1. Shiro快速入门

本章从快速入手的角度出发，搭建Shiro的应用环境，并且实现一个简单的基于Shiro的应用，以便我们在详细研究Shiro的各个组件和各种应用前，对Shiro有个基本的感性认识，并且初步体验出基于Shiro编写应用的配置和基本步骤。关于本章，我们需要把握如下几个方面：

* Shiro的安装
* Shiro基于ini的配置
* 体验Subject、SecurityManager、Session组件的意义和使用
* 从全局角度理解Shiro的编码和运行流程

目录

[2 Shiro快速入门 1](#_Toc493755123)

[2.1 安装Shiro 2](#_Toc493755124)

[2.2 运用Shiro 4](#_Toc493755125)

[2.2.1 配置Shiro 4](#_Toc493755126)

[2.2.2 使用Shiro 8](#_Toc493755127)

[2.3 总结 11](#_Toc493755128)

* 1. 安装Shiro

Shiro是一个安全框架，不是一个纯软件，所有对Shiro的安装不像JDK和其他应用软件一样，下载安装程序并执行，而是下载Shiro相关的jar包，集成到我们的项目中。所以安装Shiro的基本思路是在shiro.apache.org网址上下载相关的jar包，添加到我们工程的builder path中。

因为本书的所有案例都是基于Maven进行管理的，为了方便以后章节案例的重复使用，我们是创建一个Maven的父工程，专门完成Shiro安全框架的安装。接下来我们介绍ShiroPOM父工程的创建，并且在该工程的pom.xml中加入Shiro的相关依赖，让Maven帮我们完成Shiro相关jar包的下载。

1. 在Eclipse中创建一个简单的Maven Java工程，Maven的坐标信息是：ShiroBook（groupId）、ShiroPOM(artifactId)、0.0.1-SNAPSHOT(version)。工程的目录结构大体如图2-1 ShiroPOM目录结构。

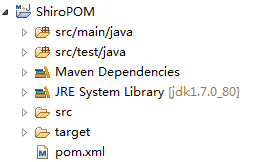


图2-1 ShiroPOM目录结构

1. 修改pom.xml中的packaging类型为pom，在properties元素中添加shiro.version子元素，指定我们要使用的shiro版本。本书案例用的shiro的版本是1.3.2。样例代码如下：

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>ShiroBook</groupId>

<artifactId>ShiroPOM</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>pom</packaging>

<name>ShiroPOM</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<shiro.version>1.3.2</shiro.version>

</properties>

</project>

1. 在pom.xml中添加shiro的核心依赖，同时添加slf4j-simple和junit依赖。Shiro-core依赖是shiro安全框架的基本依赖，slf4j-simple和junit是为了方便后面代码执行测试应用和测试案例要用到的依赖。样例代码如下：

<project>

... ...

<dependencies>

<dependency>

<groupId>org.apache.shiro</groupId>

<artifactId>shiro-core</artifactId>

<version>${shiro.version}</version>

</dependency>

<!-- 使用slf4j输出日志信息 -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>1.6.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.7</version>

<scope>test</scope>

</dependency>

</dependencies>

... ...

</project>

1. 在pom.xml中的build->pluginManagement->plugins中，添加maven-compile-plugin和exec-maven-plugin两个插件。Maven-compile-plugin插件用来编译Maven工程代码，exec-maven-plugin插件用来启动测试应用。样例代码如下：

<project>

... ...

<build>

<pluginManagement>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.0.2</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>${project.build.sourceEncoding}</encoding>

</configuration>

</plugin>

<!-- 该插件用来运行测试，在正式项目中不是必须的 -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>1.2.1</version>

</plugin>

</plugins>

</pluginManagement>

</build>

</project>

1. 鼠标右击工程，选择Run As->Maven Install，执行Maven的install命令，将ShiroPOM安装到本地资源库中，以便后面的Maven工程使用。

到这里，我们基本完成了Shiro安全框架基本组件的安装，关键是配置好ShiroPOM工程里面的pom.xml内容，最后一定要执行Maven的install命令，否则后面的Maven工程将找不到ShiroPOM依赖。pom.xm的完整代码，可以查看随书code目录中的[code/ShiroPOM/pom.xml](../code/ShiroPOM/pom.xml)。

* 1. 运用Shiro

前面2.1小结介绍了Shiro基本组件的安装，接下来在ShiroPOM工程的基础上，创建一个新的Maven工程，编写一个简单的Java应用程序，从代码的角度体验一下Shiro的配置和使用，从而理解Shiro的作用、使用和Shiro几个核心组件的关系。

* + 1. 配置Shiro

1. 创建一个Maven工程，坐标信息是：ShiroBook(groupId)、ShiroTP02(artifactId)、0.0.1-SNAPSHOT(version)，指定parent为ShiroPOM，pom.xml中样例代码如下，注意粗体标识内容。

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

**<parent>**

**<groupId>ShiroBook</groupId>**

**<artifactId>ShiroPOM</artifactId>**

**<version>0.0.1-SNAPSHOT</version>**

**</parent>**

**<artifactId>ShiroTP02</artifactId>**

<packaging>**jar**</packaging>

<name>ShiroTP02</name>

... ...

</project>

1. 在pom.xml中的build->plugins元素下，添加maven-compile-plugin和exec-maven-plugin插件，ShiroTP02的代码，将会使用maven-compile-plugin插件编译，FirstApplication将会使用exec-maven-plugin插件启动。pom.xml的样例代码如下。

<project>

... ...

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>${project.build.sourceEncoding}</encoding>

</configuration>

</plugin>

<!-- 该插件用来运行测试，在正式项目中不是必须的 -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<executions>

<execution>

<goals>

<goal>java</goal>

</goals>

</execution>

</executions>

<configuration>

<classpathScope>test</classpathScope>

**<mainClass>cn.com.shiro.book.tp02.FirstApplication</mainClass>**

</configuration>

</plugin>

</plugins>

</build>

</project>

其中粗体标识的mainClass元素内容，是我们将用exec-maven-plugin插件启动的测试应用的主程序。

1. 在ShiroTP02/src/main/java目录下，创建cn.com.shiro.book.tp01.FirstApplication类，样例代码如下：

package cn.com.shiro.book.tp01;

… …

public class FirstApplication {

private static final transient Logger log = LoggerFactory.getLogger(FirstApplication.class);

public static void main(String[] args) {

log.info("第一个Apache Shiro Application");

System.exit(0);

}

}

在main方法中添加了一条输入信息的语句log.info(“第一个Apache Shiro Application”)。

1. 基于Eclipse执行Maven的exec:java命令，测试exec-maven-plugin是否正确运行。正确结果是Eclipse控制台会输入如下图2-2测试exec结果。

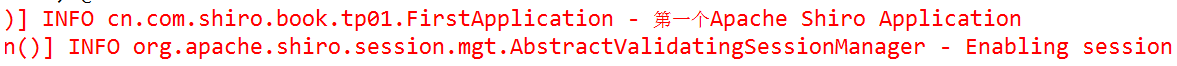


图2-2 测试exec结果

在Eclipse中默认是没有exec:java命令的，我们需要鼠标右击ShiroTP02工程，选择点击Run As->Maven build… 选项，出现如下图2-3Edit Configuration

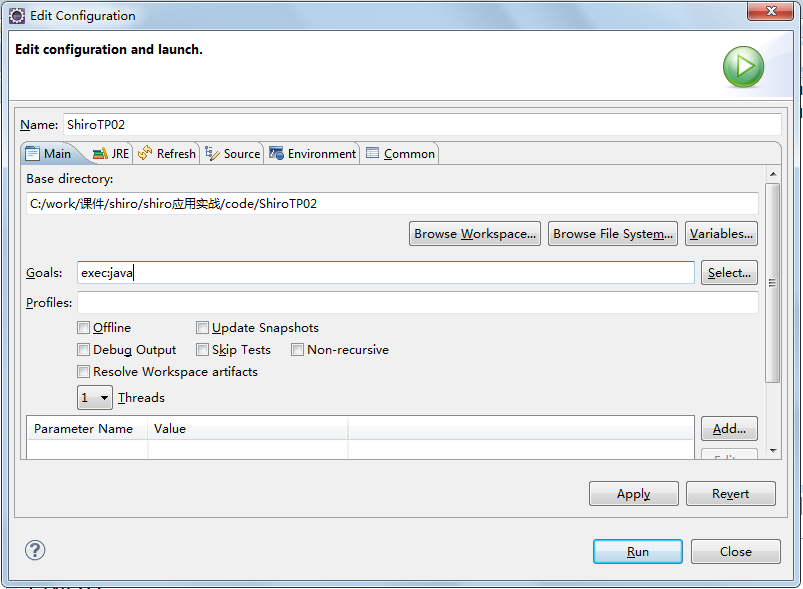


图2-3 Edit Configuration

在Goals后面的输入框中输入 exec:java，点击Run按钮运行。

1. 在ShiroTP02/src/main/resources创建shiro.ini文件，在里面添加用户账号信息，用户拥有的角色信息，和角色以及角色拥有的权限信息。样例代码如下，shiro.ini的详细信息可以查看[ShiroTP02/src/main/resources/shiro.ini](../code/ShiroTP02/src/main/resources/shiro.ini)文件。

# =============================================================================

# FirostApplication INI configuration

# =============================================================================

# -----------------------------------------------------------------------------

# 配置用户和每个用户的密码和角色信息

# username = password, role1, role2, ..., roleN

# -----------------------------------------------------------------------------

[users]

root = 123456, admin

guest = 123456, guest

lisi = 123456, orderManager, shoperManager

wangwu = 123456, shoperManager

# -----------------------------------------------------------------------------

# 配置角色和对应的权限

# roleName = perm1, perm2, ..., permN

# -----------------------------------------------------------------------------

[roles]

admin = \*

shoperManager = shoper:\*

orderManager = order:\*

guest=order:view

在shiro.ini文件中我们通过[roles]后面的内容，定义admin、shoperManager、orderManager和guest四个角色。

* admin=\* 表示角色admin拥有所有权限
* shoperManager=shoper:\* 表示角色shoperManager拥有操作shoper资源的所有权限
* orderManager=order:\* 表示角色orderManager拥有操作order资源的所有权限
* guest=order:view 表示角色guest拥有view(查看) order资源的权限

在shiro.ini文件中，我们通过[users]后面的内容定义了root、guest、lisi、王五等四个用户。

* root=123456, admin 表示root用户的密码是123456，拥有admin角色
* guest=123456, guest 表示guest用户的密码是123456，拥有guest角色
* lisi=123456,orderManager, shoperManager 表示用户lisi的密码是123456，拥有orderManager和shoperManager两个角色
* wangwu=123456, shoperManager表示 wangwu用户名的密码是123456，拥有shoperManager角色

1. 在FirstApplication.java文件中，添加创建Shiro SecurityManager组件对象的代码，样例代码块如下：

public static void main(String[] args) {

log.info("第一个Apache Shiro Application");

//1.创建IniSecurityManagerFactory，基于shiro.ini进行初始化

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

//2.获取SecurityManager对象

org.apache.shiro.mgt.SecurityManager securityManager = factory.getInstance();

//3.注册全局的SecurityManager对象

SecurityUtils.setSecurityManager(securityManager);

System.exit(0);

}

查看上面的代码，我们只要通过简单的3行代码，就可以创建Shiro安全框架的核心对象(SecurityManager对象)

* 通过1行代码，我们通过IniSecurityManagerFactory读取在classpath中的shiro.ini文件，完成对Shiro环境的初始化。在IniSecurityManagerFactory的构造器中，我们在shiro.ini文件名前添加了classpath前缀，表示从classpath中读取shiro.ini文件。处理了可以使用classpath前缀外，我们也可以使用url和file前缀，分别表示加载网络资源和文件系统中的文件。
* 通过2行代码，我们调用factory.getInstance()方法解析shiro.ini中的配置信息，并且初始化出SecurityManager对象方法
* 通过3行，我们通过SecurityUtils.setSecurityManager(securityManager)将securityManager注册成一个静态的单例对象，作用范围是整个JVM。在实际项目中，我们可以把securityManager对象保存在其他容器中，比如ServletContext、Spring IOC等。
  + 1. 使用Shiro

通过前面的配置和代码，我们获取到了SecurityManager对象，接下来就可以基于Shiro做相关的安全操作了。

在正式项目中的安全操作需求，一般是回答两个问题：当前用户是谁？当前用户有没有权限做某个事情？围绕这两个问题，我们去编写对应的实现代码或设计相关的用户接口。最终一条，所有的工作，都是针对每个用户展开的。Shiro安全框架设计了Subject类，一个Subject对象就表示一个用户。在前面注册好全局SecurityManager对象的前提下，可以通过如下代码，获取到一个代表当前用户的Subject对象。

Subject currUser =SecurityUtils.getSubject();

使用SecurityUtils.getSubject()，我们可以获取到当前正在执行的Subject(主体)对象，为什么叫Subject(主体)对象，而不用我们以前的叫法，叫User(用户)对象呢？因为我们说用户，一般是指一个人，某个正在操作的人，而当前使用Shiro安全框架的，可能是某个人，也可以是第三方的应用或服务，甚至还可以是一个独立的任务。所以我们用Subject(主体)这个称呼表示当前正在执行安全操作的对象，虽然比较拗口，但是可以避免用户带来的歧义。

通过SecurityUtils.getSubject()获取到一个Subject对象后，意味着当前Subject对象同SecurityManager(Shiro)就建立了一个单独唯一的会话，这个会话会贯穿当前Subject同Shiro安全框架的所有操作。如果当前Subject对象有某个数据，需要从当前阶段传递到后面的某个阶段使用，我们就可以获取当前Subject对象的会话对象--Session对象实现。如：

Session session = currUser.getSession();

session.setAttribute(“age”,18);

Shiro中Session对象的操作，同Web环境下的HttpSession是一样的，可以调用Subject.getSession()方法获取，也可以调用Subject.getSession(boolean)方法获取。Subject.getSession()的效果同Subject.getSession(true)效果一样；Subject.getSession(true)一定会返回一个Session对象，如果以前不存在，会创建一个返回;Subject.getSession(false)返回一个已经存在的Session对象，如果不存在，就返回一个null。一般情况下，我们推荐使用Subject.getSession()方式获取Session。

获取Session对象后，我们就可以使用Session的setAttribute(String,Object)方法往Session对象中设置一个数据，在需要的时候，使用Object getAttrinbute(String)方法获取出来使用。

需要注意一下的是，HttpSession只能在Http Web环境下才有，而Shiro的Session机制，在一个纯J2SE环境下也是有实现的，由Shiro安全框架自己创建管理。如果当前Shiro运行在Web环境下，Shiro会默认使用Web环境下的HttpSession机制，我们也可以特别指定不使用Web的HttpSession，而使用Shiro安全框架自己的Session。

前面我们使用SecurityUtils.getSubject()方法获取到了当前的Subject对象，而且能获取到Subject对象对应的Session对象了，那当前的Subject对象是谁呢？张三、李四、王五还是第三方的订单管理服务？

都不是，这时候的Subject是一个匿名主体，因为它还没登录，没有完成用户凭证验证的。我们可以用如下代码，完成当前的Subject的用户凭证认证，从而确定Subject对象最终代表谁。

if(!currUser.isAuthenticated()){

UsernamePasswordToken token = new UsernamePasswordToken("zhangsan","12345");

token.setRememberMe(true);

currUser.login(token);

}

If语句中，通过Subject的isAuthenticated()方法判断当前的Subject对象是不是登录过，如果没有登录，就创建一个用户名和密码的UsernamePasswordToken对象，调用Subject对象currUser的登录方法完成验证。

因为我们创建的UsernamePasswordToken对象有可能验证通不过，有可能用户名不存在、有可能密码错误等。这些认证失败的结果，Shiro是通过Subject.login()方法抛出的异常体现的，如下面代码块：

if (!currUser.isAuthenticated()) {

UsernamePasswordToken token = new UsernamePasswordToken("zhangsan",

"12345");

token.setRememberMe(true);

try {

currUser.login(token);

} catch (UnknownAccountException unk) {

// 没有账号异常

unk.printStackTrace();

} catch (IncorrectCredentialsException inc) {

// 账号密码不匹配异常

inc.printStackTrace();

} catch (LockedAccountException lock) {

// 账号被锁定异常

lock.printStackTrace();

} catch (AuthenticationException auth) {

// 其他凭证验证异常

auth.printStackTrace();

}

}

如果登录成功，我们就知道当前的Subject对象currUser是谁了，我可以用如下代码输出当前Subject的账号信息。

// 打印currUser的账号

log.info("当前Subject的账号是:" + currUser.getPrincipal());

我们可以通过调用Subject.hasRole(String)方法判断当前的Subject是否拥有指定的角色，样例代码块如下：

// 判断currUser是否拥有指定的角色

if(currUser.hasRole("shoperManager")){

log.info(currUser.getPrincipal() + " 拥有shoperManager角色" );

}else{

log.info(currUser.getPrincipal() + " 没有shoperManager角色" );

}

我们可以通过调用Subject.isPermitted(String)方法，判断当前的Subject是否拥有指定的操作权限，样例代码块如下：

//判断currUser是否有指定的权限

if(currUser.isPermitted("userManager:create")){

log.info(currUser.getPrincipal() + " 拥有userManager:create权限");

}else{

log.info(currUser.getPrincipal() + " 没有userManager:create权限");

}

最后，我们也可以通过调用Subject.logout()方法，是当前Subject对象退出，样例代码块如下：

// 退出登录subject

currUser.logout();

将前面的所有代码，都集中到FirstApplication.java的main方法中，就可以直接运行 exec:java，查看输出结果是否同我们期望的一样(在shiro.ini中配置的用户和角色信息是否对应)。我们可以改变UsernamePasswordToken的账号名称和密码，做不同情况的测试。FirstApplication.java详细内容，可以参考随书代码中的[code/ShiroTP02/src/main/java/cn/com/shiro/book/tp02/  
FirstApplication.java](../code/ShiroTP02/src/main/java/cn/com/shiro/book/tp02/FirstApplication.java)

* 1. 总结

本章主要从简单快速入门的角度，介绍了Shiro安全框架的安装、配置和使用。通过本章的介绍，我们需要把握如下几个方面：

* 通过在Maven工程的pom.xml中添加shiro-core依赖完成基本Shiro安全框架的安装；
* 使用ini文件，配置Shiro SecurityManager初始化时的账号信息和权限信息
* 通过IniSecurityManagerFactory可以初始化，创建一个SecurityManager对象，使用SecurityUtils的setSecurityManager方法，将SecurityManager设置全局对象；
* 通过调用SecurityUtils的getSubject方法，可以获取当前的Subject对象，接下来就可以操作Subject对象达到操作权限功能的目的。