# Shiro

# Shiro简介

Shiro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。

从应用程序角度的来观察如何使用Shiro完成工作。如下图：



**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。

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。

# Shiro使用

## 引入jar

shiro-all-1.3.2.jar(或shiro-core)、log4、slf4j-api-1.6.1.jar、slfj-log4j12-1.6.1.jarj

maven引入：

<!-- shiro -->

<dependency>

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

<artifactId>shiro-all</artifactId>

<version>1.3.2</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.1.3</version>

</dependency>

# 集成Spring

## 基本搭建

### 创建web工程

搭建spring环境；

### 配置pom文件

<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/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.wez</groupId>

<artifactId>myshiro</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>myshiro</name>

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

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<!-- shiro -->

<dependency>

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

<artifactId>shiro-all</artifactId>

<version>1.3.2</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.1.3</version>

</dependency>

<!-- ehcache缓存 -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-ehcache</artifactId>

<version>5.2.0.Final</version>

</dependency>

<!-- log4j日志 -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.5</version>

</dependency>

<!-- servlet-api -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

<!-- jstl -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- Spring上下文 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.8.RELEASE</version>

</dependency>

<!-- Spring jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.3.8.RELEASE</version>

</dependency>

<!-- Spring事务 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>4.3.8.RELEASE</version>

</dependency>

<!-- Spring web -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.8.RELEASE</version>

</dependency>

<!-- Spring集成测试 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>4.3.8.RELEASE</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.4</version>

</dependency>

<!-- SpringMVC的jar包 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.8.RELEASE</version>

</dependency>

</dependencies>

<build>

<finalName>myshiro</finalName>

</build>

</project>

### 配置ShiroFilter

在web.xml中配置shiro fileter；

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<web-app version=*"2.5"* xmlns=*"http://java.sun.com/xml/ns/javaee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee*

*http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>

<display-name>Archetype Created Web Application</display-name>

<!-- 配置SpringMVC核心控制器 -->

<servlet>

<servlet-name>spring</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- 配置SpringMVC的配置文件路径 -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mvc.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>spring</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<!-- 配置Spring容器 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-resource.xml</param-value>

</context-param>

<!-- 配置Spring容器的监听器 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 配置Spring编码集 -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

<init-param>

<param-name>forceRequestEncoding</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>forceResponseEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置shiro filter -->

<filter>

<filter-name>shiroFilter</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

<init-param>

<param-name>targetFilterLifecycle</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>shiroFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

</web-app>

### 配置Spring容器

配置spring容器配置文件；

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans

xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-4.2.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd"*>

<!-- 开启控制反转注解的支持 -->

<context:component-scan base-package=*"com.wez.sims"*>

<context:exclude-filter type=*"annotation"* expression=*"org.springframework.stereotype.Controller"*/>

</context:component-scan>

<!-- 1 配置SecurityManager(安全管理器) -->

<bean id=*"securityManager"* class=*"org.apache.shiro.web.mgt.DefaultWebSecurityManager"*>

<!-- 缓存管理器 -->

<property name=*"cacheManager"* ref=*"cacheManager"*/>

<!-- 域 -->

<property name=*"realm"* ref=*"jdbcRealm"*/>

</bean>

<!--

2 配置CacheManager(缓存管理器)

2.1 引入ehcache的jar包和配置文件

-->

<bean id=*"cacheManager"* class=*"org.apache.shiro.cache.ehcache.EhCacheManager"*>

<property name=*"cacheManagerConfigFile"* value=*"classpath:ehcache.xml"*/>

</bean>

<!--

3 配置Realm

3.1 直接实现了

-->

<bean id=*"jdbcRealm"* class=*"com.wez.shiro02.realm.ShiroRealm"*></bean>

<!-- 4 配置LifecycleBeanPostProcessor(生命周期bean的post管理器)，可以自动的调用spring IoC容器中shiro bean的生命周期方法； -->

<bean id=*"lifecycleBeanPostProcessor"* class=*"org.apache.shiro.spring.LifecycleBeanPostProcessor"*/>

<!-- 5 启用IoC容器中使用shiro注解，但必须配置了LifecycleBeanPostProcessor之后才可以使用； -->

<bean class=*"org.springframework.aop.framework.autoproxy.DefaultAdvisorAutoProxyCreator"*

depends-on=*"lifecycleBeanPostProcessor"*/>

<bean class=*"org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor"*>

<property name=*"securityManager"* ref=*"securityManager"*/>

</bean>

<!--

6 配置ShiroFilter

6.1 该bean的id必须和web.xml中配置的ShiroFilter的<filter-name>一致；

-->

<bean id=*"shiroFilter"* class=*"org.apache.shiro.spring.web.ShiroFilterFactoryBean"*>

<property name=*"securityManager"* ref=*"securityManager"*/>

<!-- 登陆页面 -->

<property name=*"loginUrl"* value=*"/login.jsp"*/>

<!-- 登陆成功后跳转到的页面 -->

<property name=*"successUrl"* value=*"/loginSuccess.jsp"*/>

<!-- 没有权限跳转到的页面 -->

<property name=*"unauthorizedUrl"* value=*"/unauthorized.jsp"*/>

<!--

配置页面的访问权限

a）anon：表示可以匿名访问；

b）authc：表示认证（即登陆）后才可以访问；

-->

<property name=*"filterChainDefinitions"*>

<value>

/login.jsp = anon

# 其他请求需要认证

/\*\* = authc

</value>

</property>

</bean>

</beans>