# Shiro权限验证

## 权限管理

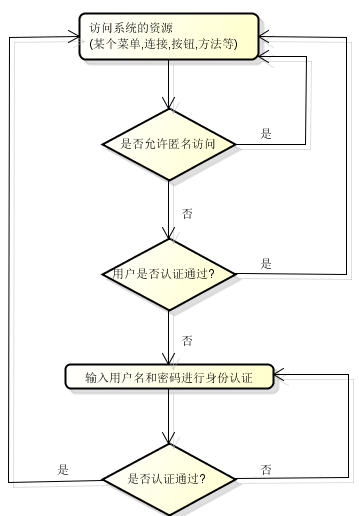
### 权限管理介绍

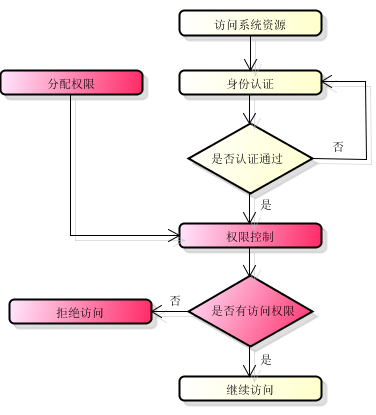
基本上涉及到用户参与的系统都要进行权限管理，权限管理属于系统安全的范畴，权限管理能实现对用户访问系统的控制，按照安全规则或者安全策略限制用户操作，只允许用户访问被授权的资源。

权限管理包括用户身份认证和授权两部分，简称认证授权。

### 身份认证

身份认证,就是判断一个用户是否为合法用户的处理过程。最常用的简单身份认证方式是系统通过核对用户输入的用户名和口令,看其是否与系统中存储的该用户的用户名和口令一致,来判断用户身份是否正确。





上边的流程图中需 要理解以下关键对象：

Subject：主体(当前登录用户)

访问系统的用户，主体可以是用户、程序等，进行认证的都称为主体；

Principal：身份信息(username)

是主体（subject）进行身份认证的标识，标识必须具有唯一性，如用户名、手机号、邮箱地址等，一个主体可以有多个身份，但是必须有一个主身份（Primary Principal）。

credential：凭证信息(password)

是只有主体自己知道的安全信息，如密码、证书等。

## Shiro介绍

### 什么是shiro

Apache Shiro 是Java 的一个安全框架。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。

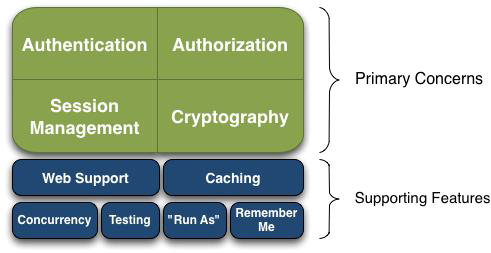
### 为什么要学shiro

既然shiro将安全认证相关的功能抽取出来组成一个框架，使用shiro就可以非常快速的完成认证、授权等功能的开发，降低系统成本。

shiro使用广泛，shiro可以运行在web应用，非web应用，集群分布式应用中越来越多的用户开始使用shiro。

java领域中spring security(原名Acegi)也是一个开源的权限管理框架，但是spring security依赖spring运行，而shiro就相对独立，最主要是因为shiro使用简单、灵活，所以现在越来越多的用户选择shiro。

### 基本功能



#### 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；

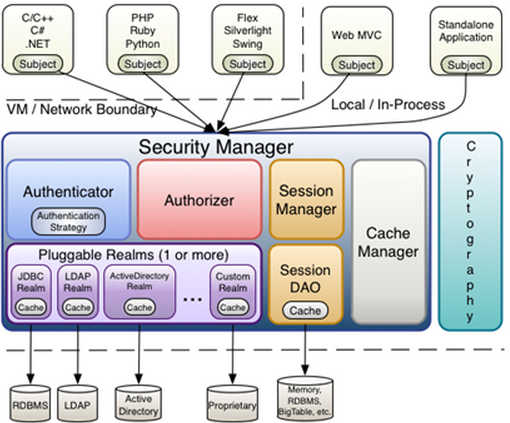
**Remember Me：**记住我，这个是非常常见的功能，即一次登录后，下次再来的话不用登录

了。

**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。**

### Shiro架构



#### Subject

Subject即主体，外部应用与subject进行交互，subject记录了当前操作用户，将用户的概念理解为当前操作的主体，可能是一个通过浏览器请求的用户，也可能是一个运行的程序。 Subject在shiro中是一个接口，接口中定义了很多认证授相关的方法，外部程序通过subject进行认证授，而subject是通过SecurityManager安全管理器进行认证授权

#### SecurityManager

SecurityManager即安全管理器，对全部的subject进行安全管理，它是shiro的核心，负责对所有的subject进行安全管理。通过SecurityManager可以完成subject的认证、授权等，实质上SecurityManager是通过Authenticator进行认证，通过Authorizer进行授权，通过SessionManager进行会话管理等。

SecurityManager是一个接口，继承了Authenticator, Authorizer, SessionManager这三个接口。

#### Authenticator

Authenticator即认证器，对用户身份进行认证，Authenticator是一个接口，shiro提供ModularRealmAuthenticator实现类，通过ModularRealmAuthenticator基本上可以满足大多数需求，也可以自定义认证器。

#### Authorizer

Authorizer即授权器，用户通过认证器认证通过，在访问功能时需要通过授权器判断用户是否有此功能的操作权限。

#### realm

Realm即领域，相当于datasource数据源，securityManager进行安全认证需要通过Realm获取用户权限数据，比如：如果用户身份数据在数据库那么realm就需要从数据库获取用户身份信息。

注意：不要把realm理解成只是从数据源取数据，在realm中还有认证授权校验的相关的代码。

#### sessionManager

sessionManager即会话管理，shiro框架定义了一套会话管理，它不依赖web容器的session，所以shiro可以使用在非web应用上，也可以将分布式应用的会话集中在一点管理，此特性可使它实现单点登录。

#### SessionDAO

SessionDAO即会话dao，是对session会话操作的一套接口，比如要将session存储到数据库，可以通过jdbc将会话存储到数据库。

#### CacheManager

CacheManager即缓存管理，将用户权限数据存储在缓存，这样可以提高性能。

#### Cryptography

Cryptography即密码管理，shiro提供了一套加密/解密的组件，方便开发。比如提供常用的散列、加/解密等功能。

## 身份认证

### 基本概念

#### 身份验证

即在应用中谁能证明他就是他本人。一般提供如他们的身份ID 一些标识信息来

表明他就是他本人，如提供身份证，用户名/密码来证明。

在 shiro 中，用户需要提供principals （身份）和credentials（证明）给shiro，从而应用能

验证用户身份：

#### principals

身份，即主体的标识属性，可以是任何东西，如用户名、邮箱等，唯一即可。

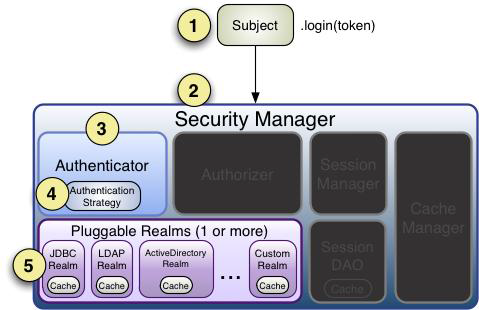
一个主体可以有多个principals，但只有一个Primary principals，一般是用户名/密码/手机号。

#### credentials

证明/凭证，即只有主体知道的安全值，如密码/数字证书等。

最常见的principals和credentials组合就是用户名/密码了。

### 认证流程



### 入门案例(登录退出)

#### 添加jar依赖

|  |
| --- |
| <!-- shiro核心包 -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-core</artifactId>  <version>1.2.4</version>  </dependency>  <!-- 日志 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.7.12</version>  </dependency> |

#### shiro配置文件

|  |
| --- |
| 在资源文件夹下创建shiro.ini配置文件（配置文件名称不固定） |
| [users]  admin=123456  test=123 |

#### 代码实现

|  |
| --- |
|  |

#### 常见异常

##### 用户名错误

|  |
| --- |
|  |

##### 密码错误

|  |
| --- |
|  |

## ini配置

### 概述

ini 配置文件类似于 Java 中的 properties（key=value），不过提供了将 key/value 分类的特性， key 是每个部分不重复即可，而不是整个配置文件。如下是 INI 配置分类：

|  |
| --- |
| [main]  #提供了对根对象 securityManager 及其依赖的配置securityManager=org.apache.shiro.mgt.DefaultSecurityManager  …………  securityManager.realms=$jdbcRealm    [users]  #提供了对用户/密码及其角色的配置，用户名=密码，角色 1，角色 2 username=password,role1,role2    [roles]  #提供了角色及权限之间关系的配置，角色=权限 1，权限 2 role1=permission1,permission2    [urls]  #用于 web，提供了对 web url 拦截相关的配置，url=拦截器[参数]，拦截器  /index.html = anon  /admin/\*\* = authc, roles[admin], perms["permission1"] |

### [main]部分

|  |
| --- |
| 提供了对根对象 securityManager 及其依赖的配置 |

### [users]部分

|  |
| --- |
| 配置用户名/密码及其角色，格式：“用户名=密码，角色 1，角色 2”，角色部分可省略。如： |

### [roles]部分

|  |
| --- |
| 配置角色及权限之间的关系，格式：“角色=权限 1，权限 2”；如： |

### [urls]部分

|  |
| --- |
| 配置 url 及相应的拦截器之间的关系，格式：“url=拦截器[参数]，拦截器[参数]，如： |

## Realm

### 概述

|  |
| --- |
| 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。如我们之前的ini配置方式 将使用org.apache.shiro.realm.text.IniRealm。  Shiro默认使用自带的IniRealm，IniRealm从ini配置文件中读取用户的信息，大部分情况下需要从系统的数据库中读取用户信息，所以需要自定义realm。 |

### JDBCRealm

#### 创建数据库

|  |
| --- |
| CREATE DATABASE db\_shiro;  USE db\_shiro;  CREATE TABLE users(  id INT PRIMARY KEY AUTO\_INCREMENT,  username VARCHAR(100),  PASSWORD VARCHAR(100)  );  INSERT INTO users (username,PASSWORD) VALUES('admin','123456'),('test','123'); |

#### 数据源c3p0版本

##### 添加相关依赖

|  |
| --- |
| <!-- shiro核心 -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-core</artifactId>  <version>1.2.4</version>  </dependency>  <!-- 日志 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.7.12</version>  </dependency>  <!-- c3p0数据源 -->  <dependency>  <groupId>c3p0</groupId>  <artifactId>c3p0</artifactId>  <version>0.9.1.2</version>  </dependency>  <!-- 日志信息 -->  <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.2</version>  </dependency>  <!-- mysql驱动 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.15</version>  </dependency> |

##### Shiro配置文件

|  |
| --- |
| [main]  #使用jdbcRealm  jdbcRealm=org.apache.shiro.realm.jdbc.JdbcRealm  #c3p0数据源  dataSource=com.mchange.v2.c3p0.ComboPooledDataSource  #数据库驱动  dataSource.driverClass=com.mysql.cj.jdbc.Driver  #连接字符串  dataSource.jdbcUrl=jdbc:mysql://localhost:3306/db\_shiro?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC  #数据库用户名  dataSource.user=root  #数据库密码  dataSource.password=root  #引用、注入数据源  jdbcRealm.dataSource=$dataSource  #引用realm  securityManager.realms=$jdbcRealm |

#### 数据源Druid版本

##### 添加相关依赖

|  |
| --- |
| <!-- shiro核心 -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-core</artifactId>  <version>1.2.4</version>  </dependency>  <!-- 日志 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.7.12</version>  </dependency>  <!-- 阿里巴巴数据源 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.1.17</version>  </dependency>  <!-- 日志信息 -->  <dependency>  <groupId>commons-logging</groupId>  <artifactId>commons-logging</artifactId>  <version>1.2</version>  </dependency>  <!-- mysql驱动 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.15</version>  </dependency> |

##### Shiro配置文件

|  |
| --- |
| [main]  #使用jdbcRealm  jdbcRealm=org.apache.shiro.realm.jdbc.JdbcRealm  #alibaba数据源  dataSource=com.alibaba.druid.pool.DruidDataSource  #数据库驱动  dataSource.driverClassName=com.mysql.cj.jdbc.Driver  #连接字符串  dataSource.url=jdbc:mysql://localhost:3306/db\_shiro?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC  #数据库用户名  dataSource.username=root  #数据库密码  dataSource.password=root  #引用、注入数据源  jdbcRealm.dataSource=$dataSource  #引用realm  securityManager.realms=$jdbcRealm |

#### 代码实现

|  |
| --- |
| **public** **static** **void** main(String[] args) {  //1.读取配置文件，初始化SecurityManager工厂  Factory<SecurityManager> factory = **new** IniSecurityManagerFactory("classpath:jdbc\_realm.ini");  //2.获取securitymanager实例  SecurityManager securityManager = factory.getInstance();  //3.把SecurityManager实例绑定到SecurityUtils  SecurityUtils.*setSecurityManager*(securityManager);  //4.得到当前登录的用户  Subject currentUser =SecurityUtils.*getSubject*();  //5.创建Token令牌，传入用户名和密码  UsernamePasswordToken token = **new** UsernamePasswordToken("admin","123456");  **try** {  //6.身份验证  currentUser.login(token);  System.***out***.println("身份验证成功！！！");  } **catch** (AuthenticationException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  System.***out***.println("身份验证失败！！！");  }  //7.退出  currentUser.logout();  } |

## 权限验证（授权）

### 验证角色

#### 常用方法

|  |
| --- |
| **boolean** hasRole(String roleIdentifier);  **boolean**[] hasRoles(List<String> roleIdentifiers);  **boolean** hasAllRoles(Collection<String> roleIdentifiers);  **void** checkRole(String roleIdentifier) **throws** AuthorizationException;  **void**checkRoles(Collection<String> roleIdentifiers) **throws** AuthorizationException;  **void** checkRoles(String... roleIdentifiers) **throws** AuthorizationException; |

#### shiro-role.ini配置文件

|  |
| --- |
| #定义用户信息  [users]  #用户名=密码，角色1，角色2，角色3  admin=123456,role1,role2,role3  #用户名=密码，角色1  test=123,role1 |

#### 测试类

|  |
| --- |
| @Test  **public** **void** test1() {  //得到当前用户  Subject currentUser = ShiroUtil.*login*("classpath:shiro\_role.ini", "admin", "123456");  //Subject currentUser = ShiroUtil.login("classpath:shiro\_role.ini", "test", "123");  System.***out***.println(currentUser.hasRole("role2")?"有这个角色":"没有这个角色");  //判断多个角色  **boolean** [] results = currentUser.hasRoles(Arrays.*asList*("role1","role2","role3"));  System.***out***.println(results[0]?"有role1这个角色":"没有role1这个角色");  System.***out***.println(results[1]?"有role2这个角色":"没有role2这个角色");  System.***out***.println(results[2]?"有role3这个角色":"没有role3这个角色");  //退出  currentUser.logout();  }      @Test  **public** **void** testCheckRole() {  //Subject currentUser = ShiroUtil.login("classpath:shiro\_role.ini", "admin", "123456");  Subject currentUser = ShiroUtil.*login*("classpath:shiro\_role.ini", "test", "123");  currentUser.checkRole("role1");  currentUser.checkRoles(Arrays.*asList*("role1","role2"));  currentUser.checkRoles("role1","role2","role3");  currentUser.logout();  } |

### 验证权限

#### 常用方法

|  |
| --- |
| **boolean** isPermitted(String permission);  **boolean**[] isPermitted(String... permissions); |

#### 配置文件

|  |
| --- |
| [users] ‘’  admin=123456,role1,role2  test=123,role1  #角色  [roles]  #角色名称=具体权限  role1=user:select  role2=user:add,user:update,user:delete |

#### 测试类

|  |
| --- |
| @Test  **public** **void** test1() {  // 得到当前用户  Subject currentUser = ShiroUtil.*login*("classpath:shiro\_permission.ini", "admin", "123456");  System.***out***.println(currentUser.isPermitted("user:select")?"有user:select这个权限":"没有user:select这个权限");  **boolean** [] results = currentUser.isPermitted("user:select","user:update","user:delete");  System.***out***.println(results[0]?"有user:select这个权限":"没有user:select这个权限");  System.***out***.println(results[1]?"有user:update这个权限":"没有user:update这个权限");  System.***out***.println(results[2]?"有user:delete这个权限":"没有user:delete这个权限");  // 退出  currentUser.logout();  } |

## 集成web