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

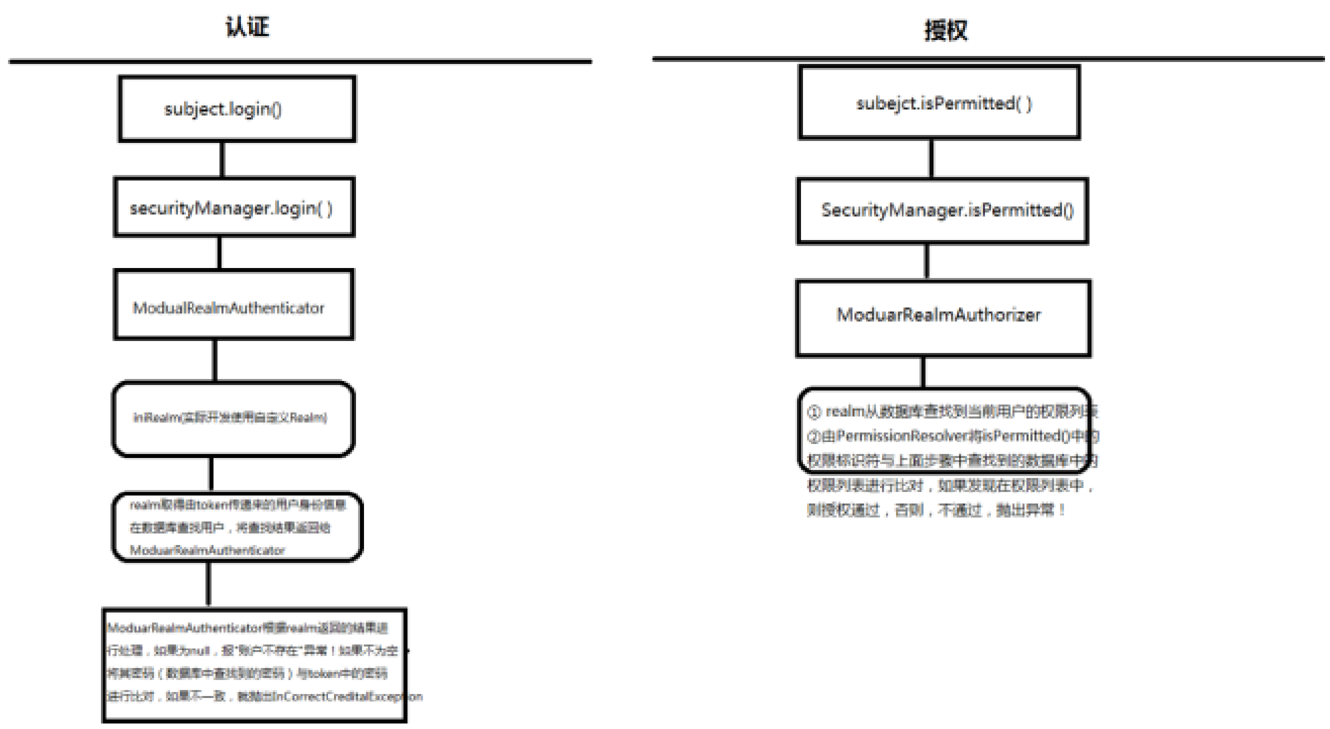
4、 退出系统-Shiro 自带logout过滤器

5、 学生查询、修改-Shiro授权

6、 Shiro缓存使用

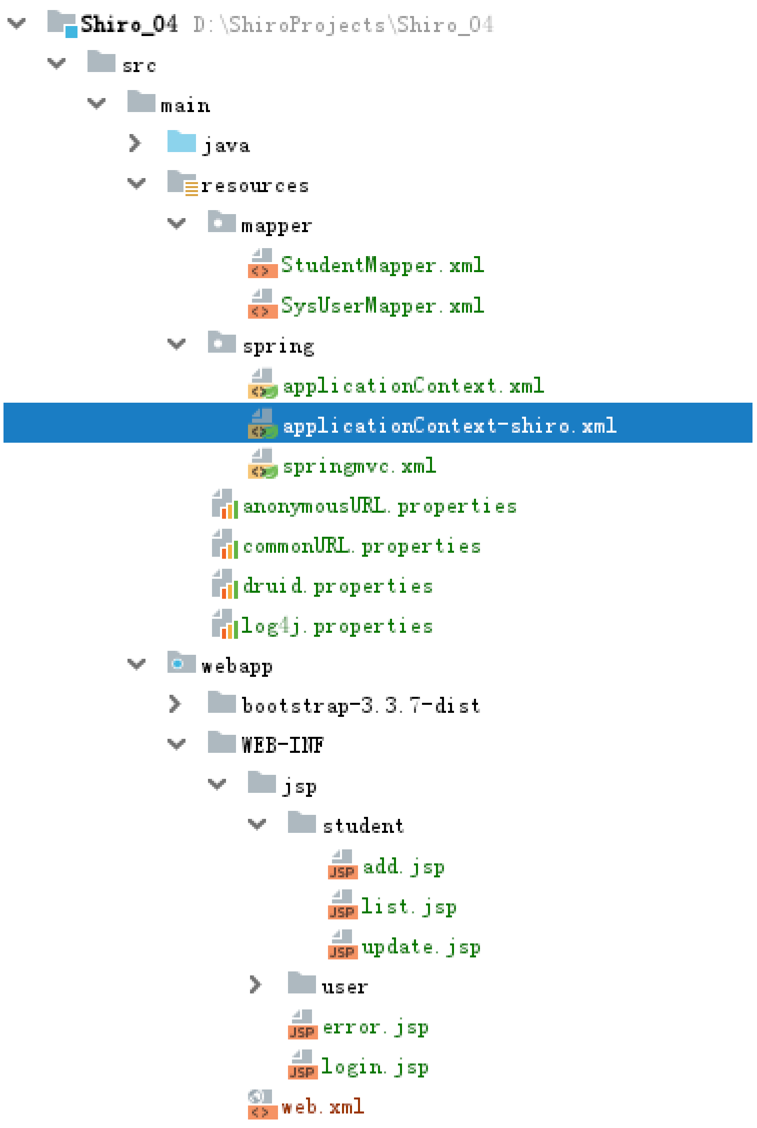
授课内容

0、上章回顾：


认证
subject.login()
securityManager.login( )
ModualRealmAuthenticator
inRealm(实际开发使用自定义Realm)
realm取得由token传递来的用户身份信息在数据库查找用户，将查找结果返回给ModuarRealmAuthenticator
ModuarRealmAuthenticator根据realm返回的结果进行处理，如果为null，报"账户不存在"异常！如果不为空将其密码（数据库中查找到的密码）与token中的密码进行比对，如果不一致，就抛出InCorrectCreditalException
授权
subejct.isPermitted( )
SecurityManager.isPermitted()
ModuarRealmAuthorizer
① realm从数据库查找到当前用户的权限列表
②由PermissionResolver将isPermitted()中的权限标识符与上面步骤中查找到的数据库中的权限列表进行比对，如果发现在权限列表中，则授权通过，否则，不通过，抛出异常！

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
    main
      java
      resources
        mapper
          StudentMapper.xml
          SysUserMapper.xml
        spring
          applicationContext.xml
          applicationContext-shiro.xml
          springmvc.xml
        anonymousURL.properties
        commonURL.properties
        druid.properties
        log4j.properties
      webapp
        bootstrap-3.3.7-dist
        WEB-INF
          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：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
  <display-name>Shiro_04</display-name>
  <!--配置shiroFilter -->
  <filter>
    <filter-name>shiroFilter</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
    <!--此参数代表此过滤器的生命周期由serlvet管理-->
    <init-param>
      <param-name>targetFilterLifecycle</param-name>
      <param-value>true</param-value>
    </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-->
    <init-param>
      <param-name>targetBeanName</param-name>
      <param-value>shiroFilter</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>shiroFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>

  <!-- 1.加载spring所有的配置文件（不包含springmvc的配置文件） -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <!-- 注意：
        classpath与classpath*区别：
        （1）classpath:代表访问指定类路径下的资源文件
        （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。
     -->
    <param-value>classpath*:spring/applicationContext*.xml</param-value>
  </context-param>
  <!-- 2.配置spring的监听器 -->
  <listener>    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <!-- 3.配置DispatcherServlet -->
  <servlet>
    <servlet-name>springmvc</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:spring/springmvc.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>

  <!-- 4.处理中文乱码的过滤 器 -->
  <filter>
    <filter-name>characterEncoding</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
    <init-param>
      <param-name>encoding</param-name>
      <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>characterEncoding</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

1.3）pom.xml文件的配置如下：

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.zelin</groupId>
  <artifactId>Shiro_04</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>Shiro_04 Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <!-- 集中定义依赖版本号 -->
  <properties>
    <junit.version>4.12</junit.version>
    <spring.version>4.2.4.RELEASE</spring.version>
    <mybatis.version>3.2.8</mybatis.version>
    <mybatis.spring.version>1.2.2</mybatis.spring.version>
    <mybatis.paginator.version>1.2.15</mybatis.paginator.version>
    <mysql.version>5.1.32</mysql.version>
    <slf4j.version>1.6.4</slf4j.version>
    <jackson.version>2.4.2</jackson.version>
    <druid.version>1.0.9</druid.version>
    <jstl.version>1.2</jstl.version>
    <servlet-api.version>2.5</servlet-api.version>
    <jsp-api.version>2.0</jsp-api.version>
    <commons-lang3.version>3.3.2</commons-lang3.version>
    <commons-io.version>1.3.2</commons-io.version>
    <pagehelper.version>4.0.0</pagehelper.version>
    <jsqlparser.version>0.9.1</jsqlparser.version>
    <commons-fileupload.version>1.3.1</commons-fileupload.version>
  </properties>


  <dependencies>
    <!-- Apache工具组件 -->
    <dependency>
      <groupId>org.apache.commons</groupId>
```

```xml
        <artifactId>commons-lang3</artifactId>
        <version>${commons-lang3.version}</version>
      </dependency>
      <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-io</artifactId>
        <version>${commons-io.version}</version>
      </dependency>
      <!-- Jackson Json处理工具包 -->
      <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>${jackson.version}</version>
      </dependency>

      <!-- 单元测试 -->
      <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>${junit.version}</version>
      </dependency>
      <!-- 日志处理 -->
      <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>${slf4j.version}</version>
      </dependency>
      <!-- Mybatis -->
      <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>${mybatis.version}</version>
      </dependency>
      <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>${mybatis.spring.version}</version>
      </dependency>
      <dependency>
        <groupId>com.github.miemiedev</groupId>
        <artifactId>mybatis-paginator</artifactId>
        <version>${mybatis.paginator.version}</version>
      </dependency>
      <dependency>
        <groupId>com.github.pagehelper</groupId>
        <artifactId>pagehelper</artifactId>
        <version>${pagehelper.version}</version>
      </dependency>
      <!-- MySql -->
      <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>${mysql.version}</version>
```

```xml
    </dependency>
    <!-- 连接池 -->
    <dependency>
      <groupId>com.alibaba</groupId>
      <artifactId>druid</artifactId>
      <version>${druid.version}</version>
    </dependency>
    <!-- Spring -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-beans</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aspects</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jms</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context-support</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <!-- JSP相关 -->
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>${jstl.version}</version>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>${servlet-api.version}</version>
```

```xml
        <scope>provided</scope>
      </dependency>
      <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jsp-api</artifactId>
        <version>${jsp-api.version}</version>
        <scope>provided</scope>
      </dependency>

      <!-- 关于shiro相关 -->
      <dependency>
        <groupId>org.apache.shiro</groupId>
        <artifactId>shiro-core</artifactId>
        <version>1.3.2</version>
      </dependency>
      <dependency>
        <groupId>org.apache.shiro</groupId>
        <artifactId>shiro-web</artifactId>
        <version>1.3.2</version>
      </dependency>
      <dependency>
        <groupId>org.apache.shiro</groupId>
        <artifactId>shiro-spring</artifactId>
        <version>1.3.2</version>
      </dependency>
      <dependency>
        <groupId>org.apache.shiro</groupId>
        <artifactId>shiro-ehcache</artifactId>
        <version>1.3.2</version>
      </dependency>
      <dependency>
        <groupId>tk.mybatis</groupId>
        <artifactId>mapper</artifactId>
        <version>4.1.5</version>
      </dependency>
    </dependencies>
    <build>
      <finalName>${project.artifactId}</finalName>
      <plugins>
        <!-- 资源文件拷贝插件 -->
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-resources-plugin</artifactId>
          <version>2.7</version>
          <configuration>
            <encoding>UTF-8</encoding>
          </configuration>
        </plugin>
      </plugins>
      <pluginManagement>
        <plugins>
          <!-- 配置Tomcat插件 -->
          <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <configuration>
              <contextReloadable>true</contextReloadable>
              <port>9000</port>
              <path>/</path>
            </configuration>
          </plugin>
        </plugins>
      </pluginManagement>
    </build>
  </project>
```

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       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">
    <!--0.定义spring的包扫描器-->
    <context:component-scan base-package="com.zelin"/>
    <!--1.引入属性文件-->
    <context:property-placeholder location="classpath*:druid.properties"/>
    <!--2.配置数据源-->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="${jdbc.driver}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>
    <!--3.配置sqlSessionFactoryBean-->
    <bean class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="typeAliasesPackage" value="com.zelin.pojo"/>
        <property name="plugins">
            <list>
                <bean class="com.github.pagehelper.PageHelper">
                    <property name="properties">
                        <value>
                            dialect=mysql
                        </value>
                    </property>
                </bean>
            </list>
        </property>
        <property name="mapperLocations" value="classpath*:mapper/*.xml"/>
    </bean>
    <!--4.接口扫描包-->
    <bean class="tk.mybatis.spring.mapper.MapperScannerConfigurer">
```

```xml
        <property name="basePackage" value="com.zelin.mapper"/>
    </bean>
</beans>
```

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

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       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">
    <!--1.定义shiro的工厂bean对象-->
    <bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
        <!--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"/>
        <!--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-->
        <property name="successUrl" value="/user/listmenu.do"/>
        <!--1.3)配置认证失败后跳转的页面-->
        <property name="unauthorizedUrl" value="/resure.html"/>
        <!--1.4)引用一个securityManager对象-->
        <property name="securityManager" ref="mySecurityManager"/>
        <!--1.5)定义过滤器链-->
        <!--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。-->
        <!--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。-->
        <!--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。-->
        <!--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-->
        <!--Logout:代表退出过滤器，此url的action可以不定义，由shiro自动销毁session-->
        <property name="filterChainDefinitions">
            <value>
                <!-- 配置可以匿名访问的资源 -->
                /bootstrap-3.3.7-dist/** = anon
                <!-- 退出系统 -->
                /logout.do = logout
                <!-- 对验证码进行放行 -->
                /validatecode.jsp = anon

                <!--对用户进行权限拦截(授权)-->
                <!--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)
                    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
                    <property name="unauthorizedUrl" value="/resure.html"/>属性指定的页面。
                -->
                /student/tolist.do = perms["student:tolist"]
                /student/add.do = perms["student:create"]
                <!--
                    /** = 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
                -->
                /** = authc
            </value>
        </property>
    </bean>
    <!--2.定义SecurityManager对象-->
    <bean id="mySecurityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
        <property name="realm" ref="customRealm"/>
    </bean>
    <!--3.构造自定义Realm对象-->
    <bean id="customRealm" class="com.zelin.realm.CustomRealm">
        <property name="credentialsMatcher" ref="credentialsMatcher"/>
    </bean>
    <!--4.定义凭证匹配器-->
    <bean id="credentialsMatcher" class="org.apache.shiro.authc.credential.HashedCredentialsMatcher">
        <!--4.1)配置加密算法-->
        <property name="hashAlgorithmName" value="md5"/>
        <!--4.2）配置加密次数-->
        <property name="hashIterations" value="1"/>
    </bean>
</beans>
```

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

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

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])

```java
package com.zelin.realm;

import com.zelin.pojo.SysPermission;
import com.zelin.pojo.SysUser;
import com.zelin.service.SysUserService;
import org.apache.shiro.authc.AuthenticationException;
import org.apache.shiro.authc.AuthenticationInfo;
import org.apache.shiro.authc.AuthenticationToken;
import org.apache.shiro.authc.SimpleAuthenticationInfo;
import org.apache.shiro.authz.AuthorizationInfo;
import org.apache.shiro.authz.SimpleAuthorizationInfo;
import org.apache.shiro.realm.AuthorizingRealm;
import org.apache.shiro.subject.PrincipalCollection;
import org.apache.shiro.util.ByteSource;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

/**
 * @Author: Feng.Wang
 * @Company: Zelin.ShenZhen
 * @Description:
 * @Date: Create in 2019/4/16 09:21
 */

public class CustomRealm extends AuthorizingRealm {
    @Autowired
    private SysUserService userService;
    //用户授权
    @Override
    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals)
{
        //1.取得主身份信息
        SysUser user = (SysUser) principals.getPrimaryPrincipal();
        List<SysPermission> permissions = null;
        if(null != user){
            //2.取得此用户的权限列表
            permissions = userService.findPermissionsByUserCode(user.getUsercode());
            if(permissions != null && permissions.size() > 0){
                user.setPermissions(permissions);
            }
        }
        //3.定义授权对象
        SimpleAuthorizationInfo authorizationInfo = new SimpleAuthorizationInfo();
        //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<>();
        //5.转换上面的List<SysPermission>对象为List<String>对象
        for (SysPermission permission : permissions) {
            permissionList.add(permission.getPercode());
        }
        //6.将权限码列表与当前授权对象关联
        authorizationInfo.addStringPermissions(permissionList);
        //7.返回授权对象
        return authorizationInfo;
    }

    //用户认证
    @Override
    protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {
        //1.得到主身份信息
        String usercode = (String) token.getPrincipal();
        //2.如果用户名存在
        if(usercode != null){
            SysUser user = userService.findUserByUserCode(usercode);
            if(user == null) return null;
            //2.1)如果不为null，就查询其对应的菜单列表
            List<SysPermission> menus = userService.findMenusByUserCode(usercode);
            user.setMenus(menus);
            //2.2)取出数据库中的密码及加盐值
            String password = user.getPassword();
            String salt = user.getSalt();
            return new SimpleAuthenticationInfo(user,password, ByteSource.Util.bytes(salt),"aaa");
        }
       return null;
    }
}

```

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;
```

```java
    }
    public void setMessage(String message) {
        this.message = message;
    }
    public MyException() {
        super();
    }
    public MyException(String message) {
        super();
        this.message = message;
    }
}
```

2.3）注解完成全局异常处理：

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

```

2.4）登录处理器login.do：

```java
/**
 * @Author: Feng.Wang
 * @Company: Zelin.ShenZhen
 * @Description:
 * @Date: Create in 2019/4/16 09:39
 */
@Controller
public class LogController {
    @RequestMapping("/login")
    //默认情况下，shiro的用户认证工作由FormAuthenticationFilter(authc)过滤器完成，
    //当出现异常时，shiro会将异常对象的类名放到以shiroLoginFailure为key的request
    //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
    public String login(HttpServletRequest request, HttpServletResponse response)
throws Exception {
        //1.得到异常的名称
        String exceptionName = (String) request.getAttribute("shiroLoginFailure");
        //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
        //2.1)判断是否为null
        if(StringUtils.isNotEmpty(exceptionName)){
            if(UnknownAccountException.class.getName().equals(exceptionName)){
                throw new MyException("账户异常!");
            }else
if(IncorrectCredentialsException.class.getName().equals(exceptionName)){
                throw new MyException("用户名或密码输入有误!");
            }else if(UnauthorizedException.class.getName().equals(exceptionName)){
                throw new MyException("无权限访问异常!");
            }else{
                throw new Exception();
            }
        }
        //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
        return "login";
    }
}
```

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

```jsp
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/12
  Time: 10:21
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@include file="/base.jsp"%>
<html>
<head>
    <title>用户登录</title>
    <style>
        .container{
            width: 500px;
            margin-top: 50px;
        }
        .form-signin{
            padding:5px;
        }
        .btn{
            margin-top: 20px;
        }
        .error{
            color:red;
        }
    </style>
</head>
```

```
<body>
<div class="container">
    <div class="panel panel-primary">
        <div class="panel-heading">
            <h3 class="panel-title">
                用户登录
            </h3>
        </div>
        <div class="panel-body">
        <form class="form-signin" action="${pageContext.request.contextPath}/login.do"
method="post">
            <label>用户名</label>
            <input type="text"  name="username" class="form-control" placeholder="输入
用户名" required autofocus>
            <label >密码</label>
            <input type="password" name="password" class="form-control"
placeholder="输入密码" required>
            <button class="btn btn-lg btn-primary btn-block" type="submit">登录
</button>
            <span class="error">${message}</span>
        </form>
    </div>
    </div>
</div> <!-- /container -->
</body>
</html>

```

2.6）/WEB-INF/user/listmenu.jsp列表菜单 ：

```
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2019/4/12
  Time: 10:51
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@include file="/base.jsp"%>
<style>
    .col-md-2{
        margin-top: 20px;
    }
    .list-group-item{
        text-align: center;
    }
    .col-md-10{
        padding-left: 0px;
        margin-left: -10px;
        margin-top: 20px;
    }
</style>
```

```
<html>
<head>
    <title>显示菜单</title>
    <meta charset="UTF-8">
</head>
<body>
    <div class="container">
        <%--分成左右栏--%>
        <%--1.左边显示菜单--%>
        <div class="row">
            <div class="col-md-2">
                <div class="list-group">
                    <a href="#" class="list-group-item list-group-item-danger disabled">
                        用户菜单
                    </a>
                    <c:forEach items="${menus}" var="m">
                        <a href="${m.url}" target="right" class="list-group-item tt">${m.name}</a>
                    </c:forEach>
                </div>
            </div>
            <%--2.右边显示效果--%>
            <div class="col-md-10">
                <iframe src="" width="1100" height="800" frameborder="0" name="right"></iframe>
            </div>
        </div>
    </div>
</body>
</html>
<script>
    $(function () {
        //切换样式
        $(".tt").click(function () {
           //1.将所有的list-group下的a标签下的active样式去除
            $(".list-group a").removeClass("active");
            //2.将当前点击的哪个a设置active样式
            $(this).addClass("active");
        })
    })
</script>
```

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

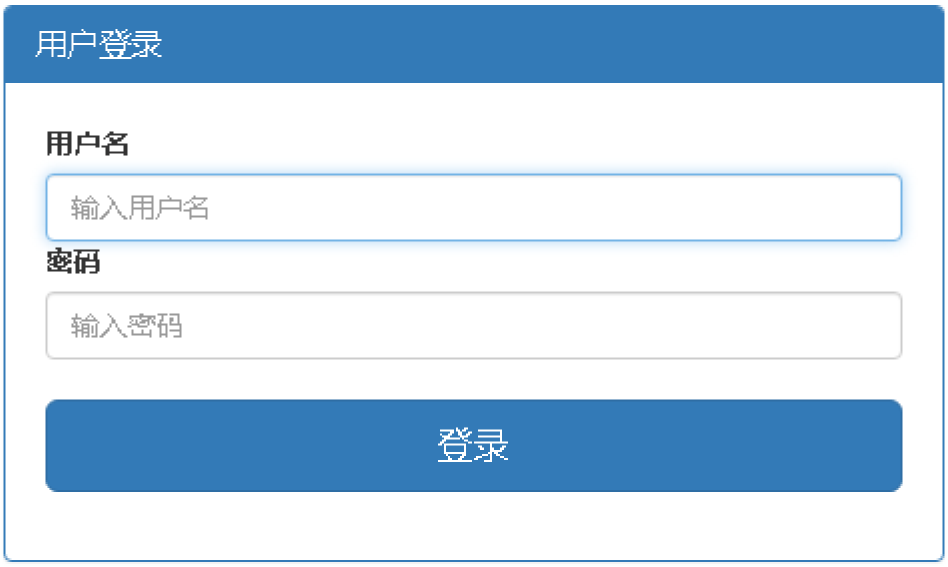
```
<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<jsp:include page="/base.jsp"/>
<head>
    <meta charset="UTF-8">
    <style>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

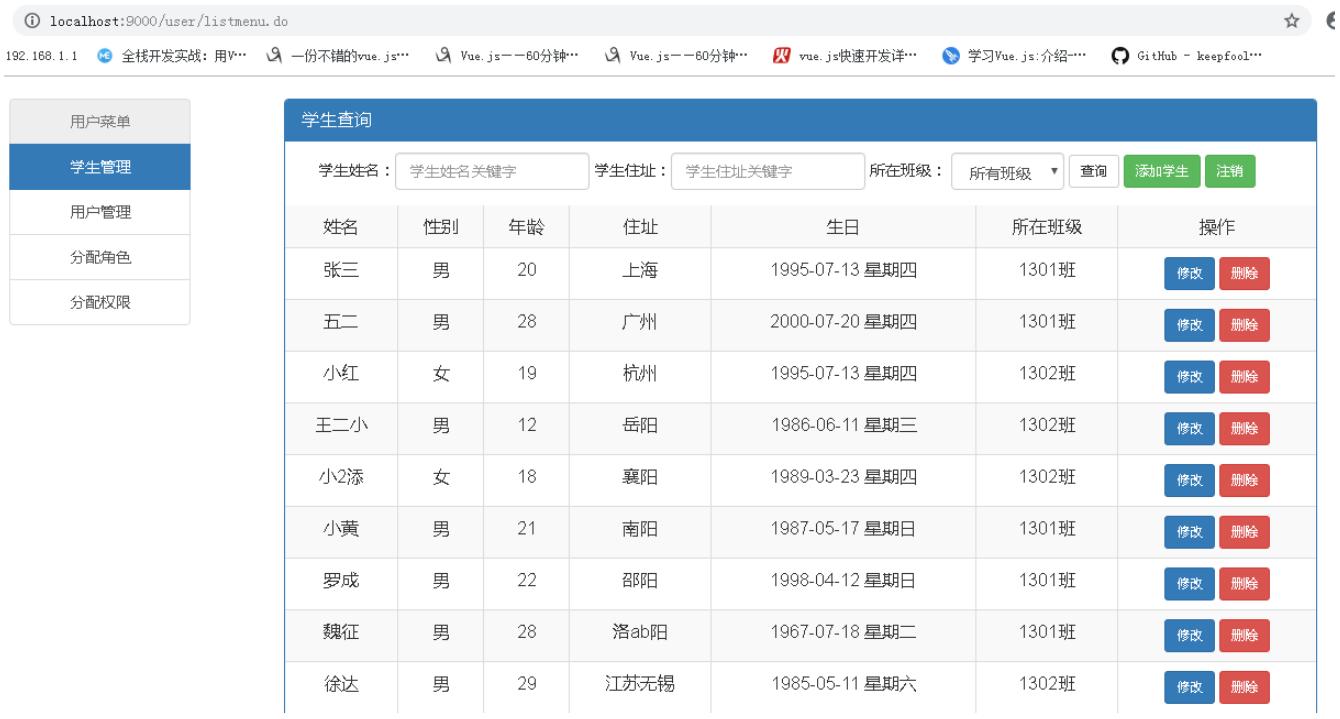
```
        .table {
            text-align: center;
        }
        .container {
            margin-top: 20px;
        }
        #form1 {
            margin-top: 10px;
            margin-left: 30px;
        }
    </style>
</head>
<body>
<div class="container">
    <div class="panel panel-primary">
        <div class="panel-heading">
            <h3 class="panel-title">学生查询</h3>
        </div>

        <form class="form-inline" id="form1" method="post"
              action="${pageContext.request.contextPath}/student/search.do">
            <input type="hidden" name="page" id="page">
            学生姓名：<input type="text" name="sname" value="${param.sname}" placeholder="学生姓名关键字" class="form-control">
            学生住址：<input type="text" name="addr" value="${param.addr}" placeholder="学生住址关键字" class="form-control">
            所在班级：
            <select name="cid" class="form-control">
                <option value="0">所有班级</option>
                <c:forEach items="${classes}" var="c">
                    <option value="${c.cid}" ${c.cid==param.cid?"selected":""}>${c.cname}</option>
                </c:forEach>
            </select>
            <input type="submit" value="查询" class="btn btn-default btn-sm">
            <input type="button" value="添加学生" class="btn btn-success btn-sm" onclick="addStudent()">
            <a href="javascript:parent.location.href = '/logout.do'" class="btn btn-success btn-sm">注销</a>
        </form>
        <table class="table table-bordered table-striped">
            <tr>
                <td>姓名</td>
                <td>性别</td>
                <td>年龄</td>
                <td>住址</td>
                <td>生日</td>
                <td>所在班级</td>
                <td>操作</td>
            </tr>
            <c:forEach items="${students}" var="stud">
                <tr>
                    <td>${stud.sname}</td>
                    <td>${stud.sex}</td>
                    <td>${stud.age}</td>
                    <td>${stud.addr}</td>
                    <td>
                        <fmt:formatDate value="${stud.birth}" pattern="yyyy-MM-dd E"/>
                    </td>
                    <td>
                            ${stud.classes.cname}
                    </td>
                    <td>
                        <a class="btn btn-primary btn-sm"

href="${pageContext.request.contextPath}/student/toupdate.do?sid=${stud.sid}">修改</a>
                        <a class="btn btn-danger btn-sm"

href="${pageContext.request.contextPath}/student/deleteBySid.do?sid=${stud.sid}"
                           onclick="return confirm('你真的要删除吗?')">删除</a>
                    </td>
                </tr>
            </c:forEach>
        </table>
    </div>
</div>
<script>
    //执行提交 表单
    function skip(i) {
        //1.对表单中的隐藏域赋值
        $("#page").val(i);
        //2.提交表单
        $("#form1").submit();
    }
    //添加学生
    function addStudent() {
        location.href = "${pageContext.request.contextPath}/student/toadd.do";
    }
</script>
</body>
</html>

```

2.8) /user/listmenu.do展示用户列表

```
/**
 * @Author: Feng.Wang
 * @Company: Zelin.ShenZhen
 * @Description:
 * @Date: Create in 2019/4/8 10:50
 */
@Controller
@RequestMapping("/student")
public class StudentController {
    @Autowired
    private StudentService studentSerivce;
    @Autowired
    private ClassesService classesService;
    int pageSize = 5;       //代表每页的大小
    //查询所有的学生
    @RequestMapping("/tolist")
    public String findAll(Model model){
        try {
            //1.查询所有的学生集合
            List<Student> students = studentSerivce.findStudents();
            System.out.println("----->" + students);
            //2.将上面的集合放到model中
            model.addAttribute("students",students);
            model.addAttribute("classes",classesService.findAll());
        } catch (Exception e) {
            e.printStackTrace();
        }
        //4.返回逻辑视图
        return "student/list";
    }
}
```

2.9）运行效果如下：





姓名	性别	年龄	住址	生日	所在班级	操作
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
小2添	女	18	襄阳	1989-03-23 星期四	1302班	修改 删除
小黄	男	21	南阳	1987-05-17 星期日	1301班	修改 删除
罗成	男	22	邵阳	1998-04-12 星期日	1301班	修改 删除
魏征	男	28	洛ab阳	1967-07-18 星期二	1301班	修改 删除
徐达	男	29	江苏无锡	1985-05-11 星期六	1302班	修改 删除

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）：

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

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

```
  <!--0.开启shiro对注解的支持-->
    <bean
class="org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor
">
        <property name="securityManager" ref="mySecurityManager"/>
    </bean>
```

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

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

3.4）在/user/userlist.do处理器：

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

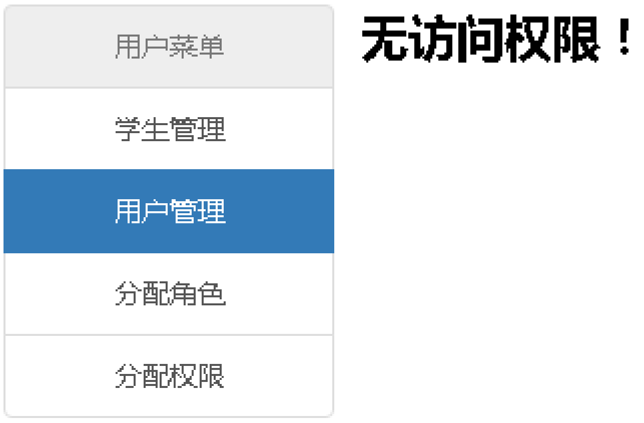
3.5）/WEB-INF/jsp/user/list.jsp用户列表页面

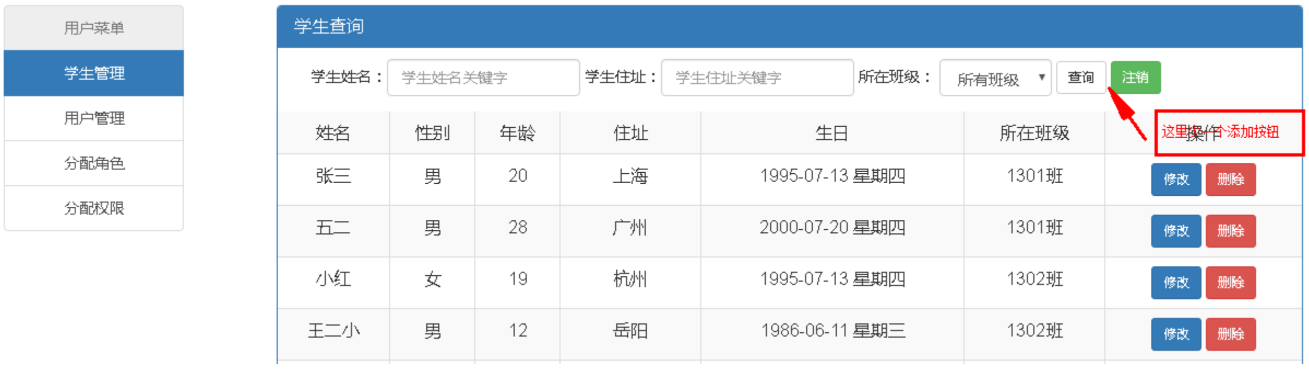
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户列表</title>
</head>
<body>
    ${users}
</body>
</html>
```

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

```
<%@taglib prefix="shiro" uri="http://shiro.apache.org/tags" %>
。。。
<%--如果有student:create这个权限，则下面的超链接就会显示出来--%>
<shiro:hasPermission name="student:createxxx">
     <input type="button" value="添加学生" class="btn btn-success btn-sm" >
</shiro:hasPermission>
```

3.7）运行效果如下：





姓名	性别	年龄	住址	生日	所在班级	操作
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

```xml
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="../config/ehcache.xsd">
    <!--diskStore：缓存数据持久化的目录 地址  -->
    <diskStore path="e:\develop\ehcache" />
    <defaultCache
            maxElementsInMemory="1000"
            maxElementsOnDisk="10000000"
            eternal="false"
            overflowToDisk="false"
            diskPersistent="false"
            timeToIdleSeconds="120"
            timeToLiveSeconds="120"
            diskExpiryThreadIntervalSeconds="120"
            memoryStoreEvictionPolicy="LRU">
    </defaultCache>
    <!--name: cache的名字，用来识别不同的cache，必须惟一。-->
    <!--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。-->
    <!--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。-->
```

```
</ehcache>
```

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

```
    <!--5.配置缓存管理器-->
    <bean id="cacheManager" class="org.apache.shiro.cache.ehcache.EhCacheManager">
        <property name="cacheManagerConfigFile" value="classpath:cache/shiro-cache.xml"/>
    </bean>
     <!--2.定义SecurityManager对象-->
    <bean id="mySecurityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
        <!--2.1)添加自定义realm-->
        <property name="realm" ref="customRealm"/>
        <!--2.2）添加缓存管理器(引入上面定义)-->
        <property name="cacheManager" ref="cacheManager"/>
    </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

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ page import="java.util.Random"%>
<%@ page import="java.io.OutputStream"%>
<%@ page import="java.awt.Color"%>
<%@ page import="java.awt.Font"%>
<%@ page import="java.awt.Graphics"%>
<%@ page import="java.awt.image.BufferedImage"%>
<%@ page import="javax.imageio.ImageIO"%>
<%
    int width = 60;
    int height = 32;
    //create the image
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
    Graphics g = image.getGraphics();
    // set the background color
    g.setColor(new Color(0xDCDCDC));
    g.fillRect(0, 0, width, height);
    // draw the border

    g.setColor(new Color(0x337ab7));
    g.drawRect(0, 0, width - 1, height - 1);
    // create a random instance to generate the codes
    Random rdm = new Random();
    String hash1 = Integer.toHexString(rdm.nextInt());
    System.out.print(hash1);
    // make some confusion
    for (int i = 0; i < 50; i++) {
        int x = rdm.nextInt(width);
        int y = rdm.nextInt(height);
        g.drawOval(x, y, 0, 0);
    }
    // generate a random code
    String capstr = hash1.substring(0, 4);
    //将生成的验证码存入session
    session.setAttribute("validateCode", capstr);
    g.setColor(new Color(0, 100, 0));
    g.setFont(new Font("Candara", Font.BOLD, 24));
    g.drawString(capstr, 8, 24);
    g.dispose();
    //输出图片
    response.setContentType("image/jpeg");
    out.clear();
    out = pageContext.pushBody();
    OutputStream strm = response.getOutputStream();
    ImageIO.write(image, "jpeg", strm);
    strm.close();
%>
```

5.2）自定义表单过滤器，重写其onAccessDenied()方法

```
/**
 * @Author: Feng.Wang
 * @Company: Zelin.ShenZhen
 * @Description: 自定义表单过滤器完成验证码的验证
 * @Date: Create in 2019/4/16 15:27
 */
public class CustomFormFilter extends FormAuthenticationFilter {
    @Override
    protected boolean onAccessDenied(ServletRequest request, ServletResponse response)
throws Exception {
        //1.将原始的请求与响应对象转换为基于http请求的对象
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;
        //2.从session中取出原始的验证码
        HttpSession session = req.getSession();
        String validateCode = (String) session.getAttribute("validateCode");
        //3.得到用户输入的验证码
        String validcode = req.getParameter("validcode");
        //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
        if(StringUtils.isNotEmpty(validcode) && !validateCode.equals(validcode)){
            req.setAttribute("shiroLoginFailure","validCodeError");
            return true;      //代表验证未通过
        }
        return super.onAccessDenied(request, response);
    }
}

```

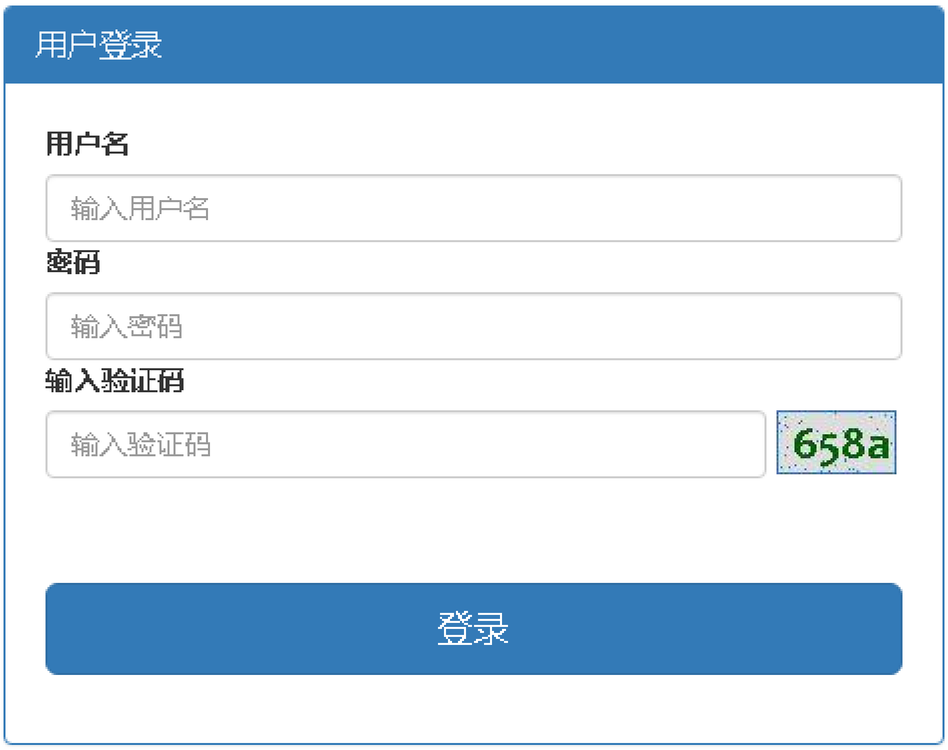
5.3）在resources/spring/applicationContext-shiro.xml文件中添加如下配置：

```xml
   <!--6.自定义表单过滤器-->
    <bean id="customFormFilter" class="com.zelin.filter.CustomFormFilter">
        <!--6.1)代表更改表单中的用户名的name属性为uname,默认为username-->
        <!--<property name="usernameParam" value="uname"/>-->
        <!--6.2)代表更改表单中的密码的name属性为pwd,默认为password-->
        <!--<property 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-->
        <!--<property name="failureKeyAttribute" value="errorKey"/>-->
    </bean>
<bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
    ...
      <!--1.6)配置自定义表单过滤器-->
        <property name="filters">
            <map>
                <!--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-->
                <entry key="authc" value-ref="customFormFilter"/>
            </map>
        </property>
</bean>
```

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

```java
@Controller
public class LogController {
    @RequestMapping("/login")
    //默认情况下，shiro的用户认证工作由FormAuthenticationFilter(authc)过滤器完成，
    //当出现异常时，shiro会将异常对象的类名放到以shiroLoginFailure为key的request
    //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
    public String login(HttpServletRequest request, HttpServletResponse response)
throws Exception {
        //1.得到异常的名称
        String exceptionName = (String) request.getAttribute("shiroLoginFailure");
        //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
        //2.1)判断是否为null
        if(StringUtils.isNotEmpty(exceptionName)){
            if(UnknownAccountException.class.getName().equals(exceptionName)){
                throw new MyException("账户异常!");
            }else
if(IncorrectCredentialsException.class.getName().equals(exceptionName)){
                throw new MyException("用户名或密码输入有误!");
            }else if(UnauthorizedException.class.getName().equals(exceptionName)){
                throw new MyException("无权限访问异常!");
            }else if("validCodeError".equals(exceptionName)){
                throw new MyException("验证码错误！");
            }else{
                throw new Exception();
            }
        }
        //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
        return "login";
    }
}
```

5.5) 测试：输入一个错误的验证码：



用户登录

用户名

输入用户名

密码

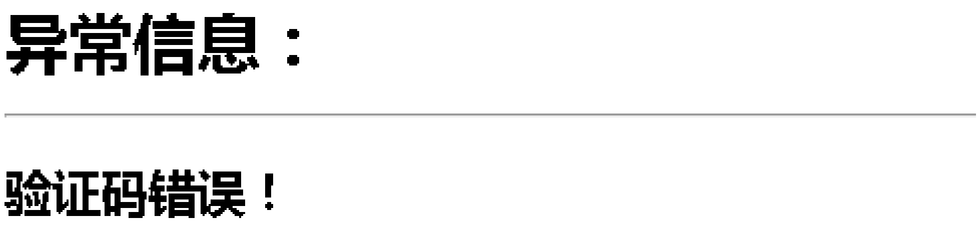
输入密码

输入验证码

输入验证码

658a

登录



异常信息：

验证码错误！