第四章 Shiro 权限管理

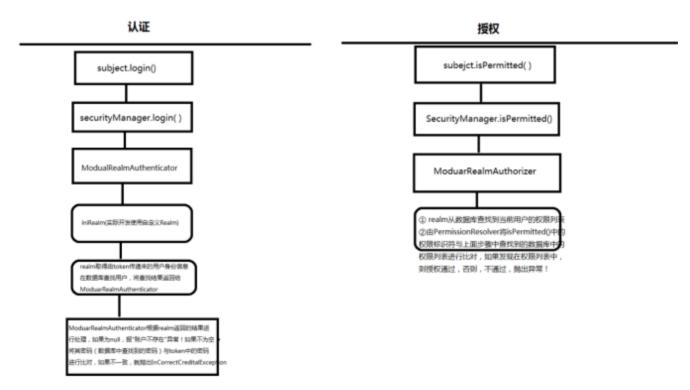
泽林.王峰 2019.4.16 星期二

授课大纲

- 1、上章回顾
- 2、Shiro与SSM整合开发
- 3、学生登录-Shiro认证
- 4、退出系统-Shiro 自带logout过滤器
- 5、学生查询、修改-Shiro授权
- 6、Shiro缓存使用

授课内容

0、上章回顾:



1、Shiro与SSM整合开发

1.1)新建maven项目Shiro_04,项目结构如下:

```
    Shiro_04 D:\ShiroProjects\Shiro_04

  ∨ src
      w main
         > java

✓ Imresources

            🗸 🖿 mapper
                  #StudentMapper.xml
                  🏭 SysUserMapper. xml

✓ Im spring

                  👼 applicationContext. xml
                  🚜 applicationContext-shire.xml
                  🌄 springmvc. xml
               🚮 anonymousURL. properties
               📊 commonURL. properties
               🚮 druid. properties
               📶 log4j. properties
           webapp
            bootstrap-3.3.7-dist

✓ WEB-INF

               V 📄 jsp
                  🗸 🗎 student
                         🚚 add. jsp
                         🚮 list.jsp
                         🚛 update. jsp
                  > user
                     error.jsp
                      📆 login. jsp
                  web.xml
```

1.2) web.xml文件的配置如下:

```
1
    <?xml version="1.0" encoding="UTF-8"?>
2
    <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"</pre>
 3
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
5
             version="3.1">
      <display-name>Shiro_04</display-name>
6
 7
      <!--配置shiroFilter -->
 8
      <filter>
9
        <filter-name>shiroFilter</filter-name>
10
        <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
11
        <!--此参数代表此过滤器的生命周期由serlvet管理-->
12
        <init-param>
13
          <param-name>targetFilterLifecycle</param-name>
          <param-value>true</param-value>
14
15
        </init-param>
        <!--此参数代表此过滤器与spring配置文件中配置的ShiroFactoryBean名称一致-->
16
17
        <init-param>
18
          <param-name>targetBeanName</param-name>
```

```
19
          <param-value>shiroFilter</param-value>
20
        </init-param>
21
      </filter>
22
      <filter-mapping>
23
        <filter-name>shiroFilter</filter-name>
        <url-pattern>/*</url-pattern>
24
25
      </filter-mapping>
26
27
      <!-- 1.加载spring所有的配置文件(不包含springmvc的配置文件) -->
28
      <context-param>
        <param-name>contextConfigLocation</param-name>
29
30
        <!-- 注意:
31
            classpath与classpath*区别:
            (1) classpath:代表访问指定类路径下的资源文件
32
33
            (2) classpath*:代表访问指定类路径及引入的第三方包中的资源文件。
34
35
        <param-value>classpath*:spring/applicationContext*.xml</param-value>
36
      </context-param>
37
      <!-- 2.配置spring的监听器 -->
38
      listener-
    class>org.springframework.web.context.ContextLoaderListener</listener-class>
39
      </listener>
40
41
      <!-- 3.配置DispatcherServlet -->
42
      <servlet>
43
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
44
45
        <init-param>
          <param-name>contextConfigLocation</param-name>
46
47
          <param-value>classpath:spring/springmvc.xml</param-value>
48
        </init-param>
49
      </servlet>
50
      <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
51
52
        <url-pattern>*.do</url-pattern>
53
      </servlet-mapping>
54
55
      <!-- 4.处理中文乱码的过滤器 -->
56
57
        <filter-name>characterEncoding</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-</pre>
58
    class>
59
        <init-param>
60
          <param-name>encoding</param-name>
          <param-value>UTF-8</param-value>
61
62
        </init-param>
63
      </filter>
64
      <filter-mapping>
65
        <filter-name>characterEncoding</filter-name>
66
        <url-pattern>/*</url-pattern>
67
      </filter-mapping>
      <welcome-file-list>
68
69
        <welcome-file>index.html</welcome-file>
```

1.3) pom.xml文件的配置如下:

```
1
    <?xml version="1.0" encoding="UTF-8"?>
 2
 3
    project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
 5
      <modelversion>4.0.0</modelversion>
 6
 7
      <groupId>com.zelin
 8
      <artifactId>Shiro_04</artifactId>
 9
      <version>1.0-SNAPSHOT</version>
10
      <packaging>war</packaging>
11
      <name>Shiro_04 Maven Webapp</name>
12
13
      <!-- FIXME change it to the project's website -->
14
      <url>http://www.example.com</url>
15
16
      <!-- 集中定义依赖版本号 -->
      properties>
17
18
        <junit.version>4.12</junit.version>
19
        <spring.version>4.2.4.RELEASE</spring.version>
20
        <mybatis.version>3.2.8</mybatis.version>
        <mybatis.spring.version>1.2.2</mybatis.spring.version>
21
        <mybatis.paginator.version>1.2.15</mybatis.paginator.version>
22
23
        <mysql.version>5.1.32</mysql.version>
        <slf4j.version>1.6.4</slf4j.version>
24
25
        <jackson.version>2.4.2</jackson.version>
        <druid.version>1.0.9</druid.version>
26
27
        <jstl.version>1.2</jstl.version>
28
        <servlet-api.version>2.5</servlet-api.version>
29
        <jsp-api.version>2.0</jsp-api.version>
30
        <commons-lang3.version>3.3.2</commons-lang3.version>
31
        <commons-io.version>1.3.2</commons-io.version>
32
        <pagehelper.version>4.0.0</pagehelper.version>
33
        <jsqlparser.version>0.9.1</jsqlparser.version>
        <commons-fileupload.version>1.3.1/commons-fileupload.version>
34
35
      </properties>
36
37
38
      <dependencies>
39
        <!-- Apache工具组件 -->
40
        <dependency>
          <groupId>org.apache.commons</groupId>
41
```

```
42
          <artifactId>commons-lang3</artifactId>
43
          <version>${commons-lang3.version}</version>
44
        </dependency>
        <dependency>
45
46
          <groupId>org.apache.commons
          <artifactId>commons-io</artifactId>
47
48
          <version>${commons-io.version}</version>
49
        </dependency>
50
        <!-- Jackson Json处理工具包 -->
51
        <dependency>
52
          <groupId>com.fasterxml.jackson.core</groupId>
          <artifactId>jackson-databind</artifactId>
53
54
          <version>${jackson.version}</version>
        </dependency>
55
56
        <!-- 单元测试 -->
57
58
        <dependency>
59
          <groupId>junit
60
          <artifactId>junit</artifactId>
61
          <version>${junit.version}</version>
62
        </dependency>
63
        <!-- 日志处理 -->
64
        <dependency>
65
          <groupId>org.slf4j</groupId>
          <artifactId>s1f4j-log4j12</artifactId>
66
          <version>${s1f4j.version}</version>
67
68
        </dependency>
69
        <!-- Mybatis -->
70
        <dependency>
71
          <groupId>org.mybatis
72
          <artifactId>mybatis</artifactId>
73
          <version>${mybatis.version}</version>
74
        </dependency>
75
        <dependency>
76
          <groupId>org.mybatis
          <artifactId>mybatis-spring</artifactId>
77
78
          <version>${mybatis.spring.version}</version>
79
        </dependency>
80
        <dependency>
81
          <groupId>com.github.miemiedev</groupId>
          <artifactId>mybatis-paginator</artifactId>
82
          <version>${mybatis.paginator.version}</version>
83
84
        </dependency>
85
        <dependency>
          <groupId>com.github.pagehelper</groupId>
86
87
          <artifactId>pagehelper</artifactId>
88
          <version>${pagehelper.version}</version>
89
        </dependency>
90
        <!-- MySql -->
91
        <dependency>
92
          <groupId>mysql</groupId>
          <artifactId>mysql-connector-java</artifactId>
93
94
          <version>${mysql.version}</version>
```

```
95
         </dependency>
 96
         <!-- 连接池 -->
 97
         <dependency>
 98
           <groupId>com.alibaba/groupId>
 99
           <artifactId>druid</artifactId>
           <version>${druid.version}</version>
100
101
         </dependency>
102
         <!-- Spring -->
103
         <dependency>
104
           <groupId>org.springframework</groupId>
105
           <artifactId>spring-context</artifactId>
106
           <version>${spring.version}</version>
107
         </dependency>
108
         <dependency>
109
           <groupId>org.springframework</groupId>
110
           <artifactId>spring-beans</artifactId>
111
           <version>${spring.version}</version>
112
         </dependency>
113
         <dependency>
114
           <groupId>org.springframework</groupId>
115
           <artifactId>spring-webmvc</artifactId>
116
           <version>${spring.version}</version>
117
         </dependency>
118
         <dependency>
119
           <groupId>org.springframework</groupId>
120
           <artifactId>spring-jdbc</artifactId>
121
           <version>${spring.version}</version>
122
         </dependency>
123
         <dependency>
124
           <groupId>org.springframework
125
           <artifactId>spring-aspects</artifactId>
126
           <version>${spring.version}</version>
127
         </dependency>
128
         <dependency>
129
           <groupId>org.springframework
130
           <artifactId>spring-jms</artifactId>
131
           <version>${spring.version}</version>
132
         </dependency>
133
         <dependency>
134
           <groupId>org.springframework</groupId>
135
           <artifactId>spring-context-support</artifactId>
136
           <version>${spring.version}</version>
137
         </dependency>
138
         <!-- JSP相关 -->
139
         <dependency>
140
           <groupId>jstl
141
           <artifactId>jstl</artifactId>
142
           <version>${jstl.version}</version>
143
         </dependency>
         <dependency>
144
145
           <groupId>javax.servlet
           <artifactId>servlet-api</artifactId>
146
147
           <version>${servlet-api.version}</version>
```

```
148
           <scope>provided</scope>
149
        </dependency>
150
        <dependency>
151
           <groupId>javax.servlet
152
           <artifactId>jsp-api</artifactId>
153
           <version>${jsp-api.version}</version>
154
           <scope>provided</scope>
155
         </dependency>
156
157
        <!-- 关于shiro相关 -->
158
        <dependency>
159
           <groupId>org.apache.shiro</groupId>
160
           <artifactId>shiro-core</artifactId>
           <version>1.3.2
161
162
        </dependency>
163
        <dependency>
164
           <groupId>org.apache.shiro
165
           <artifactId>shiro-web</artifactId>
           <version>1.3.2
166
167
        </dependency>
168
        <dependency>
169
           <groupId>org.apache.shiro
170
           <artifactId>shiro-spring</artifactId>
171
           <version>1.3.2
172
        </dependency>
        <dependency>
173
174
           <groupId>org.apache.shiro</groupId>
           <artifactId>shiro-ehcache</artifactId>
175
176
           <version>1.3.2
        </dependency>
177
        <dependency>
178
179
           <groupId>tk.mybatis
180
           <artifactId>mapper</artifactId>
181
           <version>4.1.5
182
         </dependency>
183
       </dependencies>
       <build>
184
185
        <finalName>${project.artifactId}</finalName>
186
        <plugins>
187
           <!-- 资源文件拷贝插件 -->
188
           <plugin>
189
            <groupId>org.apache.maven.plugins
190
            <artifactId>maven-resources-plugin</artifactId>
191
            <version>2.7</version>
192
            <configuration>
193
              <encoding>UTF-8</encoding>
194
            </configuration>
195
           </plugin>
196
        </plugins>
        <pluginManagement>
197
198
           <plugins>
            <!-- 配置Tomcat插件 -->
199
200
            <plugin>
```

```
<groupId>org.apache.tomcat.maven
201
202
               <artifactId>tomcat7-maven-plugin</artifactId>
203
               <configuration>
204
                 <contextReloadable>true</contextReloadable>
205
                 <port>9000</port>
206
                 <path>/</path>
207
               </configuration>
208
             </plugin>
209
           </plugins>
210
         </pluginManagement>
211
       </build>
212
     </project>
```

1.4) resources/spring/applicationContext.xml配置文件的内容如下:

```
1
    <?xml version="1.0" encoding="UTF-8"?>
 2
    <beans xmlns="http://www.springframework.org/schema/beans"</pre>
 3
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:context="http://www.springframework.org/schema/context"
 4
 5
           xsi:schemaLocation="http://www.springframework.org/schema/beans"
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">
 6
        <!--0. 定义spring的包扫描器-->
 7
        <context:component-scan base-package="com.zelin"/>
 8
        <!--1.引入属性文件-->
        <context:property-placeholder location="classpath*:druid.properties"/>
 9
10
        <!--2.配置数据源-->
        <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
11
            cproperty name="driverClassName" value="${jdbc.driver}"/>
12
13
            cproperty name="url" value="${jdbc.url}"/>
14
            cproperty name="username" value="${jdbc.username}"/>
            cproperty name="password" value="${jdbc.password}"/>
15
16
        </bean>
17
        <!--3.配置sqlSessionFactoryBean-->
18
        <bean class="org.mybatis.spring.SqlSessionFactoryBean">
19
            cproperty name="dataSource" ref="dataSource"/>
            cproperty name="typeAliasesPackage" value="com.zelin.pojo"/>
20
21
            cproperty name="plugins">
22
                st>
23
                    <bean class="com.github.pagehelper.PageHelper">
24
                         cproperty name="properties">
25
                             <value>
                                 dialect=mysql
26
27
                             </value>
28
                        </property>
29
                    </bean>
30
                </list>
31
            </property>
32
            cproperty name="mapperLocations" value="classpath*:mapper/*.xml"/>
33
        </bean>
34
        <!--4.接口扫描包-->
35
        <bean class="tk.mybatis.spring.mapper.MapperScannerConfigurer">
```

1.5) resources/spring/applicationContext-shiro.xml配置文件的内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
 1
 2
   <beans xmlns="http://www.springframework.org/schema/beans"</pre>
 3
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4
         xmlns:context="http://www.springframework.org/schema/context"
         xsi:schemaLocation="http://www.springframework.org/schema/beans"
   http://www.springframework.org/schema/beans/spring-beans.xsd
   http://www.springframework.org/schema/context
   http://www.springframework.org/schema/context/spring-context.xsd">
 6
       <!--1. 定义shiro的工厂bean对象-->
 7
       <bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
 8
          <!--1.1) 代表表单登录时的提交地址(登录页面处理器、同时也是显示登录页面的处理器)-->
          property name="loginUrl" value="/login.do"/>
 9
10
          <!--1.2)代表登录成功后跳转到的页面,默认行为是登录成功后会自动访问上一次访问的页面,所以此
   属性可不配-->
11
          cproperty name="successUrl" value="/user/listmenu.do"/>
          <!--1.3)配置认证失败后跳转的页面-->
12
13
          cproperty name="unauthorizedUrl" value="/resure.html"/>
          <!--1.4)引用一个securityManager对象-->
14
15
          cproperty name="securityManager" ref="mySecurityManager"/>
          <!--1.5)定义过滤器链-->
16
17
          <!--shiro自带的常用过滤器如下:-->
          <!--anon:例子/admins/**=anon 没有参数,表示可以匿名使用。-->
18
19
          <!--authc:例如/admins/user/**=authc表示需要认证(登录)才能使用,
   FormAuthenticationFilter是表单认证,没有参数-->
          <!--roles: 例子/admins/user/**=roles[admin],参数可以写多个,多个时必须加上引号,并且
20
   参数之间用逗号分割,当有多个参数时,例如admins/user/**=roles["admin,guest"],每个参数通过才算通
   过,相当于hasAllRoles()方法。-->
21
          <!--perms:例子/admins/user/**=perms[user:add:*],参数可以写多个,多个时必须加上引
   号,并且参数之间用逗号分割,例如/admins/user/**=perms["user:add:*,user:modify:*"],当有多个
   参数时必须每个参数都通过才通过,想当于isPermitedAll()方法。-->
22
          <!--user:例如/admins/user/**=user没有参数表示必须存在用户,身份认证通过或通过记住我认
   证通过的可以访问, 当登入操作时不做检查-->
23
          <!--Logout:代表退出过滤器,此url的action可以不定义,由shiro自动销毁session-->
24
          property name="filterChainDefinitions">
25
              <value>
26
                  <!-- 配置可以匿名访问的资源 -->
27
                  /bootstrap-3.3.7-dist/** = anon
                  <!-- 退出系统 -->
28
29
                  /logout.do = logout
30
                  <!-- 对验证码进行放行 -->
31
                  /validatecode.jsp = anon
32
                  <!--对用户进行权限拦截(授权)-->
33
34
                  <!--说明:=号左边是要访问的资源,=右右边是对应的权限标识符(权限表中的percode字
   段)
35
                     这种方式授权原理:会将perms[student:tolist]里面的权限拿来与自定义realm中
   的授权方法
```

```
中从数据库中取得的权限进行比对,如果在其中有这个权限,就通过,否则,就到达由
36
37
                       <property name="unauthorizedUrl" value="/resure.html"/>属性指定的页
    面。
38
39
                   /student/tolist.do = perms["student:tolist"]
                   /student/add.do = perms["student:create"]
40
41
42
                       /** = authc 代表系统所有资源需要经过系统认证才能访问,这个顺序放最后
43
                   /** = authc
44
45
               </value>
46
           </property>
47
        </bean>
48
        <!--2.定义SecurityManager对象-->
49
        <bean id="mySecurityManager"</pre>
    class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
50
            roperty name="realm" ref="customRealm"/>
51
        </bean>
52
        <!--3.构造自定义Realm对象-->
53
        <bean id="customRealm" class="com.zelin.realm.CustomRealm">
            cproperty name="credentialsMatcher" ref="credentialsMatcher"/>
54
55
        </bean>
56
        <!--4. 定义凭证匹配器-->
57
        <bean id="credentialsMatcher"</pre>
    class="org.apache.shiro.authc.credential.HashedCredentialsMatcher">
58
           <!--4.1)配置加密算法-->
59
            cproperty name="hashAlgorithmName" value="md5"/>
60
            <!--4.2)配置加密次数-->
            roperty name="hashIterations" value="1"/>
61
62
        </bean>
63
    </beans>
```

1.6) resources/spring/springmvc.xml配置文件的内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
 1
 2
    <beans xmlns="http://www.springframework.org/schema/beans"</pre>
 3
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:context="http://www.springframework.org/schema/context"
 4
 5
           xmlns:mvc="http://www.springframework.org/schema/mvc"
 6
           xsi:schemaLocation="http://www.springframework.org/schema/beans"
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc.xsd">
 7
        <!--1.添加对控制器的扫描-->
 8
        <context:component-scan base-package="com.zelin.controller"/>
 9
        <!--2.添加处理器映射器与处理器适配器-->
        <mvc:annotation-driven/>
10
11
        <!--3.添加视图解析器-->
12
        <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
13
            cproperty name="prefix" value="/WEB-INF/jsp/"/>
14
            roperty name="suffix" value=".jsp"/>
```

1.7) 数据源配置

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/shiro
jdbc.username=root
jdbc.password=123
```

2、Shiro认证/授权

2.1) 自定义realm(认证+授权方式一[在shiro的配置文件中定义过滤器链perms])

```
package com.zelin.realm;
 1
 2
 3
   import com.zelin.pojo.SysPermission;
 4
    import com.zelin.pojo.SysUser;
 5
    import com.zelin.service.SysUserService;
 6
    import org.apache.shiro.authc.AuthenticationException;
 7
    import org.apache.shiro.authc.AuthenticationInfo;
    import org.apache.shiro.authc.AuthenticationToken;
 9
    import org.apache.shiro.authc.SimpleAuthenticationInfo;
10
    import org.apache.shiro.authz.AuthorizationInfo;
11
    import org.apache.shiro.authz.SimpleAuthorizationInfo;
    import org.apache.shiro.realm.AuthorizingRealm;
12
13
    import org.apache.shiro.subject.PrincipalCollection;
    import org.apache.shiro.util.ByteSource;
14
15
    import org.springframework.beans.factory.annotation.Autowired;
16
    import org.springframework.stereotype.Component;
17
18
    import java.util.ArrayList;
19
    import java.util.Collection;
20
    import java.util.List;
21
22
23
    * @Author: Feng.Wang
24
    * @Company: Zelin.ShenZhen
25
    * @Description:
26
     * @Date: Create in 2019/4/16 09:21
27
     */
28
29
    public class CustomRealm extends AuthorizingRealm {
30
        @Autowired
31
        private SysUserService userService;
32
        //用户授权
33
        @override
        protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals)
34
    {
35
            //1.取得主身份信息
36
            SysUser user = (SysUser) principals.getPrimaryPrincipal();
37
            List<SysPermission> permissions = null;
```

```
if(null != user){
38
39
                //2.取得此用户的权限列表
40
               permissions = userService.findPermissionsByUserCode(user.getUsercode());
               if(permissions != null && permissions.size() > 0){
41
42
                   user.setPermissions(permissions);
43
               }
44
           }
45
            //3.定义授权对象
           SimpleAuthorizationInfo authorizationInfo = new SimpleAuthorizationInfo();
46
47
            //4.将上面的用户的权限列表字符串赋值给当前的授权对象
           List<String> permissionList = new ArrayList<>();
48
49
           //5.转换上面的List<SysPermission>对象为List<String>对象
           for (SysPermission permission : permissions) {
51
                permissionList.add(permission.getPercode());
52
           }
53
           //6.将权限码列表与当前授权对象关联
54
           authorizationInfo.addStringPermissions(permissionList);
55
           //7.返回授权对象
56
           return authorizationInfo;
57
        }
58
59
        //用户认证
60
        @override
        protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token)
61
    throws AuthenticationException {
62
           //1.得到主身份信息
63
           String usercode = (String) token.getPrincipal();
64
            //2.如果用户名存在
65
           if(usercode != null){
66
                SysUser user = userService.findUserByUserCode(usercode);
67
               if(user == null) return null;
68
               //2.1)如果不为null,就查询其对应的菜单列表
69
               List<SysPermission> menus = userService.findMenusByUserCode(usercode);
70
               user.setMenus(menus);
71
               //2.2)取出数据库中的密码及加盐值
72
               String password = user.getPassword();
73
               String salt = user.getSalt();
74
               return new SimpleAuthenticationInfo(user,password,
    ByteSource.Util.bytes(salt), "aaa");
           }
76
           return null;
77
       }
78
    }
79
```

2.2) 自定义异常处理类:

```
public class MyException extends RuntimeException {
    private static final long serialVersionUID = 1L;
    //代表异常的错误信息
    private String message;
    public String getMessage() {
        return message;
```

```
7
        public void setMessage(String message) {
 8
 9
             this.message = message;
        }
10
11
        public MyException() {
12
            super();
13
        }
14
        public MyException(String message) {
15
            super();
16
            this.message = message;
17
        }
18
   }
```

2.3) 注解完成全局异常处理:

```
1 /**
 2
    * @Author: Feng.Wang
 3
    * @Company: Zelin.ShenZhen
4
    * @Description:
5
    * @Date: Create in 2019/4/16 09:56
6
    */
7
    @ControllerAdvice
    public class MyExceptionHandler {
8
9
       //定义异常处理器
10
       @ExceptionHandler(MyException.class)
11
        public ModelAndView myException(MyException ex){
12
           //1.得到异常出错信息
13
           String message = ex.getMessage();
           //2.如果信息为null,就指定一个message值
14
15
           if(StringUtils.isEmpty(message)){
               message = "未知异常!";
16
17
           }
           //3.到错误页面进行处理
18
           return new ModelAndView("error", "message", message);
19
20
       }
21
    }
22
```

2.4) 登录处理器login.do:

```
1 /**
2
    * @Author: Feng.Wang
 3
   * @Company: Zelin.ShenZhen
    * @Description:
4
5
    * @Date: Create in 2019/4/16 09:39
6
    */
7
   @Controller
8
   public class LogController {
9
       @RequestMapping("/login")
10
       //默认情况下, shiro的用户认证工作由FormAuthenticationFilter(authc)过滤器完成,
11
       //当出现异常时, shiro会将异常对象的类名放到以shiroLoginFailure为key的request
12
       //对象中,我们只需要通过此key取出对应的异常类名就可以知道,用户是何种异常
```

```
13
       public String login(HttpServletRequest request, HttpServletResponse response)
    throws Exception {
14
           //1.得到异常的名称
           String exceptionName = (String) request.getAttribute("shiroLoginFailure");
15
16
           //2.根据异常的名称来判断执行的是何种异常,从而处理此异常
17
           //2.1)判断是否为null
18
           if(StringUtils.isNotEmpty(exceptionName)){
19
               if(UnknownAccountException.class.getName().equals(exceptionName)){
                   throw new MyException("账户异常!");
20
21
               }else
    if (IncorrectCredentialsException.class.getName().equals(exceptionName)) \\ \{
                   throw new MyException("用户名或密码输入有误!");
22
23
               }else if(UnauthorizedException.class.getName().equals(exceptionName)){
                   throw new MyException("无权限访问异常!");
24
25
               }else{
26
                   throw new Exception();
27
               }
28
           }
29
           //此方法只有在FormAuthenticationFilter认证失败时工作,认证成功返回页面就是上一次的页面
30
           return "login";
31
       }
32
   }
```

2.5) 当未登录直接访问页面,就会执行/login.do,从而到达/WEB-INF/jsp/login.jsp页面

```
1
    <%--
 2
     Created by Intellij IDEA.
 3
      User: Administrator
 4
     Date: 2019/4/12
 5
      Time: 10:21
     To change this template use File | Settings | File Templates.
 6
 7
 8
    <%@ page contentType="text/html;charset=UTF-8" language="java" %>
    <%@include file="/base.jsp"%>
 9
10
    <html>
11
    <head>
12
        <title>用户登录</title>
13
        <style>
14
            .container{
15
                width: 500px;
16
                margin-top: 50px;
17
            }
18
            .form-signin{
19
                padding:5px;
20
21
            .btn{
22
                margin-top: 20px;
23
            }
            .error{
24
25
                color:red;
26
            }
27
        </style>
28
    </head>
```

```
29
    <body>
    <div class="container">
30
31
        <div class="panel panel-primary">
            <div class="panel-heading">
32
33
                 <h3 class="panel-title">
                     用户登录
34
35
                </h3>
36
            </div>
            <div class="panel-body">
37
38
            <form class="form-signin" action="${pageContext.request.contextPath}/login.do"</pre>
    method="post">
39
                <label>用户名</label>
40
                <input type="text" name="username" class="form-control" placeholder="输入</pre>
    用户名" required autofocus>
41
                 <label >密码</label>
42
                 <input type="password" name="password" class="form-control"</pre>
    placeholder="输入密码" required>
43
                <button class="btn btn-lg btn-primary btn-block" type="submit">登录
    </button>
                 <span class="error">${message}</span>
44
45
            </form>
        </div>
46
47
        </div>
48
    </div> <!-- /container -->
    </body>
49
50
    </html>
51
```

2.6)/WEB-INF/user/listmenu.jsp列表菜单:

```
1
    <%--
     Created by Intellij IDEA.
 2
 3
      User: Administrator
 4
     Date: 2019/4/12
 5
      Time: 10:51
     To change this template use File | Settings | File Templates.
 6
 7
    --%>
    <%@ page contentType="text/html;charset=UTF-8" language="java" %>
 8
    <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
 9
10
    <%@include file="/base.jsp"%>
    <style>
11
12
        .col-md-2{
13
            margin-top: 20px;
14
        }
15
        .list-group-item{
16
            text-align: center;
17
        }
18
        .col-md-10{
            padding-left: 0px;
19
20
            margin-left: -10px;
21
            margin-top: 20px;
22
        }
23
    </style>
```

```
24
    <html>
25
    <head>
26
        <title>显示菜单</title>
27
        <meta charset="UTF-8">
28
    </head>
29
    <body>
30
        <div class="container">
31
            <%--分成左右栏--%>
            <%--1.左边显示菜单--%>
32
            <div class="row">
33
34
                <div class="col-md-2">
                    <div class="list-group">
35
36
                        <a href="#" class="list-group-item list-group-item-danger"
    disabled">
37
                            用户菜单
38
                        </a>
                        <c:forEach items="${menus}" var="m">
39
40
                             <a href="${m.url}" target="right" class="list-group-item")</pre>
    tt">${m.name}</a>
41
                        </c:forEach>
42
                    </div>
                </div>
43
44
                <%--2.右边显示效果--%>
45
                <div class="col-md-10">
                    <iframe src="" width="1100" height="800" frameborder="0" name="right">
46
    </iframe>
47
                </div>
            </div>
48
49
        </div>
50
    </body>
51
    </html>
52
    <script>
53
        $(function () {
            //切换样式
54
55
            $(".tt").click(function () {
               //1.将所有的list-group下的a标签下的active样式去除
56
                $(".list-group a").removeClass("active");
57
5.8
                //2.将当前点击的哪个a设置active样式
                $(this).addClass("active");
59
60
            })
61
        })
    </script>
62
```

2.7) /WEB-INF/student/list.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false" %>
1
2
   <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3
   <%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4
   <html>
5
   <jsp:include page="/base.jsp"/>
6
   <head>
7
       <meta charset="UTF-8">
8
       <style>
```

```
9
           .table {
10
               text-align: center;
11
           }
           .container {
12
13
               margin-top: 20px;
14
15
           #form1 {
16
               margin-top: 10px;
               margin-left: 30px;
17
18
           }
19
       </style>
   </head>
20
21
    <body>
    <div class="container">
22
23
       <div class="panel panel-primary">
24
           <div class="panel-heading">
25
               <h3 class="panel-title">学生查询</h3>
26
           </div>
27
28
           <form class="form-inline" id="form1" method="post"</pre>
29
                 action="${pageContext.request.contextPath}/student/search.do">
30
               <input type="hidden" name="page" id="page">
31
               学生姓名: <input type="text" name="sname" value="${param.sname}"
    placeholder="学生姓名关键字" class="form-control">
32
               学生住址: <input type="text" name="addr" value="${param.addr}"
    placeholder="学生住址关键字" class="form-control">
               所在班级:
33
               <select name="cid" class="form-control">
34
35
                   <option value="0">所有班级</option>
                   <c:forEach items="${classes}" var="c">
36
37
                      <option value="${c.cid}"</pre>
    ${c.cid==param.cid?"selected":""}>${c.cname}</option>
                  </c:forEach>
38
39
               </select>
               <input type="submit" value="查询" class="btn btn-default btn-sm">
40
               <input type="button" value="添加学生" class="btn btn-success btn-sm"
41
   onclick="addStudent()">
42
               <a href="javascript:parent.location.href = '/logout.do'" class="btn btn-</pre>
    success btn-sm">注销</a>
43
           </form>
           44
45
               46
                   姓名
47
                   性别
                   年龄
48
49
                   住址
50
                   生日
51
                   新在班级
52
                   操作
53
               <c:forEach items="${students}" var="stud">
54
55
56
                      ${stud.sname}
```

```
57
                       ${stud.sex}
58
                       ${stud.age}
59
                       ${stud.addr}
60
                       <fmt:formatDate value="${stud.birth}" pattern="yyyy-MM-dd E"/>
61
                       62
63
                       64
                              ${stud.classes.cname}
65
                       66
                       67
                          <a class="btn btn-primary btn-sm"</pre>
68
    href="${pageContext.request.contextPath}/student/toupdate.do?sid=${stud.sid}">修改</a>
                          <a class="btn btn-danger btn-sm"</pre>
69
70
    href="${pageContext.request.contextPath}/student/deleteBySid.do?sid=${stud.sid}"
71
                             onclick="return confirm('你真的要删除吗?')">删除</a>
72
                       73
                   74
               </c:forEach>
           75
76
       </div>
77
    </div>
78
    <script>
79
       //执行提交 表单
80
       function skip(i) {
81
           //1.对表单中的隐藏域赋值
82
           $("#page").val(i);
83
           //2. 提交表单
           $("#form1").submit();
84
85
       }
86
       //添加学生
87
       function addStudent() {
           location.href = "${pageContext.request.contextPath}/student/toadd.do";
88
89
       }
90
    </script>
91
    </body>
    </html>
92
93
```

2.8) /user/listmenu.do展示用户列表

```
1
   /**
 2
    * @Author: Feng.Wang
 3
    * @Company: Zelin.ShenZhen
 4
    * @Description:
 5
    * @Date: Create in 2019/4/8 10:50
 6
 7
    @Controller
    @RequestMapping("/student")
 9
    public class StudentController {
10
        @Autowired
11
        private StudentService studentSerivce;
```

```
12
        @Autowired
13
        private ClassesService classesService;
                            //代表每页的大小
14
        int pageSize = 5;
        //查询所有的学生
15
        @RequestMapping("/tolist")
16
17
        public String findAll(Model model){
18
           try {
               //1.查询所有的学生集合
19
20
               List<Student> students = studentSerivce.findStudents();
               System.out.println("---->" + students);
21
22
               //2.将上面的集合放到model中
23
               model.addAttribute("students", students);
               model.addAttribute("classes", classesService.findAll());
24
25
           } catch (Exception e) {
26
               e.printStackTrace();
27
28
           //4.返回逻辑视图
29
           return "student/list";
30
       }
31
   }
```

2.9) 运行效果如下:





3、授权方式二(使用@RequirePermission注解结合shiro的标签完成授权)

3.1) 打开spring对aop代理的支持(springmvc.xml文件中):

```
1 <!--0.开启aop注解-->
2 <aop:config proxy-target-class="true"/>
```

3.2) 打开shiro的注解支持(applicationContext-shiro.xml文件中):

3.3)在springmvc中添加对通用异常的处理(springmvc.xml):

```
<!-- springmvc 处理异常-->
1
2
   class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">
3
           cproperty name="exceptionMappings">
4
               ops>
5
                   <!--映射当授权出错后跳转到的逻辑错误页面/WEB-INF/jsp/refuse.jsp-->
6
                    key="org.apache.shiro.authz.UnauthorizedException">refuse
7
                   prop
   key="org.apache.shiro.authz.UnauthenticatedException">refuse</prop>
8
               </props>
9
           </property>
10
       </bean>
```

3.4) 在/user/userlist.do处理器:

```
1
     /**
2
         * 查询所有的用户
 3
         * @return
 4
        */
 5
        @RequiresPermissions("user:userlistxxx") //代表没有此权限就不能访问/user/userlist.do
        @RequestMapping("/userlist")
 6
 7
        public String findAll(Model model){
8
            try {
9
                model.addAttribute("users", userService.findUsers());
10
            } catch (Exception e) {
               System.out.println("对不起,你无权访问此页面!");
11
12
            }
13
            return "user/list";
14
        }
```

3.5)/WEB-INF/jsp/user/list.jsp用户列表页面

```
1
   <%@ page contentType="text/html;charset=UTF-8" language="java" %>
2
   <html>
3
   <head>
       <title>用户列表</title>
4
5
  </head>
6
  <body>
7
       ${users}
8
   </body>
9
   </html>
```

3.6) 在/WEB-INF/jsp/student/list.jsp中添加学生shiro标签

3.7) 运行效果如下:



无访问权限!



学生查询						
学生姓名:	学生姓名关键字		学生住址: 学生住址关键字 所在班级:		所有班级 ▼ 查询	注销
姓名	性别	年龄	住址	生日	所在班级	这里操作了添加按钮
张三	男	20	上海	1995-07-13 星期四	1301班	修改 删除
五二	男	28	广州	2000-07-20 星期四	1301班	修改 删除
小红	女	19	杭州	1995-07-13 星期四	1302班	修改 删除
王二小	男	12	岳阳	1986-06-11 星期三	1302班	修改 删除

4、Shiro缓存使用

- 4.1) 在e: 创建存放存缓文件的目录e:/develop/ehcache
- 4.2) 在resources/cache目录中新建shiro-cache.xml文件

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"</pre>
 1
 2
            xsi:noNamespaceSchemaLocation="../config/ehcache.xsd">
 3
       <!--diskStore: 缓存数据持久化的目录 地址 -->
       <diskStore path="e:\develop\ehcache" />
 4
 5
       <defaultCache
              maxElementsInMemory="1000"
 6
 7
              maxElementsOnDisk="10000000"
8
               eternal="false"
               overflowToDisk="false"
9
10
               diskPersistent="false"
               timeToIdleSeconds="120"
11
12
               timeToLiveSeconds="120"
13
               diskExpiryThreadIntervalSeconds="120"
14
              memoryStoreEvictionPolicy="LRU">
       </defaultCache>
15
       <!--name: cache的名字,用来识别不同的cache,必须惟一。-->
16
17
       <!--maxElementsInMemory:内存管理的缓存元素数量最大限值。-->
       <!--maxElementsOnDisk: 硬盘管理的缓存元素数量最大限值。默认值为0,就是没有限制。-->
18
19
       <!--eternal: 设定元素是否持久话。若设为true,则缓存元素不会过期。-->
       <!--overflowToDisk: 设定是否在内存填满的时候把数据转到磁盘上。-->
20
```

4.3)在resources/spring/applicationConext-shiro.xml文件中添加缓存管理器

```
1
      <!--5.配置缓存管理器-->
2
      <bean id="cacheManager" class="org.apache.shiro.cache.ehcache.EhCacheManager">
          3
   cache.xml"/>
4
      </bean>
5
       <!--2.定义SecurityManager对象-->
      <bean id="mySecurityManager"</pre>
6
   class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
7
          <!--2.1)添加自定义realm-->
         roperty name="realm" ref="customRealm"/>
8
9
         <!--2.2)添加缓存管理器(引入上面定义)-->
10
          cacheManager" ref="cacheManager"/>
11
      </bean>
```

4.4)测试略过。

5、验证码的使用

5.1) 在webapp下添加/validatecode.jsp,生成验证码

```
1
    <%@ page language="java" contentType="text/html; charset=UTF-8"</pre>
 2
        pageEncoding="UTF-8"%>
 3
    <%@ page import="java.util.Random"%>
 4
    <%@ page import="java.io.OutputStream"%>
 5
    <%@ page import="java.awt.Color"%>
    <%@ page import="java.awt.Font"%>
 6
 7
    </@ page import="java.awt.Graphics"%>
    <%@ page import="java.awt.image.BufferedImage"%>
 8
 9
    <%@ page import="javax.imageio.ImageIO"%>
10
        int width = 60;
11
12
        int height = 32;
13
        //create the image
14
        BufferedImage image = new BufferedImage(width, height,
    BufferedImage.TYPE_INT_RGB);
15
        Graphics g = image.getGraphics();
16
        // set the background color
17
        g.setColor(new Color(0xDCDCDC));
18
        g.fillRect(0, 0, width, height);
19
        // draw the border
20
21
        g.setColor(new Color(0x337ab7));
22
        g.drawRect(0, 0, width - 1, height - 1);
23
        // create a random instance to generate the codes
24
        Random rdm = new Random();
25
        String hash1 = Integer.toHexString(rdm.nextInt());
26
        System.out.print(hash1);
        // make some confusion
27
```

```
28
        for (int i = 0: i < 50: i++) {
29
            int x = rdm.nextInt(width);
            int y = rdm.nextInt(height);
30
31
            g.drawoval(x, y, 0, 0);
32
        }
33
        // generate a random code
34
        String capstr = hash1.substring(0, 4);
35
        //将生成的验证码存入session
        session.setAttribute("validateCode", capstr);
36
37
        g.setColor(new Color(0, 100, 0));
38
        g.setFont(new Font("Candara", Font.BOLD, 24));
39
        g.drawString(capstr, 8, 24);
40
        q.dispose();
41
        //输出图片
42
        response.setContentType("image/jpeg");
43
        out.clear();
44
        out = pageContext.pushBody();
45
        OutputStream strm = response.getOutputStream();
46
        ImageIO.write(image, "jpeg", strm);
47
        strm.close();
48 %>
```

5.2) 自定义表单过滤器, 重写其onAccessDenied()方法

```
1 /**
 2
    * @Author: Feng.Wang
 3
    * @Company: Zelin.ShenZhen
    * @Description: 自定义表单过滤器完成验证码的验证
 5
    * @Date: Create in 2019/4/16 15:27
 6
    */
 7
    public class CustomFormFilter extends FormAuthenticationFilter {
 8
        @override
 9
        protected boolean onAccessDenied(ServletRequest request, ServletResponse response)
    throws Exception {
10
           //1.将原始的请求与响应对象转换为基于http请求的对象
11
           HttpServletRequest req = (HttpServletRequest) request;
12
           HttpServletResponse resp = (HttpServletResponse) response;
           //2.从session中取出原始的验证码
13
14
           HttpSession session = req.getSession();
15
           String validateCode = (String) session.getAttribute("validateCode");
           //3.得到用户输入的验证码
16
17
           String validcode = req.getParameter("validcode");
18
           //4.比较两次验证码是否一样,如果两次验证码不相等,就将错误信息放到request对象中
19
           if(StringUtils.isNotEmpty(validcode) && !validateCode.equals(validcode)){
               req.setAttribute("shiroLoginFailure","validCodeError");
20
21
                                //代表验证未通过
               return true;
22
           return super.onAccessDenied(request, response);
23
       }
24
25
    }
26
```

5.3) 在resources/spring/applicationContext-shiro.xml文件中添加如下配置:

```
1
       <!--6. 自定义表单过滤器-->
 2
        <bean id="customFormFilter" class="com.zelin.filter.CustomFormFilter">
 3
            <!--6.1)代表更改表单中的用户名的name属性为uname,默认为username-->
 4
            <!--<pre>--roperty name="usernameParam" value="uname"/>-->
 5
            <!--6.2)代表更改表单中的密码的name属性为pwd,默认为password-->
 6
            <!--<pre>--> roperty name="passwordParam" value="pwd"/>-->
            <!--6.3)代表更改出错后放到request的默认的错误key的名称,默认叫shiroLoginFailre-->
 7
 8
            <!--<pre>---<!--<pre>roperty name="failureKeyAttribute" value="errorKey"/>-->
 9
        </bean>
10
    <bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
11
12
          <!--1.6)配置自定义表单过滤器-->
13
            roperty name="filters">
14
                <map>
15
                   <!--将shiro中原来自带的authc这个过滤器用我们自定义的过滤器代替掉-->
16
                   <entry key="authc" value-ref="customFormFilter"/>
17
                </map>
18
            </property>
19
    </bean>
```

5.4)在loginController.java中添加对新的验证码异常的处理:

```
@Controller
 1
 2
    public class LogController {
 3
       @RequestMapping("/login")
 4
       //默认情况下, shiro的用户认证工作由FormAuthenticationFilter(authc)过滤器完成,
 5
       //当出现异常时, shiro会将异常对象的类名放到以shiroLoginFailure为key的request
 6
       //对象中,我们只需要通过此key取出对应的异常类名就可以知道,用户是何种异常
 7
       public String login(HttpServletRequest request, HttpServletResponse response)
    throws Exception {
 8
           //1.得到异常的名称
9
           String exceptionName = (String) request.getAttribute("shiroLoginFailure");
10
           //2.根据异常的名称来判断执行的是何种异常,从而处理此异常
11
           //2.1)判断是否为null
12
           if(StringUtils.isNotEmpty(exceptionName)){
               if(UnknownAccountException.class.getName().equals(exceptionName)){
13
                  throw new MyException("账户异常!");
14
15
               }else
    if(IncorrectCredentialsException.class.getName().equals(exceptionName)){
                  throw new MyException("用户名或密码输入有误!");
16
17
               }else if(UnauthorizedException.class.getName().equals(exceptionName)){
                  throw new MyException("无权限访问异常!");
18
               }else if("validCodeError".equals(exceptionName)){
19
20
                  throw new MyException("验证码错误!");
21
              }else{
22
                  throw new Exception();
23
               }
24
           }
25
           //此方法只有在FormAuthenticationFilter认证失败时工作,认证成功返回页面就是上一次的页面
26
           return "login";
```

```
27 | }
28 | }
```

5.5)测试:输入一个错误的验证码:



异常信息:

验证码错误!