一、基本概念

Realm

Realm 是一个能够访问应用程序特定的安全数据（如用户，角色和权限）的组件。它可以被看作是一个特定

安全的DAO（Data Access Object）。Realm 将这些应用程序特定的数据转换成Shiro 能够理解的格式，这样Shiro

反过来能够提供一个单一的易于理解的Subject 编程API，无论有多少数据源存在或无论你的数据是什么样的

应用程序特定的格式。

Realm 通常和数据源是一对一的对应关系，如关系数据库，LDAP 目录，文件系统，或其他类似资源。因此，

Realm 接口的实现使用数据源特定的API 来展示授权数据（角色，权限等），如JDBC，文件IO，Hibernate 或

JPA，或其他数据访问API。

二、

三、集成Spring的Web配置

(1) xml中配置一个过滤器

*<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>*

*</filter-mapping>*

(2) applicationContext-shiro.xml

*<!-- Shiro Filter -->*

*<bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">*

*<property name="securityManager" ref="securityManager" />*

*<property name="loginUrl" value="/login" />*

*<property name="successUrl" value="/" />*

*<property name="filterChainDefinitions">*

*<value>*

*/login = authc*

*/logout = logout*

*/static/\*\* = anon*

*/api/\*\* = anon*

*/register/\*\* = anon*

*/admin/\*\* = roles[admin]*

*/\*\* = user*

*</value>*

*</property>*

*</bean>*

*<!-- Shiro's main business-tier object for web-enabled applications -->*

*<bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">*

*<property name="realm" ref="shiroDbRealm" />*

*<property name="cacheManager" ref="shiroEhcacheManager" />*

*</bean>*

*<!-- 項目自定义的Realm, 所有accountService依赖的dao都需要用depends-on声明 -->*

*<bean id="shiroDbRealm" class="org.springside.examples.quickstart.service.account.ShiroDbRealm">*

*<property name="accountService" ref="accountService"/>*

*</bean>*

*<!-- 用户授权信息Cache, 采用EhCache -->*

*<bean id="shiroEhcacheManager" class="org.apache.shiro.cache.ehcache.EhCacheManager">*

*<property name="cacheManagerConfigFile" value="classpath:ehcache/ehcache-shiro.xml"/>*

*</bean>*

(3)

public class ShiroDbRealm extends AuthorizingRealm {

protected AccountService accountService;

/\*\*

\* 认证回调函数,登录时调用.

\*/

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken authcToken) throws AuthenticationException {

UsernamePasswordToken token = (UsernamePasswordToken) authcToken;

User user = accountService.findUserByLoginName(token.getUsername());

if (user != null) {

byte[] salt = Encodes.decodeHex(user.getSalt());

return new SimpleAuthenticationInfo(new ShiroUser(user.getId(), user.getLoginName(), user.getName()),

user.getPassword(), ByteSource.Util.bytes(salt), getName());

} else {

return null;

}

}

/\*\*

\* 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.

\*/

@Override

protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {

ShiroUser shiroUser = (ShiroUser) principals.getPrimaryPrincipal();

User user = accountService.findUserByLoginName(shiroUser.loginName);

SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();

info.addRoles(user.getRoleList());

return info;

}