**简介：** Shiro 是一个 Apache Incubator 项目，旨在简化身份验证和授权。是一个很不错的安全框架。

下面记录一下shiro和Spring整合的过程的一个小示例：

Web.xml配置

|  |
| --- |
| <context-param>  <param-name>contextConfigLocation</param-name> <param-value>classpath:applicationContext.xml,classpath:spring-shiro.xml</param-value>  </context-param>  <!-- apache shiro权限 -->  <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>\*.do</url-pattern>  </filter-mapping>  <filter-mapping>  <filter-name>shiroFilter</filter-name>  <url-pattern>\*.jsp</url-pattern>  </filter-mapping> |

第一部分是将shiro的配置文件引入到web.xml中，我这里是spring-shiro.xml；

下面的是一个过滤器，过滤.do,.jsp的请求,引入shiro web.xml就配这么多。

spring-shiro.xml配置文件

|  |
| --- |
| <?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:util=*"http://www.springframework.org/schema/util"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*  *http://www.springframework.org/schema/util*  *http://www.springframework.org/schema/util/spring-util-3.0.xsd"*>  <description>Shiro 配置</description>  <bean id=*"shiroFilter"* class=*"org.apache.shiro.spring.web.ShiroFilterFactoryBean"*>  <property name=*"securityManager"* ref=*"securityManager"* />  <property name=*"loginUrl"* value=*"/login.jsp"* />  <property name=*"successUrl"* value=*"/login.jsp"* />  <property name=*"unauthorizedUrl"* value=*"/error/noperms.jsp"* />  <property name=*"filterChainDefinitions"*>  <value>  /login.jsp\* = anon  /login.do\* = anon  /index.jsp\*= anon  /error/noperms.jsp\*= anon  /\*.jsp\* = authc  /\*.do\* = authc  </value>  </property>  </bean>  <bean id=*"securityManager"* class=*"org.apache.shiro.web.mgt.DefaultWebSecurityManager"*>  <!--设置自定义realm -->  <property name=*"realm"* ref=*"monitorRealm"* />  </bean>  <bean id=*"lifecycleBeanPostProcessor"* class=*"org.apache.shiro.spring.LifecycleBeanPostProcessor"* />  <!--自定义Realm 继承自AuthorizingRealm -->  <bean id=*"monitorRealm"* class=*"com.springmvc.service.MonitorRealm"*></bean>  <!-- securityManager -->  <bean  class=*"org.springframework.beans.factory.config.MethodInvokingFactoryBean"*>  <property name=*"staticMethod"*  value=*"org.apache.shiro.SecurityUtils.setSecurityManager"* />  <property name=*"arguments"* ref=*"securityManager"* />  </bean>  <!-- Enable Shiro Annotations for Spring-configured beans. Only run after -->  <!-- the lifecycleBeanProcessor has run: -->  <bean  class=*"org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator"*  depends-on=*"lifecycleBeanPostProcessor"* />  <bean  class=*"org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor"*>  <property name=*"securityManager"* ref=*"securityManager"* />  </bean>  </beans> |

这里不想说太多，我这里没什么特别的，很多都是用的shiro自己的原有的，只有一个realm是我自己定义的，可能自己定义的会更好用点吧，如果不用自己定义的就用shiro的jdbcrealm，我自定义的也是跟jdbcrealm差不多的，后面我们再说我的自定义realm。另外就是给页面指定过滤器

|  |
| --- |
| /login.jsp\* = anon  /login.do\* = anon  /index.jsp\*= anon  /error/noperms.jsp\*= anon  /\*.jsp\* = authc  /\*.do\* = authc |

Anon：不指定过滤器。

Authc:验证，这些页面必须验证后才能访问，也就是我们说的登录后才能访问。

这里还有其他的过滤器，我没用，比如说授权，这个比较重要，但是这个过滤器有个不好的地方，就是要带一个参数，所以如果配在这里就不是很合适，因为每个页面，或是.do的权限不一样，而我们也没法事先知道他需要什么权限。所以这里不配，我们在代码中再授权。这里.do和.jsp后面的\*表示参数，比如login.jsp?main这种，是为了匹配这种。好行了，继续往下吧。

**验证：**

验证我们就弄一个登录页面，然后提交到后台的action

|  |
| --- |
| @RequestMapping(params = "main")  **public** ModelAndView login(User user,HttpSession session, HttpServletRequest request) {  ModelAndView modelView = **new** ModelAndView();  Subject currentUser = SecurityUtils.*getSubject*();  UsernamePasswordToken token = **new** UsernamePasswordToken(  user.getUsercode(), EncryptUtils.*encryptMD5*(user.getPassword()));  token.setRememberMe(**true**);  **try** {  currentUser.login(token);  } **catch** (AuthenticationException e) {  modelView.addObject("message", "login errors");  modelView.setViewName("/login");  e.printStackTrace();    }  **if**(currentUser.isAuthenticated()){    session.setAttribute("userinfo", user);  modelView.setViewName("/main");  }**else**{  modelView.addObject("message", "login errors");  modelView.setViewName("/login");  }  **return** modelView;  } |

这里我用的是spring MVC,你不用管他用什么mvc，我们只要知道，前台.do登录以后进入这个方法就行

|  |
| --- |
| Subject currentUser = SecurityUtils.*getSubject*() |

;就是代表当前的用户。

|  |
| --- |
| UsernamePasswordToken token = **new** UsernamePasswordToken( user.getUsercode(),EncryptUtils.*encryptMD5*(user.getPassword())); |

这里的token大家叫他令牌，也就相当于一张表格，你要去验证，你就得填个表，里面写好用户名密码，交给公安局的同志给你验证。

|  |
| --- |
| currentUser.login(token); |

这句是提交申请，验证能不能通过，也就是交给公安局同志了。这里会回调reaml里的一个方法

|  |
| --- |
| protected AuthenticationInfo doGetAuthenticationInfo（） |

验证是否通过

|  |
| --- |
| **if**(currentUser.isAuthenticated()) |

通过则转到你的页面，不通过到login页面并返回错误信息。

现在我们看看我们的reaml类，这是一个自定义的realm，

|  |
| --- |
| @Service("monitorRealm")  **public** **class** MonitorRealm **extends** AuthorizingRealm {  /\*  \* @Autowired UserService userService;  \*  \* @Autowired RoleService roleService;  \*  \* @Autowired LoginLogService loginLogService;  \*/  **public** MonitorRealm() {  **super**();  }  @Override  **protected** AuthorizationInfo doGetAuthorizationInfo(  PrincipalCollection principals) {  /\* 这里编写授权代码 \*/  Set<String> roleNames = **new** HashSet<String>();  Set<String> permissions = **new** HashSet<String>();  roleNames.add("admin");  permissions.add("user.do?myjsp");  permissions.add("login.do?main");  permissions.add("login.do?logout");  SimpleAuthorizationInfo info = **new** SimpleAuthorizationInfo(roleNames);  info.setStringPermissions(permissions);  **return** info;  }  @Override  **protected** AuthenticationInfo doGetAuthenticationInfo(  AuthenticationToken authcToken) **throws** AuthenticationException {  /\* 这里编写认证代码 \*/  UsernamePasswordToken token = (UsernamePasswordToken) authcToken;  // User user = securityApplication.findby(upToken.getUsername());  User user = **new** User();  user.setUsercode(token.getUsername());  user.setUserName("admin");  user.setPassword(EncryptUtils.*encryptMD5*("admin"));  // if (user != null) {  **return** **new** SimpleAuthenticationInfo(user.getUserName(),  user.getPassword(), getName());  }  **public** **void** clearCachedAuthorizationInfo(String principal) {  SimplePrincipalCollection principals = **new** SimplePrincipalCollection(  principal, getName());  clearCachedAuthorizationInfo(principals);  }  } |

这里我没有跟数据库打交道，如果要跟数据库打交道很简单，你调用一个service,service再调dao，根据用户名去打该用户的密码。用户我们可以前面的令牌也就是我说的表格来取的，我们前台提交给公安同志一个表格，里面有用户密码，但需要注意的是，我们在这里并不把表格中的用户密码路数据库中的用户密码进行比较，我们只是根据表格中的用户名把密码查出来，然后把数据库的用户密码放在一个SimpleAuthenticationInfo对象中返回即可，这位公安同志不负责验证，只负责验证的材料。我这里没有查库，伪造了一个用户密码放入了对象。密码是admin。什么时候验证，就是我们前面调currentUser.isAuthenticated()时验证，所有的材料都全了，需要验证的时候就调一下这个方法就可以了。我们前面spring里配了这个

|  |
| --- |
| /\*.jsp\* = authc  /\*.do\* = authc |

你配了authc过滤器，shiro会自动调currentUser.isAuthenticated()这个方法，没有登录的将被返回

|  |
| --- |
| <property name=*"unauthorizedUrl"* value=*"/error/noperms.jsp"* /> |

配置的页面。

好到这里登录就算是完成了。完成了登录下面就是要授权了，我们已经登录系统，我进入系统总要做点什么吧，比如这个系统就是一个公司的话，我现在已经进入公司了，如果我要进办公室，还得要授权(假如有门禁的话)刷卡。这们就里就是访问某个页面或是某个.do，

**授权：**

因为前面我们只配了验证过滤器，现在已经登录系统，如果我们请求一个\*.do的话就会来到后台的action，我们授权也将在这里授。

Java代码

|  |
| --- |
| @Controller  @RequestMapping(value="user")  **public** **class** UserController {    /\*\*  \* 跳转到myjsp页面  \*  \* **@return**  \*/  @RequestMapping(params = "myjsp")  **public** String home() {  Subject currentUser = SecurityUtils.*getSubject*();  **if**(currentUser.isPermitted("user.do?myjsp")){  **return** "/my";  }**else**{  **return** "error/noperms";  }  }  } |

我一直都说action，其实spring mvc里不再是action了，叫controller，我们这里的home方法的访问路径是user.do?myjsp，也就是我们登录系统后请求一个这个方法user.do?myjsqp,转到home方法后，我要看他有没有权限访问此方法，我就用下面的代码

|  |
| --- |
| Subject currentUser = SecurityUtils.*getSubject*();  currentUser.isPermitted("user.do?myjsp"); |

首先得到当前的用户，再看此用户是否有权访问user.do?myjsp,参数就是权限，这里后台数据库就会有这么一个权限，权限表中的权限地址就是user.do?myjsp,例如我们一般的系统左边是一棵功能菜单树，树的结点会有一个url链接，这个链接就是在权限表中。当然可能前面还会一个http:\\什么的。反正这里也跟后台的权限表中的地址一致就行了，shiro他是如何授权的。一样的你调用currentUser.isPermitted("user.do?myjsp");此方法后会回调realm中的**protected** AuthorizationInfo doGetAuthorizationInfo( PrincipalCollection principals)方法，这个reaml类是非常重要的。这个类上面已经给出了，我们看看他是如何授权的。因为我没有连数据库，也是伪造了一个权限。如果是连数据库也很简单，用户表，角色表，权限表这三个表是有关联的，我们根据用户名就能查出此用户拥有的角色和所有的权限。

|  |
| --- |
| Set<String> roleNames = **new** HashSet<String>();  Set<String> permissions = **new** HashSet<String>();  roleNames.add("admin");  permissions.add("user.do?myjsp");  permissions.add("login.do?main");  permissions.add("login.do?logout");  SimpleAuthorizationInfo info = **new** SimpleAuthorizationInfo(roleNames);  info.setStringPermissions(permissions);  **return** info; |

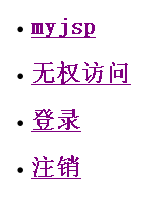
最后构造一个对象并把权限给它就OK拉。如果是数据库查出来的，直接我的字符串替成你查出来的就行了。这样在你的controller中根据权限返回到指定的页面。

|  |
| --- |
| **if**(currentUser.isPermitted("user.do?myjsp")){  **return** "/my";  }**else**{  **return** "error/noperms";  } |

没有权限就返回到没有权限的页面。那么整个权限管理系统就算是差不多了，当然还有页面标签没说，这部分不是难点，自己找找资料吧，源码我整整一并奉上。。。。。。

看看图吧





Myjsp是有权访问的，无权访问将会跳到下面的页面

