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）

SecurityManager和它管理的组件都是 JavaBeans兼容的，这使得 Shiiro 几乎可以配置任何配置格式, 如常规 Java、XML (Spring、JBoss、Guice 等)、YAML、Json、Groovy Builder 标记等。

**编程配置**

使用编程方式配置，是为应用程序提供一个可用的SecurityManager实例的最简单办法

Realm realm = //实例化一个Realm实例，realm将在以后讨论

SecurityManager securityManager = new DefaultSecurityManager(realm);

// 使SecurityManager实例在整个应用程序内可以

SecurityUtils.setSecurityManager(securityManager);

**INI文件配置**

大多数应用程序都受益于基于文本的配置, 这些配置可以独立于源代码进行修改。为了使shiro的配置在所有环境中可用，且使用最少的第三方依赖。Shiro用INI format配置SecurityManager和它的组件。

使用ini文件实例化SecurityManager。Ini资源可以从文件系统、类路径或 url 中获取资源，只需要带上不同前缀就可以file:, classpath:, url:

Factory<SecurityManager> factory = new IniSecurityManagerFactory("classpath:shiro.ini");

SecurityManager securityManager = factory.getInstance();

SecurityUtils.setSecurityManager(securityManager);

INI 配置还可以使用编程的方式通过 org.apache.shiro.config.Ini 类构造。Ini 类的功能类似于 JDK java.util.Properties 类

Ini ini = new Ini();

//配置ini实例

Factory<SecurityManager> factory = new IniSecurityManagerFactory(ini);

SecurityManager securityManager = factory.getInstance();

SecurityUtils.setSecurityManager(securityManager);

INI是由名字唯一的section组成的文件。每个section包含唯一的key/value键值对作为配置。Section内的key必须唯一，但整个文件的key不必唯一。注释行以#或;开头

[main]

# 配置应用程序的securityManager实例及其任何依赖项 (如 Realm) 的位置。

[users]

#当系统只需要很少的用户，且很少添加用户，用户不需要在运行时添加，可以在这里添加用户账户

[roles]

# 当系统只需要很少的角色，且很少添加角色，角色不需要在运行时添加，可以在这里添加角色

[urls]

#用于配置shiro自身的过滤器匹配规则

简单属性可以直接使用等号赋值

myRealm.connectionTimeout = 30000

myRealm.username = jsmith

他们等价于对象上的set方法调用

myRealm.setConnectionTimeout(30000);

myRealm.setUsername("jsmith");

需要为对象赋值时，先定义对象，然后在对象名称前加上$赋值

sha256Matcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher

myRealm.credentialsMatcher = $sha256Matcher

嵌套的属性赋值

securityManager.sessionManager.globalSessionTimeout = 1800000

由于byte arrays无法用文本格式表示，需要将它们编码成Base64或Hex格式

securityManager.rememberMeManager.cipherKey = kPH+bIxk5D2deZiIxcaaaA==

Hex格式在字符串前加上0x

securityManager.rememberMeManager.cipherKey = 0x3707344A4093822299F31D008

集合类型属性设置，使用逗号分隔

sessionListener1 = com.company.my.SessionListenerImplementation

sessionListener2 = com.company.my.other.SessionListenerImplementation

securityManager.sessionManager.sessionListeners = $sessionListener1, $sessionListener2

Map类型属性设置，使用逗号分隔键值对，key和value之间用冒号分隔

object1 = com.company.some.Class

object2 = com.company.another.Class

anObject = some.class.with.a.Map.property

anObject.mapProperty = key1:$object1, key2:$object2

注意事项

Ini文件中定义的顺序很重要。同名属性或对象会被后面定义的覆盖

myRealm = com.company.security.MyRealm

myRealm = com.company.security.DatabaseRealm

myRealm 最后是一个 com.company.security.DatabaseRealm 实例

在[main]section中可以直接设置securityManager的属性，它已经被自动实例化了。用户还可以是用自己实现的securityManager。

securityManager = com.company.security.shiro.MyCustomSecurityManager

[users] section允许定义一组少量的用户账户，通常用在程序不需要在运行的时候动态添加用户账户。格式如下

username = password, *roleName1*, *roleName2*, …, *roleNameN*

密码后可添加0到多个用户所属角色

如果希望密码是经过加密的，可以在[main]section中指定加密算法。把加密后的字符串再转换成Hex或Base64格式再添加到username =后面

[main]

sha256Matcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher

# true = hex, false = base64:

sha256Matcher.storedCredentialsHexEncoded = false

iniRealm.credentialsMatcher = $sha256Matcher

[roles] section允许定义一组少量的用户角色，通常用在程序不需要在运行的时候动态添加用户角色。格式如下

rolename = *permissionDefinition1*, *permissionDefinition2*, …, *permissionDefinitionN*

[roles]

# 'admin'使用通配符 '\*'，所以用于所有权限

admin = \*

# 'schwartz'拥有lightsaber下的所有权限

schwartz = lightsaber:\*

goodguy = winnebago:drive:eagle5