

让Apache Shiro保护你的应用

作者 **Les Hazlewood** 译者 **胡伟红** 发布于 2011年5月11日
领域 **语言 & 开发, 架构 & 设计** 主题 **安全, Java, 架构**
分享     

标记书签!






在尝试保护你的应用时，你是否有过挫败感？是否觉得现有的Java安全解决方案难以使用，只会让你更糊涂？本文介绍的**Apache Shiro**，是一个不同寻常的Java安全框架，为保护应用提供了简单而强大的方法。本文还解释了**Apache Shiro**的项目目标、架构理念以及如何使用**Shiro**为应用安全保驾护航。

什么是Apache Shiro?

Apache Shiro（发音为“shee-roh”，日语“堡垒（Castle）”的意思）是一个强大易用的Java安全框架，提供了认证、授权、加密和会话管理功能，可为任何应用提供安全保障 - 从命令行应用、移动应用到大型网络及企业应用。

相关厂商内容

-  **Flash Builder 4.5** 高级版试用版免费高速下载
-  **QCon杭州2011** 大会晚场技术沙龙 (10.21、22)
-  阿里云开发者大会，与愤怒的小鸟\水果忍者\植物大战僵尸开发团队面对面，立即报名!

Shiro为解决下列问题（我喜欢称它们为应用安全的四要素）提供了保护应用的API：

- 认证 - 用户身份识别，常被称为用户“登录”；
- 授权 - 访问控制；
- 密码加密 - 保护或隐藏数据防止被偷窥；
- 会话管理 - 每用户相关的时间敏感的状态。

Shiro还支持一些辅助特性，如Web应用安全、单元测试和多线程，它们的存在强化了上面提到的四个要素。

为何要创建Apache Shiro?

对于一个框架来讲，使其有存在价值的最好例证就是有让你去用它的原因，它应该能完成一些别人无法做到的事情。要理解这一点，需要了解**Shiro**的历史以及创建它时的其他替代方法。

在2008年加入**Apache**软件基金会之前，**Shiro**已经5岁了，之前它被称为**JSecurity**项目，始于2003年初。当时，对于Java应用开发人员而言，没有太多的通用安全替代方案 - 我们被Java认证/授权服务（或称为**JAAS**）紧紧套牢了。**JAAS**有太多的缺点 - 尽管它的认证功能尚可忍受，但授权方面却显得拙劣，用起来令人沮丧。此外，**JAAS**跟虚拟机层面的安全问题关系非常紧密，如判断JVM中是否允许装入一个类。作为应用开发者，我更关心应用最终用户能做什么，而不是我的代码在JVM中能做什么。

由于当时正从事应用开发，我也需要一个干净、容器无关的会话机制。在当时，“这场游戏”中唯一可用的会话是**HttpSessions**，它需要Web容器；或是EJB 2.1里的有状态会话Bean，这又要EJB容器。而我想要的一个与容器脱钩、可用于任何我选择的环境中的会话。

最后就是加密问题。有时，我们需要保证数据安全，但是Java密码架构（**Java Cryptography Architecture**）让人难以理解，除非你是密码学专家。API里到处都是**Checked Exception**，用起来很麻烦。我需要一个干净、开箱即用的解决方案，可以在需要时方便地对数据加密/解密。

于是，纵观2003年初的安全状况，你会很快意识到还没有一个大一统的框架满足所有上述需求。有鉴于此，**JSecurity**（即之后的**Apache Shiro**）诞生了。

今天，你为何愿意使用Apache Shiro?

从2003年至今，框架选择方面的情况已经改变了不少，但今天仍有令人信服的理由让你选择**Shiro**。其实理由相当多，**Apache Shiro**：

深度内容

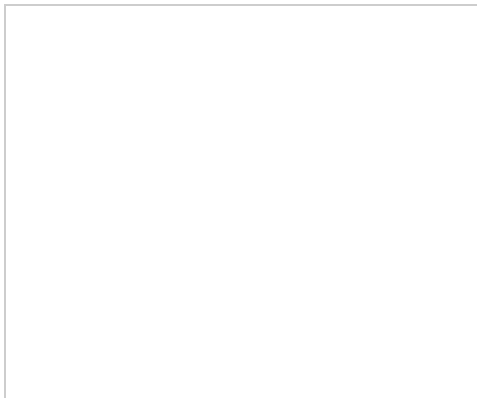
Naresh Jain：在不断演变的情境中应对变化

Naresh Jain曾获2007年度**Gordon Pask**奖。他的作品阐述了立足敏捷实践，灵活调整流程的必要性，因为没有放之四海而皆准的方法，流程必须随着我们解决越来越复杂的问题而不断演变。他对“精益创业（**Lean Startup**）”运动中出现的一些关键元素进行了考察，并展示了它们是如何成为许多敏捷实现理所当然的下一步。 **Naresh Jain** 2011年10月21日



扩展Oozie

在这篇文章中，作者展示了**Oozie**的可扩展性如何支持我们实现自定义的语言扩展。我们可以把这种方法视为针对特定的公司或者业务线的指定工作流语言。
Boris Lublinsky, Mike Segel 2011年10月20日



广告

敏捷项目的主动架构

主动架构是一种文档，在敏捷项目的用户故事和传统项目的大型设计交付物之间起到桥梁作用。它利用了用户故事的强大和简洁之处。与传统设计文档不同的是，**Active Architecture**定义了行动，或者说设计的主动状态，而传统设计文档定义的是设计的消极状态。 **Terry Bunio** 2011年10月19日



组织文化与敏捷：二者能否情投意合？

最近，敏捷教练**Michael Sahota**一直在探讨组织文化对敏捷转换的影响。我们抓到了他，并请他为我们的读者回答一些相关问题。 **Todd Charron** 2011年10月17日



从敏捷宣言理解敏捷交互设计

敏捷交互设计是敏捷方法论向交互

- 易于使用 - 易用性是这个项目的最终目标。应用安全有可能会非常让人糊涂，令人沮丧，并被认为是“必要之恶”【译注：比喻应用安全方面的编程。】。若是能将它简化到新手都能很快上手，那它将不再是一种痛苦了。
- 广泛性 - 没有其他安全框架可以达到Apache Shiro宣称的广度，它可以为你的安全需求提供“一站式”服务。
- 灵活性 - Apache Shiro可以工作在任何应用环境中。虽然它工作在Web、EJB和IoC环境中，但它并不依赖这些环境。Shiro既不强加任何规范，也无需过多依赖。
- Web能力 - Apache Shiro对Web应用的支持很神奇，允许你基于应用URL和Web协议（如REST）创建灵活的安全策略，同时还提供了一套控制页面输出的JSP标签库。
- 可插拔 - Shiro干净的API和设计模式使它可以方便地与许多的其他框架和应用进行集成。你将看到Shiro可以与诸如Spring、Grails、Wicket、Tapestry、Mule、Apache Camel、Vaadin这类第三方框架无缝集成。
- 支持 - Apache Shiro是Apache软件基金会成员，这是一个公认为了社区利益最大化而行动的组织。项目开发和用户组都有随时愿意提供帮助的友善成员。像Katasoft这类商业公司，还可以给你提供专业的支持和服务。

谁在用Shiro？

Shiro及其前身JSecurity已被各种规模和不同行业的项目采用多年。自从成为Apache软件基金会的顶级项目后，站点流量和使用呈持续增长态势。许多开源社区也正在用Shiro，这里有些例子如Spring、Grails、Wicket、Tapestry、Tynamo、Mule和Vaadin。

如Katasoft、Sonatype、MuleSoft这样的商业公司，一家大型社交网络和多家纽约商业银行都在使用Shiro来保护他们的商业软件和站点。

核心概念：Subject、SecurityManager和Realms

既然已经描述了Shiro的好处，那就让我们看看它的API，好让你能够有个感性认识。Shiro架构有三个主要概念 - Subject、SecurityManager和Realms。

Subject

在考虑应用安全时，你最常问的问题可能是“当前用户是谁？”或“当前用户允许做X吗？”。当我们写代码或设计用户界面时，问自己这些问题很平常：应用通常都是基于用户故事构建的，并且你希望功能描述（和安全）是基于每个用户的。所以，对于我们而言，考虑应用安全的最自然方式就是基于当前用户。Shiro的API用它的Subject概念从根本上体现了这种思考方式。

Subject一词是一个安全术语，其基本意思是“当前的操作用户”。称之为“用户”并不准确，因为“用户”一词通常跟人相关。在安全领域，术语“Subject”可以是人，也可以是第三方进程、后台帐户（Daemon Account）或其他类似事物。它仅仅意味着“当前跟软件交互的东西”。但考虑到大多数目的和用途，你可以把它认为是Shiro的“用户”概念。在代码的任何地方，你都能轻易的获得Shiro Subject，参见如下代码：

清单1. 获得Subject

```
import org.apache.shiro.subject.Subject;
import org.apache.shiro.SecurityUtils;
...
Subject currentUser = SecurityUtils.getSubject();
```

一旦获得Subject，你就可以立即获得你希望用Shiro为当前用户做的90%的事情，如登录、登出、访问会话、执行授权检查等 - 稍后还会看到更多。这里的关键点是Shiro的API非常直观，因为它反映了开发者以‘每个用户’思考安全控制的自然趋势。同时，在代码的任何地方都能很轻松地访问Subject，允许在任何需要的地方进行安全操作。

SecurityManager

Subject的“幕后”推手是SecurityManager。Subject代表了当前用户的安全操作，SecurityManager则管理所有用户的安全操作。它是Shiro框架的核心，充当“保护伞”，引用了多个内部嵌套安全组件，它们形成了对象图。但是，一旦SecurityManager及其内部对象图配置好，它就会退居幕后，应用开发人员几乎把他们的所有时间都花在Subject API调用上。

那么，如何设置SecurityManager呢？嗯，这要看应用的环境。例如，Web应用通常会在Web.xml中指定一个Shiro Servlet Filter，这会创建SecurityManager实例，如果你运行的是一个独立应用，你需要用其他配置方式，但有很多配置选项。

设计领域的延伸，它提倡让所有相关人参与到设计过程中，迭代演进式地进行交互设计。从2010年开始，已经有越来越多的团队在不同程度上使用敏捷交互设计的方法，而放弃了流程化的传统产品设计过程。事实上，敏捷交互设计方法在很多方面都充分体现了敏捷价值观，因此，理解敏捷交互设计实践的最好方法是从记录在敏捷宣言中的价值观开始。
熊子川 2011年10月14日



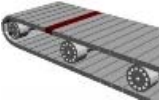
深入浅出Node.js（一）：什么是Node.js

为了更好地促进Node.js在国内的技术推广，我们决定开设辟“深入浅出Node.js”专栏，邀请来自Node.js领域的布道师、开发人员、技术专家来讲述Node.js的各方面内容。专栏的第一篇文章《什么是Node.js》尝试从各个角度来阐述Node.js的基本概念、发展历史、优势等，对该领域不熟悉的开发人员可以通过本文了解Node.js的一些基础知识。 崔康 2011年10月13日， 3



持续交付模式

实施持续交付，没有万能解决方案。团队的数目和构成会影响你的选择，以及要做出的妥协。资深编辑Jonathan Allen审视了他在过去十五年中观察到的一些模式。
Jonathan Allen 2011年10月12日



架构师（10月刊）

InfoQ中文站的电子杂志《架构师》（2011年10月刊）出炉了。本期的主编是InfoQ中文站原创翻译团队编辑李明。本期《架构师》月刊专题为大数据时代。就在你看这句话的短短数秒间，已有数以亿记GB的数据又生成了出来。是的，现在是一个数据爆炸的时代。大数据时代带给我们的不仅仅是一个概念，而是一种重新认识数据的机会。谁可以把握住这个机会，成为这个时代的宠儿，我们拭目以待。 InfoQ中文站 2011年10月12日



更早的 >

一个应用几乎总是只有一个SecurityManager实例。它实际是应用的Singleton（尽管不必是一个静态Singleton）。跟Shiro里的几乎所有组件一样，SecurityManager的缺省实现是POJO，而且可用POJO兼容的任何配置机制进行配置 - 普通的Java代码、Spring XML、YAML、.properties和.ini文件等。基本来讲，能够实例化类和调用JavaBean兼容方法的任何配置形式都可使用。

为此，Shiro借助基于文本的INI配置提供了一个缺省的“公共”解决方案。INI易于阅读、使用简单并且需要极少依赖。你还能看到，只要简单地理解对象导航，INI可被有效地用于配置像SecurityManager那样简单的对象图。注意，Shiro还支持Spring XML配置及其他方式，但这里只我们只讨论INI。

下列清单2列出了基于INI的Shiro最简配置：

清单2. 用INI配置Shiro

```
[main]
cm = org.apache.shiro.authc.credentials.HashedExceptionsMatcher
cm.hashAlgorithm = SHA-512
cm.hashIterations = 1024
# Base64 encoding (less text):
cm.storedCredentialsHexEncoded = false

iniRealm.credentialsMatcher = $cm

[users]
jdoe = TWFuIGl3IGRpc3RpbmdlaXNoZWQsIG5vdCBvbmx5IGJpcyByZWZzb2
asmith = IHNpbmd1bGFyIHBhc3Npb24gZnJvbnSBvdGhlciBhbXNoZWQsIG5vdCB
```

在清单2中，我们看到了用于配置SecurityManager实例的INI配置例子。有两个INI段落：[main]和[users]。

[main]段落是配置SecurityManager对象及其使用的其他任何对象（如Realms）的地方。在示例中，我们看到配置了两个对象：

- 1. cm对象，是Shiro的HashedExceptionsMatcher类实例。如你所见，cm实例的各属性是通过“嵌套点”语法进行配置的 - 在清单3中可以看到IniSecurityManagerFactory使用的惯例，这种方法代表了对象图导航和属性设置。
- 2. iniRealm对象，它被SecurityManager用来表示以INI格式定义的用户帐户。

[users]段落是指定用户帐户静态列表的地方 - 为简单应用或测试提供了方便。

就介绍而言，详细了解每个段落的细节并不是重点。相反，看到INI配置是一种配置Shiro的简单方式才是关键。关于INI配置的更多细节，请参见[Shiro文档](#)。

清单3. 装入shiro.ini配置文件

```
import org.apache.shiro.SecurityUtils;
import org.apache.shiro.config.IniSecurityManagerFactory;
import org.apache.shiro.mgt.SecurityManager;
import org.apache.shiro.util.Factory;

...

//1. 装入INI配置
Factory<SecurityManager> factory = new
IniSecurityManagerFactory("classpath:shiro.ini");

//2. 创建SecurityManager
SecurityManager securityManager = factory.getInstance();

//3. 使其可访问
SecurityUtils.setSecurityManager(securityManager);
```

在清单3的示例中，我们看到有三步：

- 1. 装入用来配置SecurityManager及其构成组件的INI配置文件；
- 2. 根据配置创建SecurityManager实例（使用Shiro的工厂概念，它表述了[工厂方法设计模式](#)）；
- 3. 使应用可访问SecurityManager Singleton。在这个简单示例中，我们将它设置为VM静态Singleton，但这通常不是必须的 - 你的应用配置机制可以决定你是否需要使用静态存储。

Realms

Shiro的第三个也是最后一个概念是Realm。Realm充当了Shiro与应用安全数据间的“桥梁”或者“连接器”。也就是说，当切实与像用户帐户这类安全相关数据进行交互，执行认证（登录）和授权（访问控制）时，Shiro会从应用配置的Realm中查找很多内容。

从这个意义上讲，**Realm**实质上是一个安全相关的**DAO**：它封装了数据源的连接细节，并在需要时将相关数据提供给**Shiro**。当配置**Shiro**时，你必须至少指定一个**Realm**，用于认证和（或）授权。配置多个**Realm**是可以的，但是至少需要一个。

Shiro内置了可以连接大量安全数据源（又名目录）的**Realm**，如**LDAP**、关系数据库（**JDBC**）、类似**INI**的文本配置资源以及属性文件等。如果缺省的**Realm**不能满足需求，你还可以插入代表自定义数据源的自己的**Realm**实现。下面的清单4是通过**INI**配置**Shiro**使用**LDAP**目录作为应用**Realm**的示例。

清单4. **Realm**配置示例片段：连接存储用户数据的**LDAP**

```
[main]
ldapRealm = org.apache.shiro.realm.ldap.JndiLdapRealm
ldapRealm.userDnTemplate = uid={0},ou=users,dc=mycompany,dc=com
ldapRealm.contextFactory.url = ldap://ldapHost:389
ldapRealm.contextFactory.authenticationMechanism = DIGEST-MD5
```

既然已经了解如何建立一个基本的**Shiro**环境，下面让我们来讨论，作为一名开发者该如何使用这个框架。

认证

认证是核实用户身份的过程。也就是说，当用户使用应用进行认证时，他们就在证明他们就是自己所说的那个人。有时这也理解为“登录”。它是一个典型的三步骤过程。

1. 收集用户的身份信息，称为当事人（**principal**），以及身份的支持证明，称为证书（**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/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）和支持他们的密码（证书）。如果存储在系统中的密码（或密码表示）与用户提供的匹配，他们就被认为通过认证。

Shiro以简单直观的方式支持同样的流程。正如我们前面所说，**Shiro**有一个以**Subject**为中心的**API** - 几乎你想要用**Shiro**在运行时完成的所有事情都能通过与当前执行的**Subject**进行交互而达成。因此，要登录**Subject**，只需要简单地调用它的**login**方法，传入表示被提交当事人和证书（在这种情况下，就是用户名和密码）的**AuthenticationToken**实例。示例如清单5中所示：

清单5. **Subject**登录

```
//1. 接受提交的当事人和证书：
AuthenticationToken token =
new UsernamePasswordToken(username, password);

//2. 获取当前Subject：
Subject currentUser = SecurityUtils.getSubject();
//3. 登录：
currentUser.login(token);
```

你可以看到，**Shiro**的**API**很容易地就反映了这个常见流程。你将会在所有的**Subject**操作中继续看到这种简单风格。在调用了**login**方法后，**SecurityManager**会收到**AuthenticationToken**，并将其发送给已配置的**Realm**，执行必须的认证检查。每个**Realm**都能在必要时对提交的**AuthenticationTokens**作出反应。但是如果登录失败了会发生什么？如果用户提供了错误密码又会发生什么？通过对**Shiro**的运行时**AuthenticationException**做出反应，你可以控制失败，参见清单6。

清单6. 控制失败的登录

```
//3. 登录：
try {
    currentUser.login(token);
} catch (IncorrectCredentialsException ice) {
    ...
} catch (LockedAccountException lae) {
    ...
}
...
catch (AuthenticationException ae) {...
}
```

你可以选择捕获**AuthenticationException**的一个子类，作出特定的响应，或者对任

何AuthenticationException做一般性处理（例如，显示给用户普通的“错误的用户名或密码”这类消息）。选择权在你，可以根据应用需要做出选择。

Subject登录成功后，他们就被认为是已认证的，通常会允许他们使用你的应用。但是仅仅证明了一个用户的身份并不意味着他们可以对你的应用为所欲为。这就引出了另一个问题，“我如何控制用户能做或不能做哪些事情？”，决定用户允许做哪些事情的过程被称为授权。下面我们将谈谈Shiro如何进行授权。

授权

授权实质上就是访问控制 - 控制用户能够访问应用中的哪些内容，比如资源、Web页面等等。多数用户执行访问控制是通过使用诸如角色和权限这类概念完成的。也就是说，通常用户允许或不允许做的事情是根据分配给他们的角色或权限决定的。那么，通过检查这些角色和权限，你的应用程序就可以控制哪些功能是可以暴露的。如你期望的，Subject API让你可以很容易的执行角色和权限检查。如清单7中的代码片段所示：如何检查Subject被分配了某个角色：

列表7. 角色检查

```
if ( subject.hasRole("administrator") ) {
    //显示‘Create User’按钮
} else {
    //按钮置灰？
}
```

如你所见，你的应用程序可基于访问控制检查打开或关闭某些功能。

权限检查是执行授权的另一种方法。上例中的角色检查有个很大的缺陷：你无法在运行时增删角色。角色名字在这里是硬编码，所以，如果你修改了角色名字或配置，你的代码就会乱套！如果你需要在运行时改变角色含义，或想要增删角色，你必须另辟蹊径。

为此，Shiro支持了权限（permissions）概念。权限是功能的原始表述，如‘开门’，‘创建一个博文’，‘删除jsmith’用户’等。通过让权限反映应用的原始功能，在改变应用功能时，你只需要改变权限检查。进而，你可以在运行时按需将权限分配给角色或用户。

如清单8中，我们重写了之前的用户检查，取而代之使用权限检查。

清单8. 权限检查

```
if ( subject.isPermitted("user:create") ) {
    //显示‘Create User’按钮
} else {
    //按钮置灰？
}
```

这样，任何具有“user:create”权限的角色或用户都可以点击‘Create User’按钮，并且这些角色和指派甚至可以在运行时改变，这给你提供了一个非常灵活的安全模型。

“user:create”字符串是一个权限字符串的例子，它遵循特定的解析惯例。Shiro借助它的WildcardPermission支持这种开箱即用的惯例。尽管这超出了本文的范围，你会看到在创建安全策略时，WildcardPermission非常灵活，甚至支持像实例级别访问控制这样的功能。

清单9. 实例级别的权限检查

```
if ( subject.isPermitted("user:delete:jsmith") ) {
    //删除‘jsmith’用户
} else {
    //不删除‘jsmith’
}
```

该例表明，你可以对你需要的单个资源进行访问控制，甚至深入到非常细粒度的实例级别。如果愿意，你甚至还可以发明自己的权限语法。参见[Shiro Permission](#)文档可以了解更多内容。最后，就像使用认证那样，上述调用最终会转向SecurityManager，它会咨询Realm做出自己的访问控制决定。必要时，还允许单个Realm同时响应认证和授权操作。

以上就是对Shiro授权功能的简要概述。虽然多数安全框架止于授权和认证，但Shiro提供了更多功能。下面，我们将谈谈Shiro的高级会话管理功能。

会话管理

在安全框架领域，Apache Shiro提供了一些独特的东西：可在任何应用或架构层一致地使用Session API。即，Shiro为任何应用提供了一个会话编程范式 - 从小型后台独立应用到大型集群Web应用。这意味

着，那些希望使用会话的应用开发者，不必被迫使用Servlet或EJB容器了。或者，如果正在使用这些容器，开发者现在也可以选择使用在任何层统一一致的会话API，取代Servlet或EJB机制。

但Shiro会话最重要的一个好处或许就是它们是独立于容器的。这具有微妙但非常强大的影响。例如，让我们考虑一下会话集群。对集群会话来讲，支持容错和故障转移有多少种容器特定的方式？Tomcat的方式与Jetty的不同，而Jetty又和Websphere不一样，等等。但通过Shiro会话，你可以获得一个容器无关的集群解决方案。Shiro的架构允许可插拔的会话数据存储，如企业缓存、关系数据库、NoSQL系统等。这意味着，只要配置会话集群一次，它就会以相同的方式工作，跟部署环境无关 - Tomcat、Jetty、JEE服务器或者独立应用。不管如何部署应用，毋须重新配置应用。

Shiro会话的另一好处就是，如果需要，会话数据可以跨客户端技术进行共享。例如，Swing桌面客户端在需要时可以参与相同的Web应用会话中 - 如果最终用户同时使用这两种应用，这样的功能会很有用。那你如何在任何环境中访问Subject的会话呢？请看下面的示例，里面使用了Subject的两个方法。

清单10. Subject的会话

```
Session session = subject.getSession();
Session session = subject.getSession(boolean create);
```

如你所见，这些方法在概念上等同于HttpServletRequest API。第一个方法会返回Subject的现有会话，或者如果还没有会话，它会创建一个新的并将之返回。第二个方法接受一个布尔参数，这个参数用于判定会话不存在时是否创建新会话。一旦获得Shiro的会话，你几乎可以像使用HttpSession一样使用它。Shiro团队觉得对于Java开发者，HttpSession API用起来太舒服了，所以我们保留了它的很多感觉。当然，最大的不同在于，你可以在任何应用中使用Shiro会话，不仅限于Web应用。清单11中显示了这种相似性。

清单11. 会话的方法

```
Session session = subject.getSession();
session.getAttribute("key", someValue);
Date start = session.getStartTimestamp();
Date timestamp = session.getLastAccessTime();
session.setTimeout(millis); ...
```

加密

加密是隐藏或混淆数据以避免被偷窥的过程。在加密方面，Shiro的目标是简化并让JDK的加密支持可用。

清楚一点很重要，一般情况下，加密不是特定于Subject的，所以它是Shiro API的一部分，但并不特定于Subject。你可以在任何地方使用Shiro的加密支持，甚至在不使用Subject的情况下。对于加密支持，Shiro真正关注的两个领域是加密哈希（又名消息摘要）和加密密码。下面我们来看看这两个方面的详细描述。

哈希

如果你曾使用过JDK的MessageDigest类，你会立刻意识到它的使用有点麻烦。MessageDigest类有一个笨拙的基于工厂的静态方法API，它不是面向对象的，并且你被迫去捕获那些永远都不必捕获的Checked Exceptions。如果需要输出十六进制编码或Base64编码的消息摘要，你只有靠自己 - 对上述两种编码，没有标准的JDK支持它们。Shiro用一种干净而直观的哈希API解决了上述问题。

打个比方，考虑比较常见的情况，使用MD5哈希一个文件，并确定该哈希的十六进制值。被称为‘校验和’，这在提供文件下载时常用到 - 用户可以对下载文件执行自己的MD5哈希。如果它们匹配，用户完全可以认定文件在传输过程中没有被篡改。

不使用Shiro，你需要如下步骤才能完成上述内容：

- 1. 将文件转换成字节数组。JDK中没有干这事的，故而你需要创建一个辅助方法用于打开FileInputStream，使用字节缓存区，并抛出相关的IOExceptions，等等。
- 2. 使用MessageDigest类对字节数组进行哈希，处理相关异常，如清单12所示。
- 3. 将哈希后的字节数组编码成十六进制字符。JDK中还是没有干这事的，你依旧需要创建另外一个辅助方法，有可能在你的实现中会使用位操作和位移动。

清单12. JDK的消息摘要

```
try {
    MessageDigest md = MessageDigest.getInstance("MD5");
    md.digest(bytes);
    byte[] hashed = md.digest();
} catch (NoSuchAlgorithmException e) {
```

```
e.printStackTrace();
}
```

对于这样简单普遍的需求，这个工作量实在太大了。现在看看**Shiro**是如何做同样事情的：

```
String hex = new Md5Hash(myFile).toHex();
```

当使用**Shiro**简化所有这些工作时，一切都非常简单明了。完成**SHA-512**哈希和密码的**Base64**编码也一样简单。

```
String encodedPassword = new Sha512Hash(password, salt, count).toBase64();
```

你可以看到**Shiro**对哈希和编码简化了不少，挽救了你处理在这类问题上所消耗的脑细胞。

密码

加密是使用密钥对数据进行可逆转换的加密算法。我们使用其保证数据的安全，尤其是传输或存储数据时，以及在数据容易被窥探的时候。

如果你曾经用过JDK的**Cryptography** API，特别是**javax.crypto.Cipher**类，你会知道它是一头需要驯服的极其复杂的野兽。对于初学者，每个可能的加密配置总是由一个**javax.crypto.Cipher**实例表示。必须进行公钥/私钥加密？你得用**Cipher**。需要为流操作使用块加密器（**Block Cipher**）？你得用**Cipher**。需要创建一个**AES 256**位**Cipher**来保护数据？你得用**Cipher**。你懂的。

那么如何创建你需要的**Cipher**实例？您得创建一个非直观、标记分隔的加密选项字符串，它被称为“转换字符串（**transformation string**）”，将该字符串传给**Cipher.getInstance**静态工厂方法。这种字符串方式的**cipher**选项，并没有类型安全以确保你正在用有效的选项。这也暗示没有**JavaDoc**帮你了解相关选项。并且，如果字符串格式组织不正确，你还需要进一步处理**Checked Exception**，即便你知道配置是正确的。如你所见，使用**JDK Cipher**是一项相当繁重的任务。很久以前，这些技术曾经是**Java API**的标准，但是世事变迁，我们需要一种更简单的方法。

Shiro通过引入它的**CipherService** API试图简化加密密码的整个概念。**CipherService**是多数开发者在保护数据时梦寐以求的东西：简单、无状态、线程安全的**API**，能够在一次方法调用中对整个数据进行加密或解密。你所需要做的只是提供你的密钥，就可根据需要加密或解密。如下列清单**13**中，使用**256**位**AES**加密：

清单13. Apache Shiro的加密API

```
AesCipherService cipherService = new AesCipherService();
cipherService.setKeySize(256);

// 创建一个测试密钥：
byte[] testKey = cipherService.generateNewKey();
// 加密文件的字节：
byte[] encrypted = cipherService.encrypt(fileBytes, testKey);
```

较之JDK的**Cipher** API，**Shiro**的示例要简单的多：

- 你可以直接实例化一个**CipherService** - 没有奇怪或让人混乱的工厂方法；
- **Cipher**配置选项可以表示成**JavaBean** - 兼容的**getter**和**setter**方法 - 没有了奇怪和难以理解的“转换字符串”；
- 加密和解密在单个方法调用中完成；
- 没有强加的**Checked Exception**。如果愿意，可以捕获**Shiro**的**CryptoException**。

Shiro的**CipherService** API还有其他好处，如同时支持基于字节数组的加密/解密（称为“块”操作）和基于流的加密/解密（如加密音频或视频）。

不必再忍受**Java Cryptography**带来的痛苦。**Shiro**的**Cryptography**支持就是为了减少你在确保数据安全上付出的努力。

Web支持

最后，但并非不重要，我们将简单介绍一下**Shiro**的**Web**支持。**Shiro**附带了一个帮助保护**Web**应用的强建的**Web**支持模块。对于**Web**应用，安装**Shiro**很简单。唯一需要做的就是**在web.xml中定义一个Shiro Servlet过滤器**。清单**14**中，列出了代码。

清单14. web.xml中的ShiroFilter

```
<filter>
    <filter-name>ShiroFilter</filter-name>
```

```
<filter-class>
    org.apache.shiro.web.servlet.IniShiroFilter
</filter-class>
<!-- 没有init-param属性就表示从classpath:shiro.ini装入INI配置 -->
</filter>
<filter-mapping>
    <filter-name>ShiroFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

这个过滤器可以读取上述shiro.ini配置，这样不论什么开发环境，你都拥有了一致的配置体验。一旦完成配置，Shiro Filter就会过滤每个请求并且确保在请求期间特定请求的Subject是可访问的。同时由于它过滤了每个请求，你可以执行安全特定的逻辑以保证只有满足一定标准的请求才被允许通过。

URL特定的Filter链

Shiro通过其创新的URL过滤器链功能支持安全特定的过滤规则。它允许你为任何匹配的URL模式指定非正式的过滤器链。这意味着，使用Shiro的过滤器机制，你可以很灵活的强制安全规则（或者规则的组 合） - 其程度远远超过你单独在web.xml中定义过滤器时所获得的。清单15中显示了Shiro INI中的配置 片段。

清单15. 路径特定的Filter链

```
[urls]
/assets/** = anon
/user/signup = anon
/user/** = user
/rpc/rest/** = perms[rpc:invoke], authc
/** = authc
```

如你所见，Web应用可以使用[urls] INI段落。对于每一行，等号左边的值表示相对上下文的Web应用路 径。等号右边的值定义了过滤器链 - 一个逗号分隔的有序Servlet过滤器列表，它会针对给出的路径进行 执行。每个过滤器都是普通的Servlet过滤器，你看到的上面的过滤器名字（anon, user, perms, authc）是Shiro内置的安全相关的特殊过滤器。你可以搭配这些安全过滤器来创 建高度定制的安全体验。你还可以指定任何其他现有的Servlet过滤器。

相比起使用web.xml，在其中先定义过滤区块，然后定义单独分离的过滤器模式块，这种方式带来的好处 有多少？采用Shiro的方法，可以很容易就准确知道针对给定匹配路径执行的过滤器链。如果想这么做， 你可以在web.xml中仅定义Shiro Filter，在shiro.ini中定义所有其他的过滤器和过滤器链，这要 比web.xml简洁得多，而且更容易理解过滤器链定义机制。即使不使用Shiro的任何安全特性，单凭这样 小小的方便之处，也值得让你使用Shiro。

JSP标签库

Shiro还提供了JSP标签库，允许你根据当前Subject的状态控制JSP页面的输出。一个有用的常见示例是在 用户登录后显示“Hello <username>”文本。但若是匿名用户，你可能想要显示其他内容，如换而显 示“Hello! Register Today!”。清单16显示了如何使用Shiro的JSP标签实现这个示例：

清单16. JSP 标签库示例

```
<%@ taglib prefix="shiro"
    uri="http://shiro.apache.org/tags" %>
...
<p>Hello
<shiro:user>
    <!-- shiro:principal打印出了Subject的主当事人 - 在这个示例中，就是用户名: -->
    <shiro:principal/>
</shiro:user>
<shiro:guest>
    <!-- 没有登录 - 就认为是Guest。显示注册链接: -->
    ! <a href="register.jsp">Register today!</a>
</shiro:guest>
</p>
```

除了上面例子用到的标签，还有其他标签可以让你根据用户属于（或不属于）的角色，分配（或未分 配）的权限，是否已认证，是否来自“记住我”服务的记忆，或是匿名访客，包含输出。

Shiro还支持其他许多Web特性，如简单的“记住我”服务，REST和BASIC认证。当然，如果想使用Shiro原 生的企业会话，它还提供透明的HttpSession支持。参见[Apache Shiro Web文档](#)可以了解更多内容。

Web会话管理

最后值得一提的是Shiro在Web环境中对会话的支持。

缺省Http会话

对于Web应用，Shiro缺省将使用我们习以为常的Servlet容器会话作为其会话基础设施。即，当你调用`subject.getSession()`和`subject.getSession(boolean)`方法时，Shiro会返回Servlet容器的`HttpSession`实例支持的Session实例。这种方式的曼妙之处在于调用`subject.getSession()`的业务层代码会跟一个Shiro Session实例交互 - 还没有“认识”到它正跟一个基于Web的`HttpSession`打交道。这在维护架构层之间的清晰隔离时，是一件非常好的事情。

Web层中Shiro的原生会话

如果你由于需要Shiro的企业级会话特性（如容器无关的集群）而打开了Shiro的原生会话管理，你当然希望`HttpServletRequest.getSession()`和`HttpSession API`能和“原生”会话协作，而非Servlet容器会话。如果你不得不重构所有使用`HttpServletRequest`和`HttpSession API`的代码，使用Shiro的Session API来替换，这将非常令人沮丧。Shiro当然从来不会期望你这么做。相反，Shiro完整实现了Servlet规范中的Session部分以在Web应用中支持原生会话。这意味着，不管何时你使用相应的`HttpServletRequest`或`HttpSession`方法调用，Shiro都会将这些调用委托给内部的原生会话API。结果，你无需修改Web代码，即便是你正在使用Shiro的‘原生’企业会话管理 - 确实是一个非常方便（且必要）的特性。

附加特性

Apache Shiro框架还包含有对保护Java应用非常有用的其他特性，如：

- 为维持跨线程的Subject提供了线程和并发支持（支持Executor和ExecutorService）；
- 为了将执行逻辑作为一种特殊的Subject，支持Callable和Runnable接口；
- 为了表现为另一个Subject的身份，支持“Run As”（比如，在管理应用中有用）；
- 支持测试工具，这样可以很容易的对Shiro的安全代码进行单元测试和集成测试。

框架局限

常识告诉我们，Apache Shiro不是“银弹” - 它不能毫不费力的解决所有安全问题。如下是Shiro还未解决，但是值得知道的：

- 虚拟机级别的问题：Apache Shiro当前还未处理虚拟机级别的安全，比如基于访问控制策略，阻止类加载器中装入某个类。然而，Shiro集成现有的JVM安全操作并非白日做梦 - 只是没人给项目贡献这方面的工作。
- 多阶段认证：目前，Shiro不支持“多阶段”认证，即用户可能通过一种机制登录，当被要求再次登录时，使用另一种机制登录。这在基于Shiro的应用中已经实现，但是通过应用预先收集所有必需信息再跟Shiro交互。这个功能在Shiro的未来版本中非常有可能得到支持。
- Realm写操作：目前所有Realm实现都支持“读”操作来获取验证和授权数据以执行登录和访问控制。诸如创建用户帐户、组和角色或与用户相关的角色组和权限这类“写”操作还不支持。这是因为支持这些操作的应用数据模型变化太大，很难为所有的Shiro用户强制定义“写”API。

未来的特性

Apache Shiro社区每天都在壮大，借此，Shiro的特性亦是如此。在即将发布的版本中，你可能会看到：

- 更干净的Web过滤机制，无需子类化就可支持更多的插件式过滤器。
- 更多可插拔的缺省Realm实现，优先采用组合而非继承。你可以插入查找认证和授权数据的组件，无需实现为Shiro Realm的子类；
- 强健的OpenID和OAuth（可能是混合）客户端支持；
- 支持Captcha；
- 针对纯无状态应用的配置简化（如，许多REST环境）；
- 通过请求/响应协议进行多阶段认证；
- 通过AuthorizationRequest进行粗粒度的授权；
- 针对安全断言查询的ANTLR语法（比如，(‘role(admin) && (guest || !group(developer))’）

总结

Apache Shiro是一个功能齐全、健壮、通用的Java安全框架，你可以用其为你的应用护航。通过简化应用安全的四个领域，即认证、授权、会话管理和加密，在真实应用中，应用安全能更容易被理解和实现。Shiro的简单架构和兼容JavaBean使其几乎能够在任何环境下配置和使用。附加的Web支持和辅助功能，比如多线程和测试支持，让这个框架为应用安全提供了“一站式”服务。Apache Shiro开发团队将继续

前进，精炼代码库和支持社区。随着持续被开源和商业应用采纳，可以预期Shiro会继续发展壮大。

资源

- Apache Shiro的[主页](#)。
- Shiro的[下载页面](#)，包含面向Maven和Ant+ Ivy用户的额外信息。
- Apache Shiro的[文档页面](#)，包含指南和参考手册。
- [Apache Shiro演讲视频和幻灯](#)，项目的PMC主席Les Hazlewood提供。
- 其他关于Apache Shiro的[文章和幻灯](#)
- Apache Shiro的[邮件列表](#)和[论坛](#)。
- [Katasoft](#) - 提供Apache Shiro专业支持和应用安全产品的公司。

关于作者

Les Hazlewood是Apache Shiro PMC主席以及[Katasoft](#)的CTO和合伙创办人，该公司是专注于应用安全产品以及提供Apache Shiro专业支持的创业公司。作为专业Java开发人员及企业架构师，以及之前Bloomberg，Delta Airlines和JBoss的高级角色，Les拥有10年的经验。Les积极从事开源开发已有9年时间，提交或做出贡献的项目有Spring框架、Hibernate、JBoss，OpenSpaces，当然还有JSecurity，Apache Shiro的前身。Les目前居住在加州的圣马特奥，不编程时，他会练习剑道和学习日语。

查看英文原文：[Application Security With Apache Shiro](#)

感谢[胡键](#)对本文的审校。

给InfoQ中文站投稿或者参与内容翻译工作，请邮件至[editors@cn.infoq.com](#)。也欢迎大家加入到[InfoQ中文站用户讨论组](#)中与我们的编辑和其他读者朋友交流。

14 条回复  

▸ 比spring security清爽多了 发表人 YANG LiN 发表于 11/05/2011 04:24

▸ Re: 比spring security清爽多了 发表人 刘 晓日 发表于 11/05/2011 09:03

▸ 比ralasafe呢？ 发表人 Wong Julian 发表于 12/05/2011 11:00

▸ Re: 比ralasafe呢？ 发表人 曹 云飞 发表于 13/05/2011 09:47

▸ Re: 比ralasafe呢？ 发表人 li jie 发表于 03/08/2011 03:55

▸ Re: 比ralasafe呢？ 发表人 刘 观岗 发表于 大约 23 小时 之前

▸ 谢谢伟红的翻译 发表人 xiu calvin 发表于 12/05/2011 11:41

▸ Re: 谢谢伟红的翻译 发表人 胡 伟红 发表于 12/05/2011 04:13

▸ Re: 谢谢伟红的翻译 发表人 胡 伟红 发表于 12/05/2011 08:35

▸ 真的很不错哦，我在项目里面已经用上了。就是文档方面..有些要看源码了 发表人 陈 桂忠 发表于 13/05/2011 05:33

▸ Re: 真的很不错哦，我在项目里面已经用上了。就是文档方面..有些要看源码了 发表人 watano super 发表于 16/05/2011 10:06

▸ Re: 真的很不错哦，我在项目里面已经用上了。就是文档方面..有些要看源码了 发表人 watano super 发表于 16/05/2011 10:08

▸ 条理清晰 发表人 张 艺聆 发表于 16/05/2011 12:03

▸ 比SpringSecurity如何？ 发表人 Lee Vincent 发表于 25/05/2011 09:39

按日期倒序排列

比spring security清爽多了

11/05/2011 04:24 发表人 YANG LiN

清爽



Re: 比spring security清爽多了

11/05/2011 09:03 发表人 刘 晓日

前段时间看的e文的，速度有些慢啊。不过总比木有要好。嘿嘿

http://www.infoq.com/cn/articles/apache-shiro[2011/10/21 17:39:51]

同意你的评论，从框架图上说就清晰太多了。
脱离servlet容器，吼吼，就是没试过。目前在用spring security。

 回复

比ralasafe呢?



12/05/2011 11:00 发表人 **Wong Julian**

我觉得ralasafe更有意思。（呵呵，我是ralasafe主创）

 回复

谢谢伟红的翻译



12/05/2011 11:41 发表人 **xiu calvin**

谢谢伟红的翻译，项目在用Shiro，有这个中文版的介绍文章，给其他Team的同事做介绍就更方便了。

 回复

Re: 谢谢伟红的翻译



12/05/2011 04:13 发表人 胡 伟红

我之前在IBM DW上也有一篇关于shiro的文章，有兴趣可以看看。

 回复

Re: 谢谢伟红的翻译



12/05/2011 08:35 发表人 胡 伟红

另外，Grails中也有Shiro的插件，详情请看[《Grails Shiro Plugin之初体验》](#)了解详情。

 回复

Re: 比ralasafe呢?



13/05/2011 09:47 发表人 曹 云飞

这个比较，您最有发言权呀。

 回复

真的很不错哦，我在项目里面已经用上了。就是文档方面..有些要看源码了



13/05/2011 05:33 发表人 陈 桂忠

真的很不错哦，我今年2月份开工的项目里面已经用上了。就是文档方面..有些要看源码了。速度也是赶赶滴哦。配上ehcache验证无法计时！

 回复

Re: 真的很不错哦，我在项目里面已经用上了。就是文档方面..有些要看源码了



16/05/2011 10:06 发表人 **watano super**

这个东东真的很不错的,去年年初我们团队就在项目中使用,整体感觉思路相当清晰,文档也简单,容易上手,缺点就是文档实在太少了,基本要看源码,还有就是转到apache下以后更新实在太慢了,感觉apache shiro团队实在是不作为啊,现在这个版本和JSecurity时基本没什么变化,除了包名换了.

 回复

Re: 真的很不错哦，我在项目里面已经用上了。就是文档方面..有些要看源码了



16/05/2011 10:08 发表人 watano super

呵呵,错了,是前年年初,那时候还没有转到apache下,还叫JSecurity,中途改了好几次名字.

回复

条理清晰

16/05/2011 12:03 发表人 张 艺聆

条理清晰,丝丝入扣

回复

比SpringSecurity如何?

25/05/2011 09:39 发表人 Lee Vincent

支持CAS等单点登录服务么?
ssl的控制怎么样?

回复

Re: 比ralasafe呢?

03/08/2011 03:55 发表人 li jie

您认了吧。。。

回复

Re: 比ralasafe呢?

大约 23 小时 之前 发表人 刘 观岗

ralasafe用得我想撞墙!

回复



首页

关于我们

合作伙伴

Qcon Conferences

Create account

登录

提供新闻

选择默认领域

☒ 语言 & 开发

☒ 架构 & 设计

☒ 过程 & 实践

☒ 运维 & 基础架构

☒ 企业架构

特别专题

技术社区活动日历

Flash Platform

百度技术沙龙

Scrum开发

月刊:《架构师》

线下活动: QClub

EVENTS

QCon Hangzhou Oct 21-22, 2011

QCon San Francisco November 16-18, 2011

反馈

feedback@cn.infoq.com

Bugs

bugs@cn.infoq.com

广告

sales@cn.infoq.com

编辑

editors@cn.infoq.com

Twitter

http://twitter.com/infoqchina

InfoQ.com 及其所有内容, 版权所有 © 2006-2011 C4Media Inc. InfoQ.com 服务器由 Contegix 提供, 我们最信赖的 ISP 合作伙伴。

隐私政策