# 1.shiro简介



Shiro可以非常容易的开发出足够好的应用，其不仅可以用在JavaSE环境，也可以用在JavaEE环境。Shiro可以帮助我们完成：认证、授权、加密、会话管理、与Web集成、缓存等。这不就是我们想要的嘛，而且Shiro的API也是非常简单；其基本功能点如下图所示：



**Authentication：**身份认证/登录，验证用户是不是拥有相应的身份；

**Authorization：**授权，即权限验证，验证某个已认证的用户是否拥有某个权限；即判断用户是否能做事情，常见的如：验证某个用户是否拥有某个角色。或者细粒度的验证某个用户对某个资源是否具有某个权限；

**Session Manager：**会话管理，即用户登录后就是一次会话，在没有退出之前，它的所有信息都在会话中；会话可以是普通JavaSE环境的，也可以是如Web环境的；

**Cryptography：**加密，保护数据的安全性，如密码加密存储到数据库，而不是明文存储；

**Web Support：**Web支持，可以非常容易的集成到Web环境；

Caching：缓存，比如用户登录后，其用户信息、拥有的角色/权限不必每次去查，这样可以提高效率；

**Concurrency：**shiro支持多线程应用的并发验证，即如在一个线程中开启另一个线程，能把权限自动传播过去；

**Testing：**提供测试支持；

**Run As：**允许一个用户假装为另一个用户（如果他们允许）的身份进行访问；

**Remember Me：**记住我，这个是非常常见的功能，即一次登录后，下次再来的话不用登录了。

# 1.用户身份登录

## 1.配置文件

[users]  
# user 'Zhang' with password '123456' and roles 'roleStudent' and 'goodguy'  
Zhang = 123456, roleStudent, goodguy  
  
[roles]  
# The 'roleStudent' role can do anything (\*) with any wc:  
roleStudent = wc:\*  
# The 'goodguy' role is allowed to 'delete' (action) the user (type) with  
# license plate 'zhaochunyu' (instance specific id)  
goodguy = user:delete:zhaochunyu

## 2.验证登录

**public class** Quickstart {  
  
 **private static final transient** Logger ***log*** = LoggerFactory.*getLogger*(Quickstart.**class**);  
   
 **public static void** main(String[] args) {  
 *//1、获取SecurityManager工厂，此处使用Ini配置文件初始化SecurityManager* Factory<SecurityManager> factory = **new** IniSecurityManagerFactory(**"classpath:hello.ini**  
 *//2、得到SecurityManager实例 并绑定给SecurityUtils* SecurityManager securityManager = factory.getInstance();  
 SecurityUtils.*setSecurityManager*(securityManager);  
  
 *// 获取当前的 Subject. 调用 SecurityUtils.getSubject();* Subject currentUser = SecurityUtils.*getSubject*();  
  
 *// 测试使用 Session  
 // 获取 Session: Subject#getSession()* Session session = currentUser.getSession();  
 session.setAttribute(**"someKey"**, **"aValue"**);  
 String value = (String) session.getAttribute(**"someKey"**);  
 **if** (value.equals(**"aValue"**)) {  
 ***log***.info(**"---> Retrieved the correct value! ["** + value + **"]"**);  
 }  
  
 *// 测试当前的用户是否已经被认证. 即是否已经登录.  
 // 调动 Subject 的 isAuthenticated()* **if** (!currentUser.isAuthenticated()) {  
 *// 把用户名和密码封装为 UsernamePasswordToken 对象* UsernamePasswordToken token = **new** UsernamePasswordToken(**"lonestarr"**, **"vespa"**);  
 *// rememberme* token.setRememberMe(**true**);  
 **try** {  
 *// 执行登录.* currentUser.login(token);  
 }

## 2. 如果信息不匹配抛出异常

*// 若没有指定的账户, 则 shiro 将会抛出 UnknownAccountException 异常.* **catch** (UnknownAccountException uae) {  
 ***log***.info(**"----> There is no user with username of "** + token.getPrincipal());  
 **return**;  
 }  
 *// 若账户存在, 但密码不匹配, 则 shiro 会抛出 IncorrectCredentialsException 异常。* **catch** (IncorrectCredentialsException ice) {  
 ***log***.info(**"----> Password for account "** + token.getPrincipal() + **" was incorrect!"**);  
 **return**;  
 }  
 *// 用户被锁定的异常 LockedAccountException* **catch** (LockedAccountException lae) {  
 ***log***.info(**"The account for username "** + token.getPrincipal() + **" is locked. "** +  
 **"Please contact your administrator to unlock it."**);  
 }  
 *// ... catch more exceptions here (maybe custom ones specific to your application?  
 // 所有认证时异常的父类.* **catch** (AuthenticationException ae) {  
 *//unexpected condition? error?* }  
 }  
 *//say who they are:  
 //print their identifying principal (in this case, a username):  
 //principals：身份，即主体的标识属性，可以是任何东西，如用户名、邮箱等，唯一即可。一个主体可以有多个principals，但只有一个Primary principals，一般是用户名/密码/手机号。* ***log***.info(**"----> User ["** + currentUser.getPrincipal() + **"] logged in successfully."**);

# 2.用户角色的认证

## 1.配置文件

[users] 用户名 =密码，角色，角色

# user 'Zhang' with password '123456' and roles 'roleStudent' and 'goodguy'  
Zhang = 123456, roleStudent, goodguy

## 2.代码

*//test a role:  
// 测试是否有某一个角色. 调用 Subject 的 hasRole 方法.***if** (currentUser.hasRole(**"roleStudent"**)) {  
 ***log***.info(**"----> May the roleStudent be with you!"**);  
} **else** {  
 ***log***.info(**"----> Hello, mere mortal."**);  
 **return**;  
}

# 3.角色权限的认证

## 1.配置文件

[roles]  
# The 'roleStudent' role can do anything (\*) with any wc:  
roleStudent = wc:\*  
# The 'goodguy' role is allowed to 'delete' (action) the user (type) with  
# license plate 'zhaochunyu' (instance specific id)  
goodguy = user:delete:zhaochunyu

## 2.代码测试权限

*// 测试用户是否具备某一个行为. 调用 Subject 的 isPermitted() 方法。***if** (currentUser.isPermitted(**"wc:testContent"**)) {  
 ***log***.info(**"----> You may use a wc ring. Use it wisely."**);  
} **else** {  
 ***log***.info(**"Sorry, wc rings are for roleStudent masters only."**);  
}  
 *// 测试用户是否具备某一个行为.***if** (currentUser.isPermitted(**"user:delete:zhaochunyu"**)) {  
 ***log***.info(**"----> You are permitted to 'drive' the winnebago with license plate (id) 'eagle5'. "** +  
 **"Here are the keys - have fun!"**);  
} **else** {  
 ***log***.info(**"Sorry, you aren't allowed to drive the 'eagle5' winnebago!"**);  
}

# 4.用户退出登录

*//all done - log out!  
// 执行登出. 调用 Subject 的 Logout() 方法. ---->true*System.***out***.println(**"---->"** + currentUser.isAuthenticated());  
*//用户退出登录*currentUser.logout();  
  
*//---->false*System.***out***.println(**"---->"** + currentUser.isAuthenticated());  
  
System.*exit*(0);

# 8.测试验证