1. @Controller
2. @RequestMapping(value = "/roll")
3. **public** **class** RollLoginController {
5. @RequestMapping(value = "/login", method = RequestMethod.POST)
6. @ResponseBody
7. **public** String login(@RequestParam("username") String username,
8. @RequestParam("password") String password) {
10. Subject currentUser = SecurityUtils.getSubject();
11. String result = "login";
12. **if** (!currentUser.isAuthenticated()) {
13. result = login(currentUser,username,password);
14. }**else**{//重复登录
15. ShiroUser shiroUser = (ShiroUser) currentUser.getPrincipal();
16. **if**(!shiroUser.getLoginName().equalsIgnoreCase(username)){//如果登录名不同
17. currentUser.logout();
18. result = login(currentUser,username,password);
19. }
20. }
21. **return** result;
22. }
24. **private** String login(Subject currentUser,String username,String password){
25. String result = "login";
26. UsernamePasswordToken token = **new** UsernamePasswordToken(username,
27. password);
28. token.setRememberMe(**false**);
29. **try** {
30. currentUser.login(token);
31. result = "success";
32. } **catch** (UnknownAccountException uae) {
33. result = "failure";
34. } **catch** (IncorrectCredentialsException ice) {
35. result = "failure";
36. } **catch** (LockedAccountException lae) {
37. result = "failure";
38. } **catch** (AuthenticationException ae) {
39. result = "failure";
40. }
41. **return** result;
42. }
44. @RequestMapping(value = "/logout", method = RequestMethod.POST)
45. @ResponseBody
46. **public** String logout() {
48. Subject currentUser = SecurityUtils.getSubject();
49. String result = "logout";
50. currentUser.logout();
51. **return** result;
52. }
54. @RequestMapping(value = "/chklogin", method = RequestMethod.POST)
55. @ResponseBody
56. **public** String chkLogin() {
58. Subject currentUser = SecurityUtils.getSubject();
60. **if** (!currentUser.isAuthenticated()) {
61. **return** "false";
62. }
64. **return** "true";
65. }
67. }

applicationContext-shiro.xml

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd"
4. default-lazy-init="true"**>**
6. **<description>**Shiro Configuration**</description>**
8. <!-- Shiro's main business-tier object for web-enabled applications -->
9. **<bean** id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager" depends-on="userDao,groupDao"**>**
10. **<property** name="realm" ref="shiroDbRealm" **/>**
11. **<property** name="cacheManager" ref="cacheManager" **/>**
12. **</bean>**
14. <!-- 項目自定义的Realm -->
15. **<bean** id="shiroDbRealm" class="org.springside.examples.miniweb.service.account.ShiroDbRealm"**>**
16. **<property** name="accountManager" ref="accountManager"**/>**
17. **</bean>**
19. <!-- Shiro Filter -->
20. **<bean** id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean"**>**
21. **<property** name="securityManager" ref="securityManager" **/>**
22. **<property** name="loginUrl" value="/login" **/>**
23. **<property** name="successUrl" value="/account/user/" **/>**
24. **<property** name="filterChainDefinitions"**>**
25. **<value>**
26. /rollindex = anon
27. /index.html = anon
28. /login = authc
29. /logout = logout
30. /static/\*\* = anon
31. /admin/\*\* = user
32. /account/\*\* = user
33. **</value>**
34. **</property>**
35. **</bean>**
37. <!-- 用户授权信息Cache -->
38. **<bean** id="cacheManager" class="org.apache.shiro.cache.MemoryConstrainedCacheManager" **/>**
40. <!-- 保证实现了Shiro内部lifecycle函数的bean执行 -->
41. **<bean** id="lifecycleBeanPostProcessor" class="org.apache.shiro.spring.LifecycleBeanPostProcessor"**/>**
43. <!-- AOP式方法级权限检查  -->
44. **<bean** class="org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator" depends-on="lifecycleBeanPostProcessor"**>**
45. **<property** name="proxyTargetClass" value="true" **/>**
46. **</bean>**
48. **<bean** class="org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor"**>**
49. **<property** name="securityManager" ref="securityManager"**/>**
50. **</bean>**
51. **</beans>**
52. ShiroDbRealm.java
53. **public** **class** ShiroDbRealm **extends** AuthorizingRealm {
55. **private** **static** Logger logger = LoggerFactory.getLogger(ShiroDbRealm.**class**);
57. **private** **static** **final** String ALGORITHM = "MD5";
58. **private** AccountManager accountManager;
60. /\*\*
61. \* 认证回调函数, 登录时调用.
62. \*/
63. @Override
64. **protected** AuthenticationInfo doGetAuthenticationInfo(
65. AuthenticationToken authcToken) **throws** AuthenticationException {
66. UsernamePasswordToken token = (UsernamePasswordToken) authcToken;
67. User user = accountManager.findUserByLoginName(token.getUsername());
68. **if** (user != **null**) {
69. **return** **new** SimpleAuthenticationInfo(**new** ShiroUser(
70. user.getName(), user.getRealName()), user.getPassword(),
71. getName());
72. } **else** {
73. **return** **null**;
74. }
75. }
77. /\*\*
78. \* 授权查询回调函数, 进行鉴权但缓存中无用户的授权信息时调用.
79. \*/
80. @Override
81. **protected** AuthorizationInfo doGetAuthorizationInfo(
82. PrincipalCollection principals) {
83. ShiroUser shiroUser = (ShiroUser) principals.fromRealm(getName())
84. .iterator().next();
85. User user = accountManager
86. .findUserByLoginName(shiroUser.getLoginName());
87. //基于Permission的权限信息
88. **if** (user != **null**) {
89. SimpleAuthorizationInfo info = **new** SimpleAuthorizationInfo();
90. **for** (Role role : user.getRoles()) {
91. **for**(Permission permission: role.getPermisssions()){
92. info.addStringPermission(permission.getValue());
93. }
94. }
95. **return** info;
96. } **else** {
97. **return** **null**;
98. }
99. }
101. /\*\*
102. \* 更新用户授权信息缓存.
103. \*/
104. **public** **void** clearCachedAuthorizationInfo(String principal) {
105. SimplePrincipalCollection principals = **new** SimplePrincipalCollection(
106. principal, getName());
107. clearCachedAuthorizationInfo(principals);
108. }
110. /\*\*
111. \* 清除所有用户授权信息缓存.
112. \*/
113. **public** **void** clearAllCachedAuthorizationInfo() {
114. Cache<Object, AuthorizationInfo> cache = getAuthorizationCache();
115. **if** (cache != **null**) {
116. **for** (Object key : cache.keys()) {
117. cache.remove(key);
118. }
119. }
120. }
122. @Autowired
123. **public** **void** setAccountManager(AccountManager accountManager) {
124. **this**.accountManager = accountManager;
125. }
127. **public** String encrypt(String plainText) {
128. String result = "";
129. **byte**[] hashPassword = **null**;
130. **try** {
131. hashPassword = Digests.md5(**new** ByteArrayInputStream(plainText
132. .getBytes()));
133. } **catch** (IOException e) {
134. e.printStackTrace();
135. }
136. result = Encodes.encodeHex(hashPassword);
137. **return** result;
139. }
141. @PostConstruct
142. **public** **void** initCredentialsMatcher() {//MD5加密
143. HashedCredentialsMatcher matcher = **new** HashedCredentialsMatcher(
144. ALGORITHM);
145. setCredentialsMatcher(matcher);
146. }
148. /\*\*
149. \* 自定义Authentication对象，使得Subject除了携带用户的登录名外还可以携带更多信息.
150. \*/
151. **public** **static** **class** ShiroUser **implements** Serializable {
153. **private** **static** **final** **long** serialVersionUID = -1748602382963711884L;
154. **private** String loginName;
155. **private** String name;
157. **public** ShiroUser(String loginName, String name) {
158. **this**.loginName = loginName;
159. **this**.name = name;
160. }
162. **public** String getLoginName() {
163. **return** loginName;
164. }
166. /\*\*
167. \* 本函数输出将作为默认的<shiro:principal/>输出.
168. \*/
169. @Override
170. **public** String toString() {
171. **return** loginName;
172. }
174. **public** String getName() {
175. **return** name;
176. }
177. }
178. }