陆功能快速开发

(1) 找到项目 `webapp/login.jsp` 登陆页面, 修改form表单提交的action路径

```

1 <form action="${pageContext.request.contextPath}/login?operation=login" method="post">
2 </form>

```

(2) 在后台 `UserServlet` 添加登陆的方法

```

1 @Override
2 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
3     String operation = request.getParameter("operation");
4     if("list".equals(operation)){
5         this.list(request,response);
6     }
7     //中间省略
8     else if("login".equals(operation)){
9         this.login(request,response);
10    }
11 }
12 private void login(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
13     String email = request.getParameter("email");
14     String pwd = request.getParameter("password");
15     User user = userService.login(email,pwd);
16     if(user != null) {
17         request.getSession().setAttribute("loginUser", user);
18         //跳转页面
19         request.getRequestDispatcher("/WEB-INF/pages/home/main.jsp").forward(request,
response);
20     }else{
21         response.sendRedirect(request.getContextPath()+"/login.jsp");
22     }

```

```
23 } }
```

(3) 在因为层接口 `UserService` 中添加登陆的方法

```
1 /**
2  * 根据邮箱和密码登录
3  * @param email
4  * @param pwd
5  * @return
6  */
7 User login(String email, String pwd);
```

(4) 在对应的实现类中去实现登陆方法

```
1 @Override
2 public User login(String email, String pwd) {
3     SqlSession sqlSession = null;
4     try{
5         //1.获取SqlSession
6         sqlSession = MapperFactory.getSqlSession();
7         //2.获取Dao
8         UserDao userDao = MapperFactory.getMapper(sqlSession, UserDao.class);
9         //3.调用Dao层操作
10        pwd = MD5Util.md5(pwd);
11        return userDao.findByEmailAndPwd(email, pwd);
12    }catch (Exception e){
13        throw new RuntimeException(e);
14        //记录日志
15    }finally {
16        try {
17            TransactionUtil.close(sqlSession);
18        }catch (Exception e){
19            e.printStackTrace();
20        }
21    }
22 }
```

(5) 在 `UserDao` 接口中添加查询方法

```
1 User findByEmailAndPwd(@Param("email")String email, @Param("password")String pwd);
```

(6) 在 `UserDao.xml` 中添加查询

```
1 <select id="findByEmailAndPwd" parameterType="map" resultMap="BaseResultMap">
2     select
3     <include refid="Base_Column_List"/>
4     from ss_user
5     where email = #{email,jdbcType=VARCHAR} and password = #{password,jdbcType=VARCHAR}
6 </select>
```

(7) 将原本在 `webapp/pages/home` 下的所有页面统一挪到 `/WEB-INF/pages` 下，注意是连同 `home` 目录一起挪

(8) 修改 `/WEB-INF/pages/home/main.jsp` 内容区的路径

```
1 <!-- 内容区域 -->
2 <div class="content-wrapper">
3     <iframe id="iframe" name="iframe"
4         style="overflow:visible;"
5         scrolling="auto"
6         frameborder="no" height="100%" width="100%"
7         src="${ctx}/system/user?operation=home"></iframe>
8 </div>
```

(9) 在后台 `UserServlet` 中添加方法

```
1 @Override
2 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
   ServletException, IOException {
3     String operation = request.getParameter("operation");
4     if("list".equals(operation)){
5         this.list(request,response);
6     }
7     //中间省略
8     else if("home".equals(operation)){
9         this.home(request,response);
10    }
11 }
12 private void home(HttpServletRequest request, HttpServletResponse response) throws
   ServletException, IOException {
13     request.getRequestDispatcher("/WEB-INF/pages/home/home.jsp").forward(request, response);
14 }
```

(10) 启动项目测试

2.2 用户菜单控制数据准备

我们先来完成登陆后的注销操作，这是一套的

(1) 找到 `/WEB-INF/pages/home/header.jsp` 中找到注销，添加请求连接

```
1 <div class="pull-right">
2     <a href="${ctx}/system/user?operation=logout" class="btn btn-default btn-flat">注销</a>
3 </div>
```

(2) 在后台 `UserServlet` 中添加对应的方法logout

```
1 @Override
2 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
```

```

ServletException, IOException {
3     String operation = request.getParameter("operation");
4     if("list".equals(operation)){
5         this.list(request,response);
6     }
7     //中间的省略
8     else if("login".equals(operation)){
9         this.login(request,response);
10    }else if("logout".equals(operation)){
11        this.logout(request,response);
12    }else if("home".equals(operation)){
13        this.home(request,response);
14    }
15 }
16 private void logout(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
17     request.getSession().removeAttribute("loginUser");
18     response.sendRedirect(request.getContextPath()+"/login.jsp");
19 }

```

(3) 在用户登陆的时候需要去查询该用户对应的角色对应的所有模块，因此需要在后台的 `UserServlet` 中修改用户登陆的方法，添加数据的查询

```

1 private void login(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
2     String email = request.getParameter("email");
3     String pwd = request.getParameter("password");
4     User user = userService.login(email,pwd);
5     if(user != null) {
6         request.getSession().setAttribute("loginUser", user);
7         //如果登录成功，加载该用户对应的角色对应的所有模块
8         List<Module> moduleList = userService.findModuleById(user.getId());
9         request.setAttribute("moduleList",moduleList);
10        //跳转页面
11        request.getRequestDispatcher("/WEB-INF/pages/home/main.jsp").forward(request,
response);
12    }else{
13        response.sendRedirect(request.getContextPath()+"/login.jsp");
14    }
15 }

```

(4) 在 `UserService` 接口中添加方法 `findModuleById`

```

1 /**
2  * 根据用户id查询所有可以操作的菜单对象
3  * @param id 用户的id
4  * @return
5  */
6 List<Module> findModuleById(String id);

```

(5) 在实现类中去实现该方法


```

1  @Override
2  public List<Module> findModuleById(String id) {
3      SqlSession sqlSession = null;
4      try{
5          //1.获取SqlSession
6          sqlSession = MapperFactory.getSqlSession();
7          //2.获取Dao
8          ModuleDao moduleDao = MapperFactory.getMapper(sqlSession,ModuleDao.class);
9          //3.调用Dao层操作
10         return moduleDao.findModuleByUserId(id);
11     }catch (Exception e){
12         throw new RuntimeException(e);
13         //记录日志
14     }finally {
15         try {
16             TransactionUtil.close(sqlSession);
17         }catch (Exception e){
18             e.printStackTrace();
19         }
20     }
21 }

```

(6) 在 `ModuleDao` 接口中添加查询方法 `findModuleByUserId`

```

1  List<Module> findModuleByUserId(String id);

```

(7) 在 `ModuleDao.xml` 中添加对应的查询

```

1  <select id="findModuleByUserId" parameterType="java.lang.String" resultMap="BaseResultMap">
2      /*userid->用户角色关系表->roleid->角色模块关系表->moduleid->module信息*/
3      SELECT DISTINCT
4          m.module_id, m.parent_id, m.name, m.ctype, m.state, m.curl, m.remark
5      FROM
6          ss_module AS m,
7          ss_role_module AS rm,
8          ss_role_user AS ru
9      WHERE
10         m.module_id = rm.module_id
11         AND rm.role_id = ru.role_id
12         AND ru.user_id = #{id,jdbcType=VARCHAR}
13  </select>

```

至此：用户的角色对应的模块数据已查询出来了，后续就是要在页面进行控制展示

2.3 登陆用户菜单控制

(1) 找到 `/WEB-INF/pages/home/aside.jsp` 页面，添加用户菜单的展示

```

1  <!-- sidebar menu: : style can be found in sidebar.less -->

```

```

2 <ul class="sidebar-menu">
3   <li class="header">菜单</li>
4
5   <c:forEach items="${moduleList}" var="item">
6     <c:if test="${item.ctype==0}">
7       <li class="treeview">
8         <a href="#">
9           <i class="fa fa-cube"></i> <span>${item.name}</span>
10          <span class="pull-right-container"><i class="fa fa-angle-left pull-
right"></i></span>
11        </a>
12        <ul class="treeview-menu">
13          <c:forEach items="${moduleList}" var="item2">
14            <c:if test="${item2.ctype==1 && item2.parentId == item.id}">
15              <li id="${item2.id}">
16                <a onclick="setSidebarActive(this)"
href="${ctx}/${item2.cur1}" target="iframe">
17                  <i class="fa fa-circle-o"></i> ${item2.name}
18                </a>
19              </li>
20            </c:if>
21          </c:forEach>
22        </ul>
23      </li>
24    </c:if>
25  </c:forEach>

```

启动项目进行测试

3. 权限校验

3.1 获取请求url

(1) 创建过滤器: com.itheima.web.filters.AuthorFilter

```

1 @WebFilter(value = "/*")
2 public class AuthorFilter implements Filter {
3
4     private FilterConfig filterConfig;
5
6     /**
7      * 初始化方法, 获取过滤器的配置对象
8      * @param filterConfig
9      * @throws ServletException
10     */
11     @Override
12     public void init(FilterConfig filterConfig) throws ServletException {
13         this.filterConfig = filterConfig;
14     }
15
16     @Override
17     public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws

```

```

IOException, ServletException {
18     //1.定义和协议相关的请求和响应对象
19     HttpServletRequest request ;
20     HttpServletResponse response;
21     try{
22         //2.把参数转换成协议相关的对象
23         request = (HttpServletRequest)req;
24         response = (HttpServletResponse)resp;
25
26         //1.获取本次操作
27         String url = request.getRequestURI();
28         String queryString = request.getQueryString();
29
30         //1.当前获取到的url:    /system/dept
31         url = url.substring(1);
32         //2.当前获取到的查询参数: operation=list          operation=toEdit&id=100
33         int index = queryString.indexOf('&');
34         if(index != -1){
35             queryString = queryString.substring(0,index);
36         }
37         url = url + "?" + queryString;
38
39         //2.获取到当前登录人允许的操作
40
41         //3.比对本次操作是否在当前登录人允许的操作范围内
42         //3.1如果允许, 放行
43         //3.2不允许跳转到非法访问页
44
45         //6.放行
46         chain.doFilter(request,response);
47     }catch (Exception e){
48         e.printStackTrace();
49     }
50 }
51
52 @Override
53 public void destroy() {
54     //可以做一些清理操作
55 }
56 }

```

3.2 获取登陆用户可执行操作

(1) 登陆成功后需要将用户的觉得对应的模块信息存放到session, 找到 `UserServlet` 中的登陆方法 `login`,

```

1 private void login(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
2     String email = request.getParameter("email");
3     String pwd = request.getParameter("password");
4     User user = userService.login(email,pwd);
5     if(user != null) {

```

```

6         request.getSession().setAttribute("loginUser", user);
7         //如果登录成功, 加载该用户对应的角色对应的所有模块
8         List<Module> moduleList = userService.findModuleById(user.getId());
9         request.setAttribute("moduleList", moduleList);
10
11         //当前登录用户对应的可操作模块的所有url拼接成一个大的字符串
12         StringBuffer sbf = new StringBuffer();
13         for(Module m: moduleList){
14             sbf.append(m.getCurl());
15             sbf.append(',');
16         }
17         request.getSession().setAttribute("authorStr", sbf.toString());
18
19         //跳转页面
20         request.getRequestDispatcher("/WEB-INF/pages/home/main.jsp").forward(request,
response);
21     }else{
22         response.sendRedirect(request.getContextPath()+"/login.jsp");
23     }
24 }

```

(2) 修改 AuthorFilter

```

1  @Override
2  public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws
IOException, ServletException {
3      //1.定义和协议相关的请求和响应对象
4      HttpServletRequest request ;
5      HttpServletResponse response;
6      HttpSession session;
7      try{
8          //2.把参数转换成协议相关的对象
9          request = (HttpServletRequest)req;
10         response = (HttpServletResponse)resp;
11         session = request.getSession();
12         //1.获取本次操作
13         String url = request.getRequestURI();
14         //1.css .js .png .jpg .index
15         if(url.endsWith(".css")
16             || url.endsWith(".js")
17             || url.endsWith(".png")
18             || url.endsWith(".jpg")
19             || url.endsWith("index.jsp")
20             || url.endsWith("login.jsp")){
21             chain.doFilter(request, response);
22             return;
23         }
24         String queryString = request.getQueryString();
25         if(queryString.endsWith("operation=login")){
26             chain.doFilter(request, response);
27             return;
28         }

```

```

29
30
31 //1.当前获取到的url: /system/dept
32 url = url.substring(1);
33 //2.当前获取到的查询参数: operation=list operation=toEdit&id=100
34 int index = queryString.indexOf('&');
35 if(index != -1){
36     queryString = queryString.substring(0,index);
37 }
38 url = url + "?" + queryString;
39
40 //2.获取到当前登录人允许的操作
41 String authorStr = session.getAttribute("authorStr").toString();
42 //3.比对本次操作是否在当前登录人允许的操作范围内
43 //3.1如果允许,放行
44 //3.2不允许跳转到非法访问页
45
46 //6.放行
47 chain.doFilter(request,response);
48 }catch (Exception e){
49     e.printStackTrace();
50 }
51 }

```

(3) 启动项目在模块管理功能中去添加一些数据, 如下所示

系统管理
模块管理

模块信息

| | | | |
|-----|---|------|--|
| 模块名 | <input type="text" value="跳转到添加部门"/> | 上级模块 | <div>部门管理</div> |
| 类型 | <input checked="" type="radio"/> 主菜单 <input checked="" type="radio"/> 二级菜单 <input type="radio"/> 按钮 | 状态 | <input type="radio"/> 停用 <input checked="" type="radio"/> 启用 |
| 链接 | <input type="text" value="system/dept?operation=toAdd"/> | 备注 | <input type="text" value="备注"/> |

保存

返回

系统管理
模块管理

模块信息

| | | | |
|-----|--|------|--|
| 模块名 | <input type="text" value="添加部门"/> | 上级模块 | <div>部门管理</div> |
| 类型 | <input type="radio"/> 主菜单 <input checked="" type="radio"/> 二级菜单 <input type="radio"/> 按钮 | 状态 | <input type="radio"/> 停用 <input checked="" type="radio"/> 启用 |
| 链接 | <input type="text" value="system/dept?operation=save"/> | 备注 | <input type="text" value="备注"/> |

保存

返回

系统管理 模块管理 首页

模块信息

| | | | |
|-----|--|------|--|
| 模块名 | <input type="text" value="跳转到修改部门"/> | 上级模块 | <input type="text" value="部门管理"/> |
| 类型 | <input type="radio"/> 主菜单 <input checked="" type="radio"/> 二级菜单 <input type="radio"/> 按钮 | 状态 | <input type="radio"/> 停用 <input checked="" type="radio"/> 启用 |
| 链接 | <input type="text" value="system/dept?operation=toEdit"/> | 备注 | <input type="text" value="备注"/> |

保存 返回

系统管理 模块管理 首页

模块信息

| | | | |
|-----|--|------|--|
| 模块名 | <input type="text" value="修改部门"/> | 上级模块 | <input type="text" value="部门管理"/> |
| 类型 | <input type="radio"/> 主菜单 <input checked="" type="radio"/> 二级菜单 <input type="radio"/> 按钮 | 状态 | <input type="radio"/> 停用 <input checked="" type="radio"/> 启用 |
| 链接 | <input type="text" value="system/dept?operation=edit"/> | 备注 | <input type="text" value="备注"/> |

系统管理 模块管理 首页

模块信息

| | | | |
|-----|--|------|--|
| 模块名 | <input type="text" value="删除部门"/> | 上级模块 | <input type="text" value="部门管理"/> |
| 类型 | <input type="radio"/> 主菜单 <input checked="" type="radio"/> 二级菜单 <input type="radio"/> 按钮 | 状态 | <input type="radio"/> 停用 <input checked="" type="radio"/> 启用 |
| 链接 | <input type="text" value="system/dept?operation=delete"/> | 备注 | <input type="text" value="备注"/> |

保存 返回

然后需要在角色管理中为对应的角色进行授权

3.3 权限校验

开始授权

- (1) 从 `day03` 的课程资料中找到 `模块页面/unauthorized.jsp`，拷贝到项目的 `webapp` 下即可
- (2) 更改 `AuthorFilter`，

```
1  @Override
2  public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws
   IOException, ServletException {
3      HttpServletRequest request ;
4      HttpServletResponse response;
5      HttpSession session;
6      try{
7          request = (HttpServletRequest)req;
8          response = (HttpServletResponse)resp;
9          session = request.getSession();
10
11         //1.获取本次操作
12         String url = request.getRequestURI();
13
14         // .css .js .png .jpg .index
```

```

14         if(url.endsWith(".css")
15             || url.endsWith(".js")
16             || url.endsWith(".png")
17             || url.endsWith(".jpg")
18             || url.endsWith("index.jsp")
19             || url.endsWith("unauthorized.jsp")
20             || url.endsWith("login.jsp")){
21             chain.doFilter(request,response);
22             return;
23         }
24         String queryString = request.getQueryString();
25         if(queryString.endsWith("operation=login")
26             || queryString.endsWith("operation=home")
27             || queryString.endsWith("operation=logout")){
28             chain.doFilter(request,response);
29             return;
30         }
31         //1.当前获取到的url:    /system/dept
32         url = url.substring(1);
33         //2.当前获取到的查询参数: operation=list        operation=toEdit&id=100
34         int index = queryString.indexOf('&');
35         if(index != -1){
36             queryString = queryString.substring(0,index);
37         }
38         url = url + "?" + queryString;
39
40         //2.获取到当前登录人允许的操作
41         String authorStr = session.getAttribute("authorStr").toString();
42
43         //3.对比本次操作是否在当前登录人允许的操作范围内
44         if(authorStr.contains(url)){
45             //3.1如果允许,放行
46             chain.doFilter(request,response);
47             return;
48         }else{
49             //3.2不允许跳转到非法访问页
50             response.sendRedirect(request.getContextPath()+"/unauthorized.jsp");
51         }
52     }catch (Exception e){
53         e.printStackTrace();
54     }
55 }

```

(3) 对于页面上的元素, 如果没有操作权限, 我们直接让用户看不到即可, 怎么操作呢? 在页面上做一个判断, 我们举一个例子, 其他操作都是一样的

找到 `/WEB-INF/pages/system/user/list.jsp`,

```
1 <div class="btn-group">
2   <c:if test="${sessionScope.authorStr.contains('system/user?operation=toAdd')}}">
3     <button type="button" class="btn btn-default" title="新建"
onclick='location.href="${ctx}/system/user?operation=toAdd"'><i class="fa fa-file-o"></i> 新
建</button>
4   </c:if>
5   <button type="button" class="btn btn-default" title="删除" onclick='deleteById()'><i
class="fa fa-trash-o"></i> 删除</button>
6   <button type="button" class="btn btn-default" title="刷新"
onclick="window.location.reload();"><i class="fa fa-refresh"></i> 刷新</button>
7   <c:if test="${sessionScope.authorStr.contains('system/user?operation=userRoleList')}}">
8     <button type="button" class="btn btn-default" title="角色" onclick="roleList()"><i
class="fa fa-user-circle-o"></i> 角色</button>
9   </c:if>
10 </div>
```

(4) 启动项目，测试