**shiro身份认证**

认证有时也理解为“登录”。它是一个典型的三步骤过程。

1. 收集登陆用户的身份信息，称为**当事人（principal）**，以及身份的证明，称为**凭据（Credential）**。
2. 将当事人和证书提交给系统。
3. 如果提交的证书与系统期望的该用户身份（当事人）匹配，该用户就被认为是经过认证的，反之则被认为未经认证的。

这个过程的常见例子是大家都熟悉的“用户名 / 密码”组合。多数用户在登录软件系统时，通常提供自己的用户名（当事人）和他们的密码（证书）

Subject是需要通过认证的主体，principal是这个主体的唯一标识。登陆软件系统的这个人是subject，而这个人可以使用邮箱，手机号或身份证号这些唯一标识登陆。这些标识就是principal

Credentials通常是只有主体知道的秘密值, 被用作他们实际上 ' 拥有 ' 声称的身份的佐证。凭据的一些常见示例包括密码、生物识别数据 (如指纹和视网膜扫描) 以及 X.509 证书。

上述的3个步骤在程序中可以写成

//1. 收集的当事人和证书：

AuthenticationToken token =new UsernamePasswordToken(username, password);

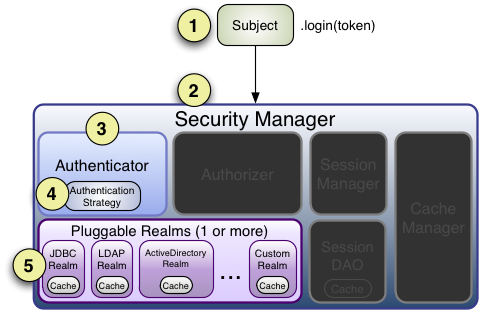
//2. 获取当前 Subject，调用此方法前程序中需有一个实例化的SecurityManager

Subject currentUser = SecurityUtils.getSubject();

//3. 登录认证：

currentUser.login(token);

等应用程序执行身份验证功能时，shiro的内部会执行如下操作



1. 应用程序使用收集到的凭据AuthenticationToken调用Subject.login（） ，
2. Subject实例委托securityManager处理认证。调用securityManager的login(token)方法
3. securityManager把认证交由自己内部Authenticator实例处理。Authenticator实例调用authenticator.authenticate(token)方法 。SecurityManager默认使用ModularRealmAuthenticator 类型的 Authenticator 。ModularRealmAuthenticator 可以处理单个realm的认证，也可以协调多个realm的认证。当处理多个realm时需要指定AuthenticationStrategy用于协调。
4. 每个Realm都会查看自己是否能识别处理用户提交的token（AuthenticationToken）。如果能够处理，将会调用realm的getAuthenticationInfo方法执行认证。

**AuthenticationStrategy**

使用多个Realm认证时，ModularRealmAuthenticator 使用内置的 AuthenticationStrategy 组件，决定认证成功的条件。例如，只有1个Realm通过，其他都失败时，认证过程算成功还是失败。是否需要所有realm通过才算成功？在所有Realm认证前和认证后，每个Realm认证前和后AuthenticationStrategy都会被调用。

AuthenticationStrategy是一个无状态组件, 在身份验证尝试期间会调用 4次 (这4个交互所需的任何必要状态都可以使用方法参数提供):

1. 在所有Realm执行认证前被调用
2. 在每个Realm执行认证前被调用
3. 在每个Realm执行认证后被调用
4. 在所有Realm执行认证后被调用

如果只配置了1个realm则不需要AuthenticationStrategy。直接调用realm认证即可

AuthenticationStrategy还负责聚合每个成功验证的结果, 并将其 "合并" 到AuthenticationInfo。AuthenticationInfo是Authenticator 实例认证后返回的结果

**Shiro是一个模块化框架。Shiro的大部分组件都是可插拔，用户可以根据自己的需要使用自定义组件。为了满足大部分需要，框架的组件提供了很多默认实现，用户可以自由的组合这些组件**

Shiro提供了3个AuthenticationStrategy的实现类

AtLeastOneSuccessfulStrategy：如果一个 (或多个) Realms成功进行身份验证, 则整个尝试被视为成功。会验证所有realms，返回所有成果验证结果

FirstSuccessfulStrategy：将只使用从第一个成功通过身份验证的领域返回的信息。所有进一步的领域将被忽略。

AllSuccessfulStrategy：所有配置的Realms都必须成功进行身份验证, 才能将整个尝试视为成功。

ModularRealmAuthenticator 默认使用 **AtLeastOneSuccessfulStrategy** 这个策略,

**前面说过shiro的核心是**SecurityManager。SecurityManager管理的组件甚至它本身都可以根据用户的需要进行配置（使用shiro默认的实现或用户根据接口的自定义实现）

[main]

authcStrategy = org.apache.shiro.authc.pam.FirstSuccessfulStrategy

securityManager.authenticator.authenticationStrategy = $authcStrategy

**当配置了多个****[Realm](http://shiro.apache.org/authentication.html" \l "realm-authentication-order)时，认证顺序可以隐式指定或显示指定**

隐式指定顺序

使用 Shiroo 的 INI 配置格式时。将按照 INI 中定义的顺序使用Realm认证

blahRealm = com.company.blah.Realm

fooRealm = com.company.foo.Realm

barRealm = com.company.another.Realm

显示指定顺序

设置securityManager.realms将忽略Realm定义的顺序。未列出的realm不会被使用

blahRealm = com.company.blah.Realm

fooRealm = com.company.foo.Realm

barRealm = com.company.another.Realm

#只使用fooRealm和barRealm这两个realm

securityManager.realms = $fooRealm, $barRealm

引用

<http://shiro.apache.org/tutorial.html>

<http://shiro.apache.org/authentication.html>

https://github.com/zhangkaitao/shiro-example