**目录贴：[跟我学Shiro目录贴](http://jinnianshilongnian.iteye.com/blog/2018398)**

之前章节我们已经接触过一些INI配置规则了，如果大家使用过如Spring之类的IoC/DI容器的话，Shiro提供的INI配置也是非常类似的，即可以理解为是一个IoC/DI容器，但是区别在于它从一个根对象securityManager开始。

**4.1 根对象SecurityManager**

从之前的Shiro架构图可以看出，Shiro是从根对象SecurityManager进行身份验证和授权的；也就是所有操作都是自它开始的，这个对象是线程安全且真个应用只需要一个即可，因此Shiro提供了SecurityUtils让我们绑定它为全局的，方便后续操作。

因为Shiro的类都是POJO的，因此都很容易放到任何IoC容器管理。但是和一般的IoC容器的区别在于，Shiro从根对象securityManager开始导航；Shiro支持的依赖注入：public空参构造器对象的创建、setter依赖注入。

1、纯Java代码写法（com.github.zhangkaitao.shiro.chapter4.NonConfigurationCreateTest）：

**Java代码  [收藏代码](javascript:void())**

1. DefaultSecurityManager securityManager = **new** DefaultSecurityManager();
2. //设置authenticator
3. ModularRealmAuthenticator authenticator = **new** ModularRealmAuthenticator();
4. authenticator.setAuthenticationStrategy(**new** AtLeastOneSuccessfulStrategy());
5. securityManager.setAuthenticator(authenticator);
7. //设置authorizer
8. ModularRealmAuthorizer authorizer = **new** ModularRealmAuthorizer();
9. authorizer.setPermissionResolver(**new** WildcardPermissionResolver());
10. securityManager.setAuthorizer(authorizer);
12. //设置Realm
13. DruidDataSource ds = **new** DruidDataSource();
14. ds.setDriverClassName("com.mysql.jdbc.Driver");
15. ds.setUrl("jdbc:mysql://localhost:3306/shiro");
16. ds.setUsername("root");
17. ds.setPassword("");
19. JdbcRealm jdbcRealm = **new** JdbcRealm();
20. jdbcRealm.setDataSource(ds);
21. jdbcRealm.setPermissionsLookupEnabled(**true**);
22. securityManager.setRealms(Arrays.asList((Realm) jdbcRealm));
24. //将SecurityManager设置到SecurityUtils 方便全局使用
25. SecurityUtils.setSecurityManager(securityManager);
27. Subject subject = SecurityUtils.getSubject();
28. UsernamePasswordToken token = **new** UsernamePasswordToken("zhang", "123");
29. subject.login(token);
30. Assert.assertTrue(subject.isAuthenticated());

2.1、等价的INI配置（shiro-config.ini）

**Java代码  [收藏代码](javascript:void())**

1. [main]
2. #authenticator
3. authenticator=org.apache.shiro.authc.pam.ModularRealmAuthenticator
4. authenticationStrategy=org.apache.shiro.authc.pam.AtLeastOneSuccessfulStrategy
5. authenticator.authenticationStrategy=$authenticationStrategy
6. securityManager.authenticator=$authenticator
8. #authorizer
9. authorizer=org.apache.shiro.authz.ModularRealmAuthorizer
10. permissionResolver=org.apache.shiro.authz.permission.WildcardPermissionResolver
11. authorizer.permissionResolver=$permissionResolver
12. securityManager.authorizer=$authorizer
14. #realm
15. dataSource=com.alibaba.druid.pool.DruidDataSource
16. dataSource.driverClassName=com.mysql.jdbc.Driver
17. dataSource.url=jdbc:mysql://localhost:3306/shiro
18. dataSource.username=root
19. #dataSource.password=
20. jdbcRealm=org.apache.shiro.realm.jdbc.JdbcRealm
21. jdbcRealm.dataSource=$dataSource
22. jdbcRealm.permissionsLookupEnabled=**true**
23. securityManager.realms=$jdbcRealm

即使没接触过IoC容器的知识，如上配置也是很容易理解的：

1、对象名=全限定类名  相对于调用public无参构造器创建对象

2、对象名.属性名=值    相当于调用setter方法设置常量值

3、对象名.属性名=$对象引用    相当于调用setter方法设置对象引用

2.2、Java代码（com.github.zhangkaitao.shiro.chapter4.ConfigurationCreateTest）

**Java代码  [收藏代码](javascript:void())**

1. Factory<org.apache.shiro.mgt.SecurityManager> factory =
2. **new** IniSecurityManagerFactory("classpath:shiro-config.ini");
4. org.apache.shiro.mgt.SecurityManager securityManager = factory.getInstance();
6. //将SecurityManager设置到SecurityUtils 方便全局使用
7. SecurityUtils.setSecurityManager(securityManager);
8. Subject subject = SecurityUtils.getSubject();
9. UsernamePasswordToken token = **new** UsernamePasswordToken("zhang", "123");
10. subject.login(token);
12. Assert.assertTrue(subject.isAuthenticated());

如上代码是从Shiro INI配置中获取相应的securityManager实例：

1、默认情况先创建一个名字为securityManager，类型为org.apache.shiro.mgt.DefaultSecurityManager的默认的SecurityManager，如果想自定义，只需要在ini配置文件中指定“securityManager=SecurityManager实现类”即可，名字必须为securityManager，它是起始的根；

2、IniSecurityManagerFactory是创建securityManager的工厂，其需要一个ini配置文件路径，其支持“classpath:”（类路径）、“file:”（文件系统）、“url:”（网络）三种路径格式，默认是文件系统；

3、接着获取SecuriyManager实例，后续步骤和之前的一样。

从如上可以看出Shiro INI配置方式本身提供了一个简单的IoC/DI机制方便在配置文件配置，但是是从securityManager这个根对象开始导航。

**4.2 INI配置**

ini配置文件类似于Java中的properties（key=value），不过提供了将key/value分类的特性，key是每个部分不重复即可，而不是整个配置文件。如下是INI配置分类：

**Java代码  [收藏代码](javascript:void())**

1. [main]
2. #提供了对根对象securityManager及其依赖的配置
3. securityManager=org.apache.shiro.mgt.DefaultSecurityManager
4. …………
5. securityManager.realms=$jdbcRealm
7. [users]
8. #提供了对用户/密码及其角色的配置，用户名=密码，角色1，角色2
9. username=password,role1,role2
11. [roles]
12. #提供了角色及权限之间关系的配置，角色=权限1，权限2
13. role1=permission1,permission2
15. [urls]
16. #用于web，提供了对web url拦截相关的配置，url=拦截器[参数]，拦截器
17. /index.html = anon
18. /admin/\*\* = authc, roles[admin], perms["permission1"]

**[main]部分**

提供了对根对象securityManager及其依赖对象的配置。

**创建对象**

**Java代码  [收藏代码](javascript:void())**

1. securityManager=org.apache.shiro.mgt.DefaultSecurityManager

其构造器必须是public空参构造器，通过反射创建相应的实例。

**常量值setter注入**

**Java代码  [收藏代码](javascript:void())**

1. dataSource.driverClassName=com.mysql.jdbc.Driver
2. jdbcRealm.permissionsLookupEnabled=**true**

会自动调用jdbcRealm.setPermissionsLookupEnabled(true)，对于这种常量值会自动类型转换。

**对象引用setter注入**

**Java代码  [收藏代码](javascript:void())**

1. authenticator=org.apache.shiro.authc.pam.ModularRealmAuthenticator
2. authenticationStrategy=org.apache.shiro.authc.pam.AtLeastOneSuccessfulStrategy
3. authenticator.authenticationStrategy=$authenticationStrategy
4. securityManager.authenticator=$authenticator

会自动通过securityManager.setAuthenticator(authenticator)注入引用依赖。

**嵌套属性setter注入**

**Java代码  [收藏代码](javascript:void())**

1. securityManager.authenticator.authenticationStrategy=$authenticationStrategy

也支持这种嵌套方式的setter注入。

**byte数组setter注入**

**Java代码  [收藏代码](javascript:void())**

1. #base64 **byte**[]
2. authenticator.bytes=aGVsbG8=
3. #hex **byte**[]
4. authenticator.bytes=0x68656c6c6f

默认需要使用Base64进行编码，也可以使用0x十六进制。

**Array/Set/List setter注入**

**Java代码  [收藏代码](javascript:void())**

1. authenticator.array=1,2,3
2. authenticator.set=$jdbcRealm,$jdbcRealm

多个之间通过“，”分割。

**Map setter注入**

**Java代码  [收藏代码](javascript:void())**

1. authenticator.map=$jdbcRealm:$jdbcRealm,1:1,key:abc

即格式是：map=key：value，key：value，可以注入常量及引用值，常量的话都看作字符串（即使有泛型也不会自动造型）。

**实例化/注入顺序**

**Java代码  [收藏代码](javascript:void())**

1. realm=Realm1
2. realm=Realm12
4. authenticator.bytes=aGVsbG8=
5. authenticator.bytes=0x68656c6c6f

后边的覆盖前边的注入。

测试用例请参考配置文件shiro-config-main.ini。

**[users]部分**

配置用户名/密码及其角色，格式：“用户名=密码，角色1，角色2”，角色部分可省略。如：

**Java代码  [收藏代码](javascript:void())**

1. [users]
2. zhang=123,role1,role2
3. wang=123

密码一般生成其摘要/加密存储，后续章节介绍。

**[roles]部分**

配置角色及权限之间的关系，格式：“角色=权限1，权限2”；如：

**Java代码  [收藏代码](javascript:void())**

1. [roles]
2. role1=user:create,user:update
3. role2=\*

如果只有角色没有对应的权限，可以不配roles，具体规则请参考授权章节。

**[urls]部分**

配置url及相应的拦截器之间的关系，格式：“url=拦截器[参数]，拦截器[参数]，如：

**Java代码  [收藏代码](javascript:void())**

1. [urls]
2. /admin/\*\* = authc, roles[admin], perms["permission1"]

具体规则参见web相关章节。