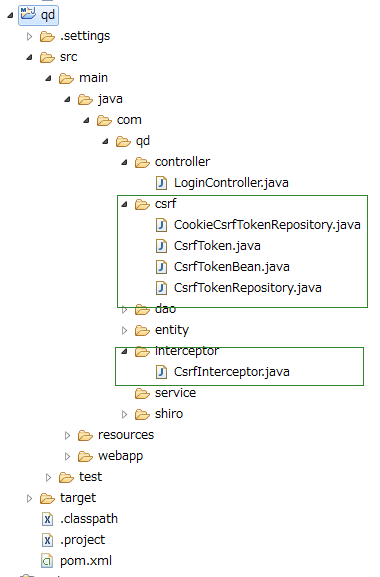
版本1 添加了防止csrf攻击代码

首先,看项目 结构图



通过上图发现,防止csrf是在最简版的基础上添加了2个包.一个是Csrf相关类,一个是拦截器interceptor.

下面说一下csrf在这里的作用:

Csrf就是跨域攻击一种手段,就是先获取到用户名和密码,然后仿造请求来登录到他人的系统.

因为我们已经有shiro来控制为认证的用户都会自动跳到登录页面.所以只要在登录时来防止crsf攻击就行.主要做法就是通过token.

应为这里用到了shiro和interceptor,两个功能上比较相似,所以下面说一下基本的流程：

首先在页面访问时先执行shiro过滤器，来判断未认证的用户无法登录系统，只能从登录页面登录（自动跳转到登录页）

然后，登录时，通过我们自定义的防止crsf的interceptor拦截器判断是不是跨域的访问。从而防止crsf攻击。

下面我们把基于版本1的基础上，改动过的文件，在下面进行详细展示：

**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" />  *<!-- =====================使用拦截器防止csrf跨域攻击配置 start ============= -->*    *<!-- 注入给CsrfInterceptor拦截器中的CsrfTokenRepository接口 -->*  <bean *class*="com.qd.csrf.CookieCsrfTokenRepository" />  *<!-- Spring csrf 拦截器 -->*  <mvc:interceptors>  <mvc:interceptor>  <mvc:mapping *path*="/mvc/login" />  <bean *class*="com.qd.interceptor.CsrfInterceptor" />  </mvc:interceptor>  </mvc:interceptors>  *<!-- ==========================使用拦截器防止csrf跨域攻击配置 end ================ -->*  *<!-- 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> |

上面绿框部分，就是用来防止csrf攻击添加的配置。

关于相关的类，直接看工程里对应的代码即可。