1. **简介**

Apache Shiro是Java的一个安全框架。功能强大，使用简单的Java安全框架，它为开发人员提供一个直观而全面的认证，授权，加密及会话管理的解决方案。

实际上，Shiro的主要功能是管理应用程序中与安全相关的全部，同时尽可能支持多种实现方法。Shiro是建立在完善的接口驱动设计和面向对象原则之上的，支持各种自定义行为。Shiro提供的默认实现，使其能完成与其他安全框架同样的功能，这不也是我们一直努力想要得到的吗！

Apache Shiro相当简单，对比Spring Security，可能没有Spring Security做的功能强大，但是在实际工作时可能并不需要那么复杂的东西，所以使用小而简单的Shiro就足够了。对于它俩到底哪个好，这个不必纠结，能更简单的解决项目问题就好了。

Shiro可以非常容易的开发出足够好的应用，其不仅可以用在JavaSE环境，也可以用在JavaEE环境。Shiro可以帮助我们完成：认证、授权、加密、会话管理、与Web集成、缓存等。这不就是我们想要的嘛，而且Shiro的API也是非常简单；其基本功能点如下图所示：



**Authentication：**身份认证/登录，验证用户是不是拥有相应的身份；

**Authorization：**授权，即权限验证，验证某个已认证的用户是否拥有某个权限；即判断用户是否能做事情，常见的如：验证某个用户是否拥有某个角色。或者细粒度的验证某个用户对某个资源是否具有某个权限；

**Session Manager：**会话管理，即用户登录后就是一次会话，在没有退出之前，它的所有信息都在会话中；会话可以是普通JavaSE环境的，也可以是如Web环境的；

**Cryptography：**加密，保护数据的安全性，如密码加密存储到数据库，而不是明文存储；

**Web Support：**Web支持，可以非常容易的集成到Web环境；

Caching：缓存，比如用户登录后，其用户信息、拥有的角色/权限不必每次去查，这样可以提高效率；

**Concurrency：**shiro支持多线程应用的并发验证，即如在一个线程中开启另一个线程，能把权限自动传播过去；

**Testing：**提供测试支持；

**Run As：**允许一个用户假装为另一个用户（如果他们允许）的身份进行访问；

**Remember Me：**记住我，这个是非常常见的功能，即一次登录后，下次再来的话不用登录了。

**记住一点，Shiro不会去维护用户、维护权限；这些需要我们自己去设计/提供；然后通过相应的接口注入给Shiro即可。**

接下来我们分别从外部和内部来看看Shiro的架构，对于一个好的框架，从外部来看应该具有非常简单易于使用的API，且API契约明确；从内部来看的话，其应该有一个可扩展的架构，即非常容易插入用户自定义实现，因为任何框架都不能满足所有需求。

首先，我们从外部来看Shiro吧，即从应用程序角度的来观察如何使用Shiro完成工作。如下图：



可以看到：应用代码直接交互的对象是Subject，也就是说Shiro的对外API核心就是Subject；其每个API的含义：

**Subject：**主体，代表了当前“用户”，这个用户不一定是一个具体的人，与当前应用交互的任何东西都是Subject，如网络爬虫，机器人等；即一个抽象概念；所有Subject都绑定到SecurityManager，与Subject的所有交互都会委托给SecurityManager；可以把Subject认为是一个门面；SecurityManager才是实际的执行者；

**SecurityManager：**安全管理器；即所有与安全有关的操作都会与SecurityManager交互；且它管理着所有Subject；可以看出它是Shiro的核心，它负责与后边介绍的其他组件进行交互，如果学习过SpringMVC，你可以把它看成DispatcherServlet前端控制器；

**Realm：**域，Shiro从Realm获取安全数据（如用户、角色、权限），就是说SecurityManager要验证用户身份，那么它需要从Realm获取相应的用户进行比较以确定用户身份是否合法；也需要从Realm得到用户相应的角色/权限进行验证用户是否能进行操作；可以把Realm看成DataSource，即安全数据源。

也就是说对于我们而言，最简单的一个Shiro应用：

1、应用代码通过Subject来进行认证和授权，而Subject又委托给SecurityManager；

2、我们需要给Shiro的SecurityManager注入Realm，从而让SecurityManager能得到合法的用户及其权限进行判断。

**从以上也可以看出，Shiro不提供维护用户/权限，而是通过Realm让开发人员自己注入。**

接下来我们来从Shiro内部来看下Shiro的架构，如下图所示：



**Subject：**主体，可以看到主体可以是任何可以与应用交互的“用户”；

**SecurityManager：**相当于SpringMVC中的DispatcherServlet或者Struts2中的FilterDispatcher；是Shiro的心脏；所有具体的交互都通过SecurityManager进行控制；它管理着所有Subject、且负责进行认证和授权、及会话、缓存的管理。

**Authenticator：**认证器，负责主体认证的，这是一个扩展点，如果用户觉得Shiro默认的不好，可以自定义实现；其需要认证策略（Authentication Strategy），即什么情况下算用户认证通过了；

**Authrizer：**授权器，或者访问控制器，用来决定主体是否有权限进行相应的操作；即控制着用户能访问应用中的哪些功能；

**Realm：**可以有1个或多个Realm，可以认为是安全实体数据源，即用于获取安全实体的；可以是JDBC实现，也可以是LDAP实现，或者内存实现等等；由用户提供；注意：Shiro不知道你的用户/权限存储在哪及以何种格式存储；所以我们一般在应用中都需要实现自己的Realm；

**SessionManager：**如果写过Servlet就应该知道Session的概念，Session呢需要有人去管理它的生命周期，这个组件就是SessionManager；而Shiro并不仅仅可以用在Web环境，也可以用在如普通的JavaSE环境、EJB等环境；所有呢，Shiro就抽象了一个自己的Session来管理主体与应用之间交互的数据；这样的话，比如我们在Web环境用，刚开始是一台Web服务器；接着又上了台EJB服务器；这时想把两台服务器的会话数据放到一个地方，这个时候就可以实现自己的分布式会话（如把数据放到Memcached服务器）；

**SessionDAO：**DAO大家都用过，数据访问对象，用于会话的CRUD，比如我们想把Session保存到数据库，那么可以实现自己的SessionDAO，通过如JDBC写到数据库；比如想把Session放到Memcached中，可以实现自己的Memcached SessionDAO；另外SessionDAO中可以使用Cache进行缓存，以提高性能；

**CacheManager：**缓存控制器，来管理如用户、角色、权限等的缓存的；因为这些数据基本上很少去改变，放到缓存中后可以提高访问的性能

**Cryptography：**密码模块，Shiro提高了一些常见的加密组件用于如密码加密/解密的。

1. **过滤器 权限拦截器**

|  |  |
| --- | --- |
| 过滤器简称 | 对应的java类 |
| anon | org.apache.shiro.web.filter.authc.AnonymousFilter |
| authc | org.apache.shiro.web.filter.authc.FormAuthenticationFilter |
| authcBasic | org.apache.shiro.web.filter.authc.BasicHttpAuthenticationFilter |
| perms | org.apache.shiro.web.filter.authz.PermissionsAuthorizationFilter |
| port | org.apache.shiro.web.filter.authz.PortFilter |
| rest | org.apache.shiro.web.filter.authz.HttpMethodPermissionFilter |
| roles | org.apache.shiro.web.filter.authz.RolesAuthorizationFilter |
| ssl | org.apache.shiro.web.filter.authz.SslFilter |
| user | org.apache.shiro.web.filter.authc.UserFilter |
| logout | org.apache.shiro.web.filter.authc.LogoutFilter |

1. **Jsp shiro标签**

|  |  |
| --- | --- |
| **标签名称** | **标签条件（均是显示标签内容）** |
| <shiro:authenticated> | 登录之后 |
| <shiro:notAuthenticated> | 不在登录状态时 |
| <shiro:guest> | 用户在没有RememberMe时 |
| <shiro:user> | 用户在RememberMe时 |
| <shiro:hasAnyRoles name=*"abc,123" >* | 在有abc或者123角色时 |
| <shiro:hasRole name=*"abc">* | 拥有角色abc |
| <shiro:lacksRole name=*"abc">* | 没有角色abc |
| <shiro:hasPermission name=*"abc">* | 拥有权限资源abc |
| <shiro:lacksPermission name=*"abc">* | 没有abc权限资源 |
| <shiro:principal> | 默认显示用户名称 |

1. **Spring security 与apache shiro 差别：**

shiro配置更加容易理解，容易上手；security配置相对比较难懂。

在spring的环境下，security整合性更好。Shiro对很多其他的框架兼容性更好，号称是无缝集成。

shiro 不仅仅可以使用在web中，它可以工作在任何应用环境中。

在集群会话时Shiro最重要的一个好处或许就是它的会话是独立于容器的。

Shiro提供的密码加密使用起来非常方便。

1. **控制精度：**

Shiro也支持注解方式。

注解方式控制权限只能是在方法上控制，无法控制类级别访问。

过滤器方式控制是根据访问的URL进行控制。允许使用\*匹配URL，所以可以进行粗粒度，也可以进行细粒度控制。