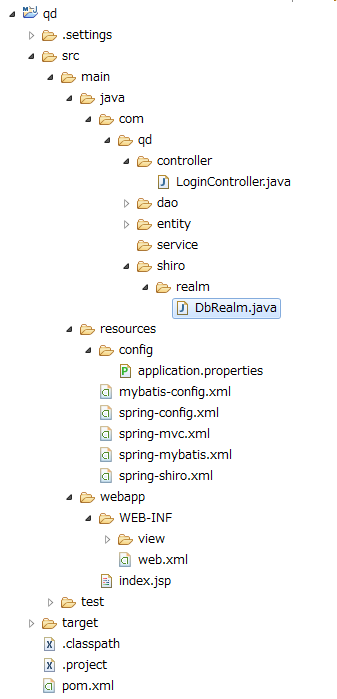
版本1 Shiro最简单的配置

首先,看项目 结构图



**web.xml**

|  |
| --- |
| **<?**xml *version*="1.0" *encoding*="UTF-8"**?>**  <web-app *xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"  *xmlns*="http://java.sun.com/xml/ns/javaee"  *xsi:schemaLocation*="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  *version*="3.0">  <display-name>spring-shiro-training</display-name>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-config.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <filter>  <filter-name>encodingFilter</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>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <filter>  <filter-name>shiroFilter</filter-name>  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>  <init-param>  <param-name>targetFilterLifecycle</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>shiroFilter</filter-name>  <url-pattern>/\*</url-pattern>  <dispatcher>REQUEST</dispatcher>  <dispatcher>FORWARD</dispatcher>  </filter-mapping>  <servlet>  <servlet-name>spring-mvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring-mvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  <async-supported>true</async-supported>  <multipart-config>  <location>/tmp</location>  <max-file-size>5242880</max-file-size>  <max-request-size>20971520</max-request-size>  <file-size-threshold>0</file-size-threshold>  </multipart-config>  </servlet>  <servlet-mapping>  <servlet-name>spring-mvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  <session-config>  <session-timeout>30</session-timeout>  </session-config>  <error-page>  <error-code>404</error-code>  <location>/WEB-INF/commons/errorpage/404.jsp</location>  </error-page>  <error-page>  <error-code>500</error-code>  <location>/WEB-INF/commons/errorpage/500.jsp</location>  </error-page>  </web-app> |

**spring-config.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:aop*="http://www.springframework.org/schema/aop" *xmlns:cache*="http://www.springframework.org/schema/cache"  *xsi:schemaLocation*="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.3.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop-4.3.xsd  http://www.springframework.org/schema/cache  http://www.springframework.org/schema/cache/spring-cache-4.3.xsd"  *default-autowire*="byName">  *<!-- 使用properties文件中的内容填充spring配置文件中的占位符 -->*  <bean  *class*="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  <property *name*="locations">  <list>  <value>classpath:config/\*.properties</value>  </list>  </property>  </bean>  <import *resource*="classpath:spring-mybatis.xml" />  <import *resource*="classpath:spring-shiro.xml" />  </beans> |

Spring-config.xml这里的spring-mybatis.xml可以不用导入。最简版本不适用数据库。放在这也没关系。

**Spring-mvc.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"  *xmlns:aop*="http://www.springframework.org/schema/aop"  *xsi:schemaLocation*="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-4.3.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-4.3.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">    *<!-- don't handle the static resource -->*  <mvc:default-servlet-handler />    *<!-- 将指定包下的类，注册到spring -->*  <context:component-scan *base-package*="com.qd.controller"/>    *<!-- 配置结果页面 前缀和后缀 -->*  <bean *class*="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property *name*="order" *value*="10"></property>  <property *name*="prefix" *value*="/WEB-INF/view/"></property>  <property *name*="suffix" *value*=".jsp"></property>  </bean>    *<!--*  *激活mvc注解驱动，就是我们使用@ReqeustMapping这样的注解的时候，*  *当我们url请求的时候，就可以跟controller中的@requestMapping进行匹配，然后调用对应的方法了。*  *-->*  <mvc:annotation-driven/>    <mvc:annotation-driven>  <mvc:message-converters *register-defaults*="false">  *<!-- 将StringHttpMessageConverter的默认编码设为UTF-8 -->*  <bean *class*="org.springframework.http.converter.StringHttpMessageConverter">  <constructor-arg *value*="UTF-8"/>  </bean>  *<!-- json -->*  <ref *bean*="jacksonMessageConverter" />  </mvc:message-converters>  </mvc:annotation-driven>  *<!-- 这个bean，可以使用@ResponseBody的时候，直接返回一个对象，然后spring直接通过这个MessageConverter Bean将对象转换为json。免去了在代码中手动转换的步骤。 -->*  <bean *id*="jacksonMessageConverter" *class*="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">  <property *name*="supportedMediaTypes">  <list>  <value>text/html;charset=UTF-8</value>  <value>application/json;charset=UTF-8</value>  </list>  </property>  </bean>    *<!-- 使用spring servlet3.0 默认上传 -->*  *<!-- <bean id="multipartResolver" class="org.springframework.web.multipart.support.StandardServletMultipartResolver"/> -->*  *<!-- <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver"/> -->*  <bean *id*="multipartResolver" *class*="org.springframework.web.multipart.commons.CommonsMultipartResolver">  <property *name*="maxUploadSize" *value*="104857600"/>  <property *name*="maxInMemorySize" *value*="4096"/>  </bean>  </beans> |

主要用于配置spring mvc的配置文件。与shiro无关系。

Spring-mybatis.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:aop*="http://www.springframework.org/schema/aop"  *xmlns:tx*="http://www.springframework.org/schema/tx"  *xsi:schemaLocation*="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop.xsd  http://www.springframework.org/schema/tx  http://www.springframework.org/schema/tx/spring-tx.xsd">    *<!-- dataSource -->*  <bean *id*="dataSource" *class*="com.alibaba.druid.pool.DruidDataSource" *init-method*="init" *destroy-method*="close">  <property *name*="url" *value*="${db.master.url}" />  <property *name*="username" *value*="${db.master.user}" />  <property *name*="password" *value*="${db.master.password}" />  *<!-- 配置监控统计拦截的filters -->*  <property *name*="filters" *value*="mergeStat,wall,log4j2" />  <property *name*="initialSize" *value*="5" />  <property *name*="maxActive" *value*="100" />  <property *name*="minIdle" *value*="10" />  <property *name*="maxWait" *value*="60000" />  <property *name*="validationQuery" *value*="SELECT 'x'" />  <property *name*="testOnBorrow" *value*="true" />  <property *name*="testOnReturn" *value*="true" />  <property *name*="testWhileIdle" *value*="true" />  <property *name*="timeBetweenEvictionRunsMillis" *value*="60000" />  <property *name*="minEvictableIdleTimeMillis" *value*="300000" />  <property *name*="removeAbandoned" *value*="true" />  <property *name*="removeAbandonedTimeout" *value*="1800" />  <property *name*="logAbandoned" *value*="true" />  </bean>  *<!-- Spring整合Mybatis -->*  <bean *id*="sqlSessionFactory" *class*="com.baomidou.mybatisplus.spring.MybatisSqlSessionFactoryBean">  <property *name*="dataSource" *ref*="dataSource"/>  *<!-- 自动扫描Mapping.xml文件 -->*  <property *name*="mapperLocations" *value*="classpath\*:/com/qd/\*/mapper/\*.xml"></property>  <property *name*="configLocation" *value*="classpath:mybatis-config.xml"></property>  <property *name*="typeAliasesPackage" *value*="com.qd.entity"/>  <property *name*="globalConfig" *ref*="globalConfig"/>  <property *name*="plugins">  <array>  *<!-- 分页插件配置 -->*  <bean *id*="paginationInterceptor" *class*="com.baomidou.mybatisplus.plugins.PaginationInterceptor">  <property *name*="dialectType" *value*="mysql"/>  <property *name*="optimizeType" *value*="aliDruid" />  </bean>  </array>  </property>  </bean>  *<!-- MP 全局配置 -->*  <bean *id*="globalConfig" *class*="com.baomidou.mybatisplus.entity.GlobalConfiguration">  *<!--*  *AUTO->`0`("数据库ID自增")*  *INPUT->`1`(用户输入ID")*  *ID\_WORKER->`2`("全局唯一ID")*  *UUID->`3`("全局唯一ID")*  *-->*  <property *name*="idType" *value*="0"/>  *<!-- 全局表为下划线命名设置 true -->*  <property *name*="dbColumnUnderline" *value*="true"/>  </bean>  *<!-- MyBatis 动态实现 -->*  <bean *id*="mapperScannerConfigurer" *class*="org.mybatis.spring.mapper.MapperScannerConfigurer">  *<!-- 对Dao 接口动态实现，需要知道接口在哪 -->*  <property *name*="basePackage" *value*="com.qd.dao"/>  </bean>  *<!-- 事务管理 -->*  <bean *id*="transactionManager" *class*="org.springframework.jdbc.datasource.DataSourceTransactionManager">  <property *name*="dataSource" *ref*="dataSource"></property>  </bean>  *<!-- 事务注解 -->*  <tx:annotation-driven *transaction-manager*="transactionManager" *proxy-target-class*="true"/>  *<!-- 事务管理 属性 -->*  <tx:advice *id*="transactionAdvice" *transaction-manager*="transactionManager">  <tx:attributes>  <tx:method *name*="select\*" *propagation*="REQUIRED" *read-only*="true" />  <tx:method *name*="delete\*" *propagation*="REQUIRED" *rollback-for*="Exception" />  <tx:method *name*="update\*" *propagation*="REQUIRED" *rollback-for*="Exception" />  <tx:method *name*="insert\*" *propagation*="REQUIRED" *rollback-for*="Exception" />  <tx:method *name*="\*" *propagation*="REQUIRED" />  </tx:attributes>  </tx:advice>  *<!-- 配置切面 -->*  <aop:config *expose-proxy*="true" *proxy-target-class*="true">  <aop:advisor *advice-ref*="transactionAdvice" *pointcut*="execution(\* com.qd.service..\*.\*(..))"/>  </aop:config>  </beans> |

用于配置数据库连接。此版本没用到从数据库读数据。所以这个xml可以不用，当不用时，不要忘记spring-config.xml中不要导入这个文件。

**spring-shiro.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"  *xsi:schemaLocation*="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd"  *default-lazy-init*="true">  <description>Shiro安全配置</description>  *<!-- 在方法中 注入 securityManager ，进行代理控制 -->*  <bean *class*="org.springframework.beans.factory.config.MethodInvokingFactoryBean">  <property *name*="staticMethod" *value*="org.apache.shiro.SecurityUtils.setSecurityManager"/>  <property *name*="arguments" *ref*="securityManager"/>  </bean>    *<!--shiro核心，shiro所有功能都由它来管理。配置shiro必须配置securityManager-->*  <bean *id*="securityManager" *class*="org.apache.shiro.web.mgt.DefaultWebSecurityManager">  *<!--设置自定义Realm-->*  <property *name*="realm" *ref*="dbRealm"/>  </bean>    *<!-- 項目自定义的Realm -->*  <bean *id*="dbRealm" *class*="com.qd.shiro.realm.DbRealm"></bean>    *<!-- Shiro Filter -->*  <bean *id*="shiroFilter" *class*="org.apache.shiro.spring.web.ShiroFilterFactoryBean">  *<!-- 安全管理器 -->*  <property *name*="securityManager" *ref*="securityManager"/>  *<!-- 默认的登陆访问url -->*  <property *name*="loginUrl" *value*="/mvc/login"/>  *<!-- 登陆成功后跳转的url -->*  <property *name*="successUrl" *value*="/mvc/index"/>  *<!-- 没有权限跳转的url -->*  <property *name*="unauthorizedUrl" *value*="/view/unauth"/>  <property *name*="filterChainDefinitions">  <value>  *<!--*  *anon 不需要认证*  *authc 需要认证*  *user 验证通过或RememberMe登录的都可以*  *-->*  /style/\*\* = anon  /static/\*\* = anon  /backgd/system/code/\*\* = anon  /backgd/system/login/\*\* = anon  /\*\* = user  </value>  </property>  </bean>  </beans> |

此配置是springMvc使用shiro的重点配置，当前版本是最简单配置。后期版本会提供更多的自定义配置。如缓存，session共享等

**application. properties**

|  |
| --- |
| #----- web hook password ------  git.hook.pwd=  git.hook.script\_path=  #--------- database production -----------  db.master.url=jdbc:mysql://127.0.0.1:3306/mysql?useUnicode=true&characterEncoding=utf-8&zeroDateTimeBehavior=convertToNull&transformedBitIsBoolean=true&useSSL=false  db.master.user=root  db.master.password=root  #--------- database development -----------  #db.master.url=jdbc:mysql://10.2.0.236:3306/jeehelp?useUnicode=true&characterEncoding=utf-8&zeroDateTimeBehavior=convertToNull&transformedBitIsBoolean=true&useSSL=false  #db.master.user=root  #db.master.password=123456  #--------- redis --------------  redis.master.ip=localhost  redis.master.port=6379  #----------upload file path---------  upload.basedir=D:/crhtpmsfile  #最大分配的对象数  redis.pool.maxActive=1024  #最大能够保持idel状态的对象数  redis.pool.maxIdle=200  #当池内没有返回对象时，最大等待时间  redis.pool.maxWait=1000  #当调用borrow Object方法时，是否进行有效性检查  redis.pool.testOnBorrow=true  #当调用return Object方法时，是否进行有效性检查  redis.pool.testOnReturn=true |

Properties文件主要用于配合spring中的占位符一块使用。如spring的xml中使用${redis.pool.testOnReturn}的时候，就会到properties文件中找对应的键值。${}就是占位符

下面是对应的xml中用到的类。包括springmvc和shiro的

SpringMVC相关的（这里只写Controller。其他的都先不考虑。能让程序跑起来就行）

**LoginController.java**

|  |
| --- |
| package com.qd.controller;  import java.util.HashMap;  import java.util.Map;  import org.apache.shiro.SecurityUtils;  import org.apache.shiro.authc.DisabledAccountException;  import org.apache.shiro.authc.IncorrectCredentialsException;  import org.apache.shiro.authc.UnknownAccountException;  import org.apache.shiro.authc.UsernamePasswordToken;  import org.apache.shiro.subject.Subject;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.ResponseBody;  @Controller  @RequestMapping("/mvc")  public class LoginController {  @RequestMapping("/login")  @ResponseBody  public Map<String, String> login(String name,String password){  Map<String, String> map= new HashMap<String,String>();    try {  Subject subject = SecurityUtils.getSubject();  UsernamePasswordToken token = new UsernamePasswordToken(name,password);    token.setRememberMe(true);  subject.login(token);  } catch (UnknownAccountException e) {  e.printStackTrace();  map.put("result", "false");  map.put("msg", "账号不存在");  } catch (DisabledAccountException e) {  e.printStackTrace();  map.put("result", "false");  map.put("msg", "账号未启用");  } catch (IncorrectCredentialsException e) {  e.printStackTrace();  map.put("result", "false");  map.put("msg", "密码错误");  } catch (RuntimeException e) {  e.printStackTrace();  map.put("result", "false");  map.put("msg", "未知错误,请联系管理员");  }    map.put("m1", "hello");  map.put("m2", "world");  return map;  }  } |

**DbRealm.java**

|  |
| --- |
| **package** com.qd.shiro.realm;  **import** org.apache.shiro.authc.AuthenticationException;  **import** org.apache.shiro.authc.AuthenticationInfo;  **import** org.apache.shiro.authc.AuthenticationToken;  **import** org.apache.shiro.authc.IncorrectCredentialsException;  **import** org.apache.shiro.authc.SimpleAuthenticationInfo;  **import** org.apache.shiro.authc.UnknownAccountException;  **import** org.apache.shiro.authc.UsernamePasswordToken;  **import** org.apache.shiro.authz.AuthorizationInfo;  **import** org.apache.shiro.realm.AuthorizingRealm;  **import** org.apache.shiro.realm.Realm;  **import** org.apache.shiro.subject.PrincipalCollection;  /\*\*  \* 数据库Realm  \*  \* **@author** liuhb  \*  \*/  **public** **class** *DbRealm* **extends** *AuthorizingRealm* {  @Override  **public** *String* **getName**() {  **return** "db\_realm";  }  @Override  **public** **boolean** **supports**(AuthenticationToken token) {  // 仅支持UsernamePasswordToken类型的Token  **return** token **instanceof** *UsernamePasswordToken*;  }    /\*\*  \* 父类的父类AuthenticatingRealm中的抽象方法  \*/  @Override  **protected** AuthenticationInfo **doGetAuthenticationInfo**(AuthenticationToken token) {  *String* username = (*String*) token.getPrincipal(); // 得到用户名  *String* password = **new** **String**((**char**[]) token.getCredentials()); // 得到密码    **if** (!"zhang".**equals**(username)) {  **throw** **new** **UnknownAccountException**(); // 如果用户名错误  }  **if** (!"123".**equals**(password)) {  **throw** **new** **IncorrectCredentialsException**(); // 如果密码错误  }  // 如果身份认证验证成功，返回一个AuthenticationInfo实现；  **return** **new** **SimpleAuthenticationInfo**(username, password, **getName**());  }  @Override  **protected** AuthorizationInfo **doGetAuthorizationInfo**(PrincipalCollection principals) {  // **TODO** Auto-generated method stub  **return** **null**;  }  } |

**下面讲一下关于shiro在springmvc中的一个大体流程**

当点击登录页面的【登录】按钮后，进入controller的login方法后。



图中代码用来往shiro中设置登录的账号和密码。后面shiro的securityManager再进行密码验证时，就会用这个对象中保存的登录账号和密码。

当执行subject.login（token）时，就会启动shiro开始进行验证工作了。如下：

1. 调用继承的父类AuthenticatingRealm的

然后在调用自定义的DbRealm（这里为了说明原理，所以虽然叫做DbRealm，但实际没有操作数据库。）。DbRealm主要用来验证登录的用户名密码是否正确。

然后在执行这个方法中的来比较用户登录的用户名密码和系统中的用户名密码是否相同。不同就说明么被认证通过，就会跳到之前spring-shiro.xml中设置的那个页面中。认证通过就可以进入指定页面了。

打开浏览器输入: <http://localhost:8088/qd/mvc/login?name=zhang&password=123>

就可以看到,直接通过url方式模拟登陆页面登录成功,并在页面显示一段json字符