第一部分 Shiro基础

1. Apache Shiro简介

本章主要从全局的角度，介绍Apache Shiro全貌。包括Apache Shiro的作用，优势，核心组件等等，还包括学习使用Apache Shiro前的准备事项，如下载安装JDK环境、Eclipse、Maven、Shiro等等。

作为读者，主要把握如下几个方面：

* Apache Shiro的意义和优势
* Apache Shiro的体系结构
* Apache Shiro的核心组件
* Apache Shiro的环境准备

目录

[1 Apache Shiro简介 2](#_Toc493755061)

[1.1 软件安全控制 3](#_Toc493755062)

[1.2 Apache Shiro的意义和优势 3](#_Toc493755063)

[1.3 Apache Shiro功能特征 3](#_Toc493755064)

[1.4 Apache Shiro体系结构 4](#_Toc493755065)

[1.4.1 Shiro概貌 5](#_Toc493755066)

[1.4.2 Shiro内部组成结构 6](#_Toc493755067)

[1.5 搭建Apache Shiro的外部环境 8](#_Toc493755068)

[1.6 总结 8](#_Toc493755069)

* 1. 软件安全控制

只要我们实现过、甚至使用过软件，就可以体会出安全控制是每个软件不可或缺的功能。最基本的控制，也必须得有用户名和密码验证登录后，才能进入软件内部进行操作。ATM机上操作，需要输入密码验证；打开手机玩微信，需要输入密码验证；QQ聊天，需要输入密码登录验证；甚至打开日记本写日记，也有一个小锁扣。因为这些操作后面，都包含个人需要保密，不希望别人看到的信息。

而实际软件里面，对安全控制往往不仅仅只是用户名和密码验证就够了的，使用用户名和密码，只是验证是不是可以进入系统，而在系统里面有很多操作，是因人而异的。比如一个电子商务平台，有些人进去系统，可以选购商品、下单、支付等，有些人就可以维护商品的上下架，还有些人可以统计整个平台的销售业绩等等。这里每个用户名，只能做自己权限范围内的事情，权限范围外的操作，我们需要系统自动校验并且屏蔽。

也就是说，一个完整的软件产品，我们不仅仅要实现用户的具体业务需求，还必须实现系统的权限控制，保证合适的人，能而且只能操作自己范围内的功能，保证系统使用的安全性。因为这样的功能是独立于其他业务功能的，所以我们会把实现安全控制功能的部分，设计成一个独立的安全控制模块。

安全控制模块按功能可以分两部分：

一是维护管理权限信息，最终实现哪些人可以做哪些事的维护管理，比如张三只能浏览公告信息；李四只能查看自己的订单；王五可以查看整个平台的订单等等；

一是系统运行的时候，根据管理好哪些人可以做哪些事的权限信息，对每个人的具体操作进行验证，如果操作确实是在他的权限范围内，就可以顺利操作，否则就阻止操作，提示没有权限操作等信息。

* 1. Apache Shiro的意义和优势

Apache Shiro是Apache软件基金会(Apache Software Foundation)提供的一个软件开源项目，它为我们实现了一个功能强大并且使用方便的Java安全框架，该框架实现了权限的认证、验证、加密和会话管理、缓存管理等公共功能，能运用到任何Java项目中，从简单的命令行应用、到Web应用、手机应用，甚至分布式企业级应用。

当然，除了Apache Shiro安全框架实现安全控制功能外，也有其他的安全框架，比如JavaEE中有Java Security解决方案；Spring内部也提供了Spring Security模块。

Apache Shiro安全框架，相对其他Java安全框架有如下优势：

* 有一套易于理解的Java Security API
* 简单的身份认证(登录)，支持多种数据源(LDAP、JDBC、ActiveDirectory、ini配置等)
* 支持角色基本的和自定义细粒度的访问控制
* 支持缓存，提高系统性能
* 自定义的会话管理，同时支持Web和非web应用
* 支持异构的客户端
* 内置加密服务API
* 不同任何框架捆绑，独立运行
  1. Apache Shiro功能特征

Apache Shiro作为一个基于Java的强大易用的安全框架很多安全功能。图1-1Shiro功能 描述了Shiro经典功能，这一节主要简单介绍这些功能。

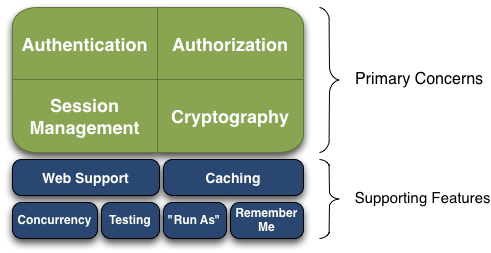


图1-1 Shiro功能

Shiro安全框架的最主要目标是为基于Shiro安全框架开发的程序员，提供四个最基本核心的系统安全功能：身份认证/登录(Authentication)、授权(Authorization)、会话管理(Session Manager)、加密(Cryptography)。

* Authentication：身份认证/登录，验证用户的身份是否正确
* Authorization：授权，即验证权限，验证某个已经证明身份的人，是否拥有操作某个功能的权限。如判断用户是不是拥有某个角色，或判断用户是否能修改某个数据等。
* Session Manager：会话管理。一个用户从登陆到退出，所有信息都保存在会话中。Shiro不仅仅支持Web环境下的会话管理，还支持J2SE环境下的会话管理。
* Cryptography：加密。使用加密技术保证数据的安全性，比如用户密码都是加密后保存到数据库中，避免明码的不安全性。

Shiro安全框架除了提供了上面的4个核心功能外，还提供了一些其他扩展辅助功能，我们在需要的时候，可以自由的将它们集成到系统中使用：

* Web Support：对Web应用的支持，能很方便的将Shiro集成到Web应用中
* Caching：缓存支持。比如用户登录后，该用户的权限信息可以保存在缓存中，避免每次授权的时候，都要访问数据库，提高系统性能。
* Concurrency：支持多线程并发认证。可以在一个线程中开启另外一个线程，自动传播权限。
* Testing：内置测试模块，方便测试。
* Run As：支持一个用户冒充另外一个用户的身份进行访问
* Remember Me：记住用户登录状态，在一定的时间内，不需要重复登录验证
  1. Apache Shiro体系结构

Apache Shiro的设计目标实现一个简单、直观、易用的安全框架，实现应用系统的安全功能，它的核心设计模式的目标同我们在项目中实现安全管理目标一样，在一个会话（上下）文中，实现用户（或第三方系统）同系统的安全交互。

在实际项目中，我们的安全模块的最终需求就是实现一个用户如果正确登录了，我们就在界面上显示按钮或链接，让用户可以操作某个功能，否则的话，就提示一个登录页面，要求用户登录后才能操作。为了满足这样的需求，我们需要自己根据每个应用定义大量的API接口和实现类，实现这样的功能。甚至有可能不是人，而是第三方的系统访问我们的系统，我们还需要编写针对第三方软件访问的接口。

Shiro将这些API抽象出公共的接口，并且做出来简单易用的封装和实现，程序员只需要将Shiro集成到各种应用中，用同一套API就可以实现不同系统的各种安全功能的实现。接下来我们开始逐步的研究Shiro内部的体系结构。

* + 1. Shiro概貌

从外部看Shiro，Shiro由三个主要组件组成：Subject、SecurityManager和Realms。图1-2Shiro概貌描述了这三个组件的关系，接下来我们先研究一下这三个组件。

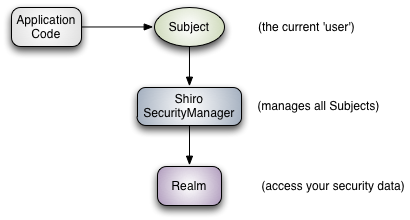


图1-2 Shiro概貌

通过图1-2 Shiro概貌，我们可以看到应用程序的代码，同Shiro安全框架直接交互的对象是Subject，也就是说Shiro对外的API就是Subject。

* Subject：主体，代表“用户”。这个用户不仅仅是代表人，也代表第三方的系统，如网络爬虫、财务系统等，总之，代表的所有访问Shiro的第三方。Subject对象是同SecurityManager对象绑定在一起的，当用户同Shiro进行安全交互的时候，都会体现成一个对应的Subject对象同SecurityManager对象进行安全交互。
* SecurityManager：安全管理器。每个Subject的安全相关的操作，都是传给SecurityManager具体实现的，而且它还管理者系统中所有的Subject。通过图1-2我们可以看出，SecurityManager是Shiro的核心。关于SecurityManager的详细情况，我们在后面章节逐个介绍，在这里我们要重点把握的是用户基于Subject，同Shiro的所有安全操作，都是交给SecurityManager对象幕后完成的。
* Realms：域。可以把Realm理解成“桥梁”或“连接器”，通过Realm这个桥梁，Shiro可以同我们的应用交互安全数据。比如在登录的时候，需要知道用户账户信息和对应的密码信息，这些信息都是存在我们自己的系统中的，Shiro就需要通过Realm获取这些信息，完成对账户和密码的匹配验证。Realm可以理解成是一个特殊的DAO，当Shiro需要验证和授权的基本信息完成安全操作的时候，就是通过Realm DAO，同我们的系统进行数据交互。Shiro在运行的时候，至少要给SecurityManager对象配置一个Realm对象，当然，我们也可以给SecurityManager对象配置多个Realm，但是至少得配置一个。Shiro提供了很多种Realm，以便SecurityManager可以访问各种外部数据，比如LDAP、JDBC、ini配置文件和其他属性文件等等，当然，我们也可以根据自己的情况，自己实现Realm，为SecurityManager提供数据。

通过前面的介绍，我们可以得出如下结论：

1. 我们的应用代码，是通过Subject对象同Shiro完成安全操作交互的
2. 所有的安全操作，都是Subject交给绑定的SecurityManager对象完成的
3. SecurityManager完成安全操作，所需要的数据，由Realm对象提供
4. Shiro不维护用户和权限等相关的安全信息，这些信息需要开发人员自己独立维护，并且通过Realm提供给SecurityManager对象使用。
   * 1. Shiro内部组成结构

前面我们了解了Shiro的概貌组件，接下来我们详细研究一下Shiro内部的体系结构，图1-3 Shiro内部结构 描述Shiro运行时的内部组件和关系。



图1-3 Shiro内部结构

* Subject(org.apache.shiro.subject.Subject)

代表需要基于Shiro实现安全操作的用户（用户、第三方系统或独立任务等）的主题

* SecurityManager(org.apache.shiro.mgt.SecurityManager)

安全管理器。上面有提到的，SecurityManager是Shiro的核心，它像一个主事的总管，协调被它管理的所有对象顺利的完成所有的安全操作功能。它同时还管理所有同Shiro进行交互的用户，它知道所有用户的安全权限。

* Authenticator(org.apache.shiro.authc.Authenticator)

认证器。在用户登录的时候，由Authenticator完成对用户身份的认证，验证用户是不是真正的用户，也就是认证用户的凭证是否正确。当Authenticator认证用户的凭证真实性的时候，会自动调用Realm对象，从我们的系统中获取对应用户的账号信息（一般是用户名和密码）。

Authentication Strategy(org.apache.shiro.authc.pam.AuthenticationStrategy)

如果在SecurityManager配置了多个Realm，就由AuthenticationStrategy协调按什么方式进行用户凭证的认证，比如是只要一个Realm认证通过了就算认证通过呢，还是需要所有的Realm认证通过了才算认证通过。

* Authorizer(org.apache.shiro.authz.Authorizer)

授权器。Authorizer负责判断用户是否有权限进行安全操作。也就是说，当前用户能不能做什么事情，由Authorizer说了算。同Authenticator一样，Authorizer知道怎么样同后台数据进行交互，从而根据用户拥有的角色和权限进行判断是否允许用户做什么事情。

* SessionManager(org.apache.shiro.session.mgt.SessionManager)

Session管理器。SessionManger会根据每个用户创建一个Session来维护当前用户的会话周期，并且实现用户会话的过期控制。对用户会话管理是Shiro特有的功能，其他安全框架都没有单独提供的。Shiro自动的对所有用户都进行会话管理，哪怕是当前应用不是运行在Web容器中或EJB容器中。Shiro会自动的使用已有的容器环境进行Session管理，如果整个应用不是运行在容器中的话，Shiro也会独立实现Session管理并且控制会话的过期。同样的，也可以基于SessionDAO实现Session对象的持久化。

SessionDAO(org.apache.shiro.session.mgt.eis.SessionDAO)

SessionDAO为SecurityManager提供对Session对象的持久化操作(CRUD)。根据用户提供的不同插件，可以实现各种方式的持久化，比如数据库、内存、缓存等等都支持。

* CacheManager(org.apache.cache.CacheManager)

缓存管理器。CacheManager是Shiro中创建并且管理缓存对象的组件。因为Shiro在进行认证、授权和Session管理的时候，有很多数据需要交互，如果每次都直接访问后面的数据源的话，性能会比较低。有缓存后，就可以先访问缓存中的数据，没有找到合适的数据，再访问后面的数据源，这样就可以提高系统性能。Shiro提供了对缓存操作的接口，具体使用哪个缓存组件，用户可以根据自己的需要选择，比如Ehcache等。

* Cryptography(org.apache.shiro.crypto.\*)

加密模块。加密模块是企业级安全框架的一个功能。Shiro的crypto包中定义很多容易理解，且简单方便使用的基于Ciphers、Hashes等不同算法的加密实现。我们可以基于这些实现类，实现数据的加密解密，从而避免明码的不安全性。

* Realms(org.apache.shiro.realm.Realm)

域。前面提到了，Realm就等用于一个桥梁或连接器，实现Shiro同后台安全数据之间的交互。当Shiro要进行认证、授权的时候，自动通过配置的Realm同后台数据进行交互，获取必要的认证或授权的依据数据。我们需要在SecurityManager中配置至少一个Realm，也可以配置自定义的Realm实现Shiro同后台数据的交互。

* 1. 搭建Apache Shiro的外部环境

要研究并且使用Apache Shiro，我们需要先搭建好Shiro的开发和运行环境。

因为Shiro是一个Java安全框架，所以研究Shiro的基本环境是Java环境。我们可以从Oracle官方网站(http://www.oracle.com/technetwork/java/index.html)上下载JDK进行安装，建议下载jdk1.7以上的版本，而且注意jdk的位数（32位或64位），也可以安装随书软件目录下的[jdk-7u80-windows-x64.exe](../softwares/jdk-7u80-windows-x64.exe)。本书所有的案例都是是基于64位JDK1.7的。对JDK的安装和配置，参考Oracle官方资料，或查找其他的网络资源。最终的效果是，打开cmd命令窗口，输入java –version命令 和 javac –version命令，可以查看到java运行环境版本信息和java编译器版本信息。如图1-4 java环境测试 所示。

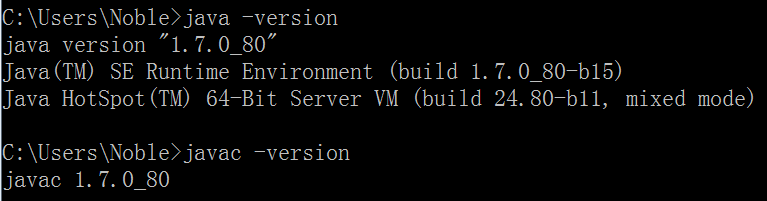


图1-4 java环境测试

安装好java环境，接下来我们需要安装好Eclipse IDE工具，我们可以从Eclipse的官方网站(https://www.eclipse.org/downloads/)下载，也可以安装随书software目录下的[eclipse-java-mars-R-win32-x86\_64.zip](../softwares/eclipse-java-mars-R-win32-x86_64.zip)。

因为本书所有案例，都是基于Maven进行项目管理的，要能顺利运行本书案例的话，请安装好Maven，可以从maven官网下载(http://maven.apache.org/)，也可以使用随书软件目录下的[apache-maven-3.3.9-bin.zip](../softwares/apache-maven-3.3.9-bin.zip)。具体关于Maven的安装和使用，请参考网络上的相关资料。

另外，因为本书后面部分，会研究Shiro同Web应用的集成，所以建议安装好Web服务器。本书使用的是Tomcat7服务器，我们可以从tomcat的官网下载tomcat(http://tomcat.apache.org/)，也可以使用随书软件目录下的[apache-tomcat-7.0.69-windows-x64.zip](../softwares/apache-tomcat-7.0.69-windows-x64.zip)。

* 1. 总结

本章节主要从全局的角度介绍了Apache Shiro的意义、作用和优势，还有Shiro的架构体系和核心组件，以及每个组件的作用和各个组件之间的关系，最后为了方便以后章节对Shiro的研究，还介绍了需要安装的软件。

对Shiro体系结构和核心组件的理解，是这章节的重点，因为它涉及到Shiro的运行过程和后续Shiro安全框架的运用。

从全局讲，Shiro有三个关键组件，Subject、SecurityManager和Realm。Subject代码用户对象；SecurityManager是Shiro框架的核心，它协调管理Shiro中所有的组件对象，具体完成用户通过Subject传递过来的所有安全操作；Realm对象实现Shiro同后台安全数据的交互。

从内部详细结构分析，Shiro包括Subject、SecurityManager、Authenticator、Authorizer、Realm、SessionManager、SessionDAO、CacheManager、Cryptography等九个组件。