**Shiro**

# Shiro简介

Apache Shiro 是 Java 的一个安全（权限）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 可以完成：认证、授权、加密、会话管理、与Web 集成、缓存 等。

* [下载：http://shiro.apache.org/](http://shiro.apache.org/)

## 功能简介

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 户是否能进行什么操作，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，也可以是 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 架构(Shiro外部来看)

从外部来看Shiro ，即从应用程序角度的来观察如何使用 Shiro 完成

工作：

**Subject：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。**Subject 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；**与 Subject 的所有交互都会委托给 SecurityManager**； **Subject 其实是一个门面，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**；且其管理着所有 Subject；可以看出它是 **Shiro 的核心**，它**负责与 Shiro 的其他组件进行交互**，它相当于 SpringMVC 中 DispatcherServlet 的角色

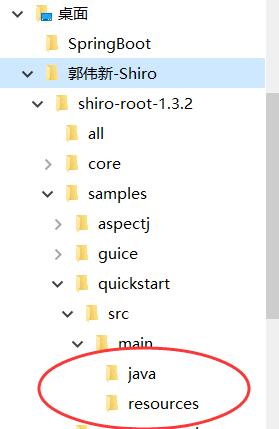
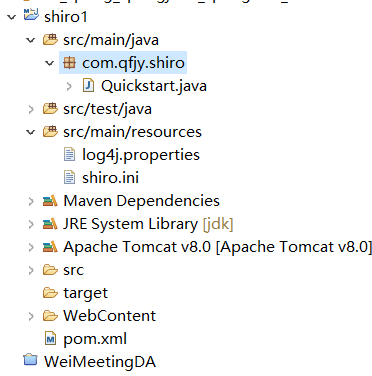
**Realm：**Shiro **从 Realm 获取安全数据（如用户、角色、权限），**就是说 SecurityManager 要验证用户身份，那么它需要从 Realm 获取相应的用户 进行比较以确定用户身份是否合法；也需要从 Realm 得到用户相应的角色/ 权限进行验证用户是否能进行操作；可以把 Realm 看成 DataSource

## Shiro 架构(Shiro内部来看)

# Shiro使用

## 第一个Shiro 案例

### 创建maven project项目

将quickstart中的 java代码、resources代码复制到工程中

### 导入相关需要jar/pom.xml

**加入如下 jar 包：**

shiro-all-1.3.2.jar log4j-1.2.15.jar slf4j-api-1.6.1.jar slf4j-log4j12-1.6.1.jar

**maven pom.xml：**

|  |
| --- |
| <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <shiro.version>1.3.2</shiro.version>  </properties>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <!-- shiro需要的jar -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-core</artifactId>  <version>${shiro.version}</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-web</artifactId>  <version>${shiro.version}</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-cas</artifactId>  <version>${shiro.version}</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-spring</artifactId>  <version>${shiro.version}</version>  </dependency>  <!-- shiro需要的jar -->  <!-- log4j 解决方案 -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.15</version>  <exclusions>  <exclusion>  <groupId>com.sun.jmx</groupId>  <artifactId>jmxri</artifactId>  </exclusion>  <exclusion>  <groupId>com.sun.jdmk</groupId>  <artifactId>jmxtools</artifactId>  </exclusion>  <exclusion>  <groupId>javax.jms</groupId>  <artifactId>jms</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.6.1</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.6.1</version>  </dependency>  <!-- shiro需要的jar -->  </dependencies> |

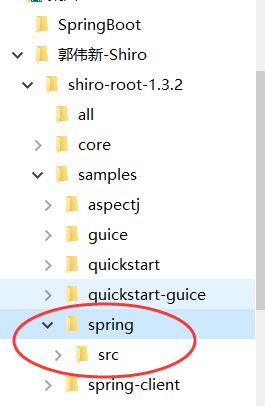
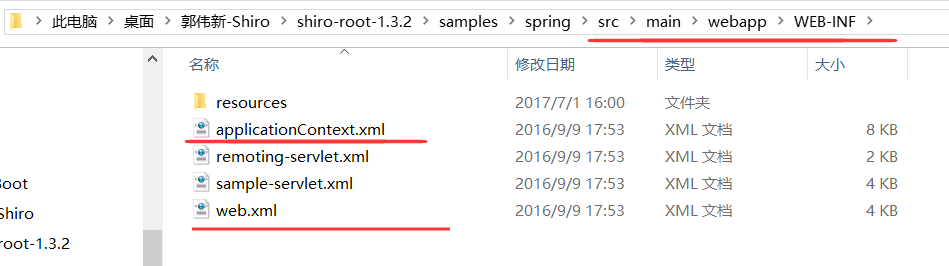
# Shiro整合Spring

## 加入 Spring 和 Shiro 的 jar 包

|  |
| --- |
| <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <!-- 自定义版本号 -->  <spring.version>4.3.8.RELEASE</spring.version>  <shiro.version>1.3.2</shiro.version>  </properties>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <!-- Spring jar -->  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aop</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjrt</artifactId>  <version>1.8.0</version>  </dependency>  <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjweaver</artifactId>  <version>1.8.0</version>  </dependency>  <!-- mybatis -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.4.2</version>  </dependency>  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.3.0</version>  </dependency>  <!-- Mysql -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.42</version>  </dependency>  <!-- druid连接池 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.1.10</version>  </dependency>  <!-- jackson -->  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.8.8</version>  </dependency>  <!--JSTL标签 -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>taglibs</groupId>  <artifactId>standard</artifactId>  <version>1.1.2</version>  </dependency>  <!-- shiro需要的jar <dependency> <groupId>org.apache.shiro</groupId> <artifactId>shiro-all</artifactId>  <version>1.3.2</version> </dependency> -->  <!-- shiro需要的jar -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-core</artifactId>  <version>${shiro.version}</version>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-web</artifactId>  <version>${shiro.version}</version>  </dependency>  <!-- shiro spring -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-spring</artifactId>  <version>${shiro.version}</version>  </dependency>  <!-- shiro enchche 缓存 -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-ehcache</artifactId>  <version>${shiro.version}</version>  </dependency>  <!-- shiro需要的jar -->  <!-- log4j 解决方案 -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.15</version>  <exclusions>  <exclusion>  <groupId>com.sun.jmx</groupId>  <artifactId>jmxri</artifactId>  </exclusion>  <exclusion>  <groupId>com.sun.jdmk</groupId>  <artifactId>jmxtools</artifactId>  </exclusion>  <exclusion>  <groupId>javax.jms</groupId>  <artifactId>jms</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.6.1</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.6.1</version>  </dependency>  <!-- shiro需要的jar -->  <!--Encache 缓存 -->  <dependency>  <groupId>net.sf.ehcache</groupId>  <artifactId>ehcache</artifactId>  <version>2.10.2</version>  </dependency>  <!--Encache 缓存 -->      <!-- lombok.jar -->  <dependency>  <groupId>org.projectlombok</groupId>  <artifactId>lombok</artifactId>  <version>1.16.18</version>  </dependency>      </dependencies> |

## 与Web集成

参照：1.3.2\shiro-root-1.3.2-source- release\shiro-root-1.3.2\samples\spring **配置 web.xml 文件和 Spring 的配置文件**

##### 集成第一步：ShiroFilter (配置web.xml)

Shiro 提供了与 Web 集成的支持，其通过一个 **ShiroFilter** 入口来拦截需要安全控制的URL，然后 进行相应的控制

ShiroFilter 类似于如 Strut2/SpringMVC 这种 web 框架的前端控制器，是**安全控制的入口点**，其 负责读取配置（如ini 配置文件），然后**判断URL 是否需要登录/权限等工作**。

**DelegatingFilterProxy 作用是自动到 Spring 容器查找名 字为 shiroFilter（filter-name）的 bean 并把所有 Filter 的操作委托给它。**

##### 集成第二步：spring-shiro.xml(applicationContext.xml)



##### Ehcache

EhCache 是一个纯Java的进程内缓存框架，具有快速、精干等特点。它是Hibernate中的默认缓存框架

**Ehcache**  
如果你的项目使用maven管理，添加以下依赖到你的*pom.xml*中

<!--Ehcache 缓存 -->

<dependency>

<groupId>net.sf.ehcache</groupId>

<artifactId>ehcache</artifactId>

<version>2.10.2</version>

</dependency>

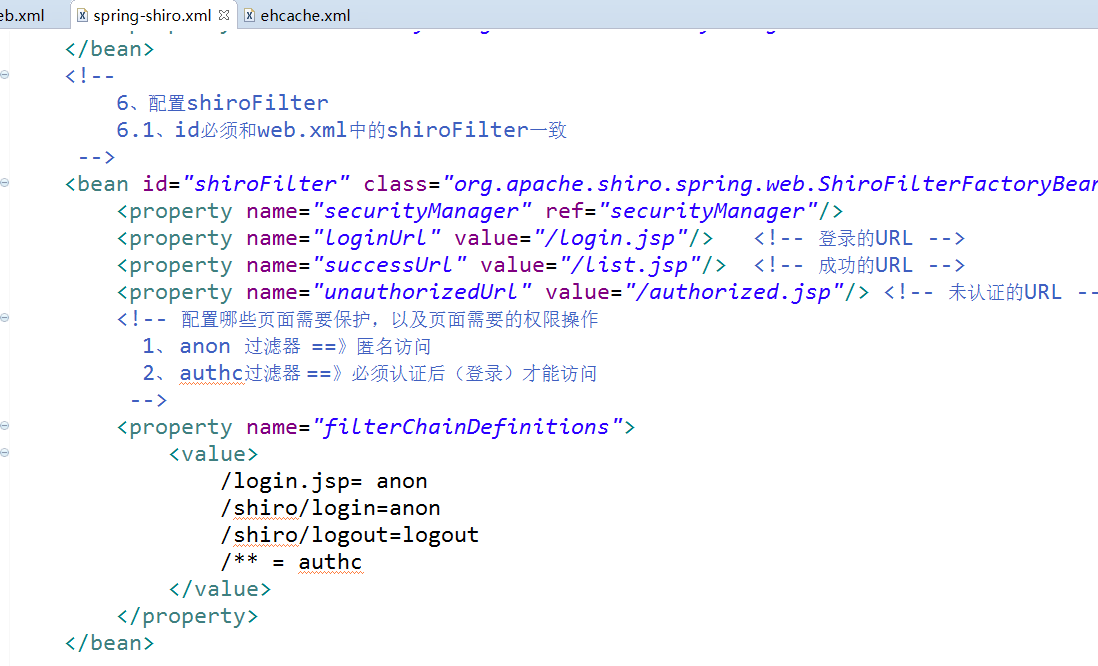
<!--Ehcache 缓存 -->

**在classpath下添加*ehcache.xml***

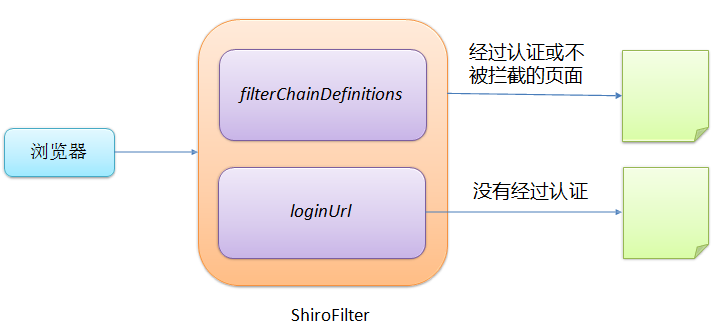
参考该配置信息：<https://blog.csdn.net/chinabinggo/article/details/100491870>

# ShiroFilter 案例

Shiro帮助我们进行拦截，未认证的访问页面，都需要跳转到登录页面中



## ShiroFilter 的工作原理



**细节描述：**

**[url] 部分的配置**，其格式是： “**url=*拦截器*[*参数*]**，拦截 器[参数]”；

如果当前请求的 url 匹配 [urls] 部分的某个 url 模式，将会

执行其配置的拦截器。

**anon（anonymous）** 拦截器表示匿名访问（即不需要登 录即可访问）

**authc （authentication）**拦截器表示需要身份认证通过后 才能访问

## shiro中默认的过滤器

## URL 匹配模式

**url 模式使用 Ant 风格模式**

Ant 路径通配符支持 ?、\*、\*\*，注意通配符匹配不包括目录分隔符“/”：

**?：匹配一个字符**，如 /admin? 将匹配 /admin1，但不

匹配 /admin 或 /admin/；

**\*：匹配零个或多个字符串**，如 /admin 将匹配 /admin、

/admin123，但不匹配 /admin/1；

**\*\*：匹配路径中的零个或多个路径**，如 /admin/\*\* 将匹 配 /admin/a 或 /admin/a/b

## URL 匹配顺序

**URL 权限采取第一次匹配优先的方式**，即从头开始

使用第一个匹配的 url 模式对应的拦截器链。

• 如：

* /bb/\*\*=filter1
* /bb/aa=filter2
* /\*\*=filter3

– 如果请求的url是“/bb/aa”，因为按照声明顺序进行匹 配，那么将使用 filter1 进行拦截。

# Shiro认证

**Shiro 架构(Shiro外部来看)**

从外部来看Shiro ，即从应用程序角度的来观察如何使用 Shiro 完成

工作：

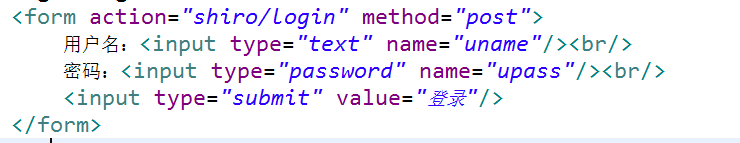
## Realm定义

**Realm：**Shiro 从 Realm 获取安全数据（如用户、角色、 权限），即 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

* Realm接口如下：

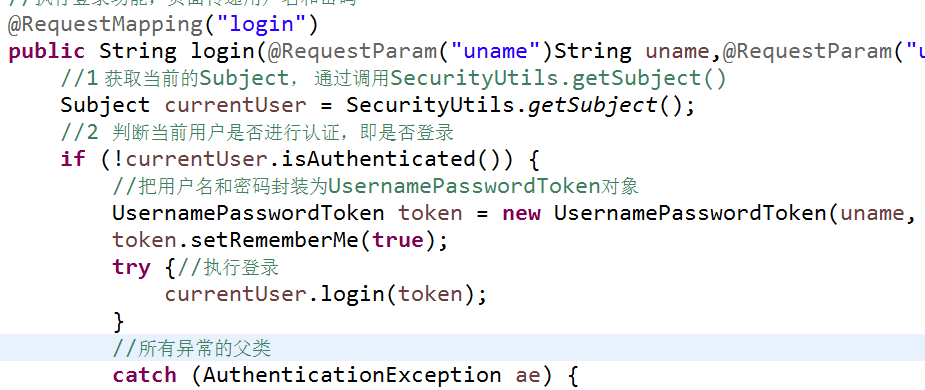
## 身份验证基本流程

##### 1、收集用户身份/凭证，即如用户名/密码 login.jsp页面：



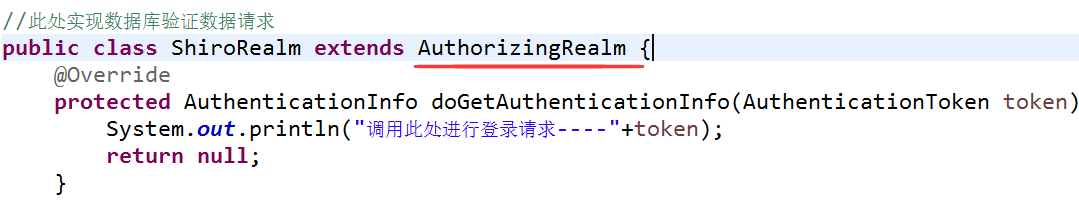
##### 2、调用 Subject.login 进行登录

如果失败将得到相应 的 AuthenticationException 异常，根据异常提示用户 错误信息；否则登录成功



##### 3、创建自定义的 Realm 类

**继承 org.apache.shiro.realm.AuthorizingRealm 类，实现 doGetAuthenticationInfo() 方法**



##### 4、详细身份验证

**身份验证**：一般需要提供如身份 ID 等一些标识信息来表明登录者的身

份，如提供 email，**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**，可以是任何属性，**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/邮箱/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

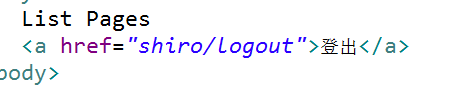


## 登出

* 默认Shiro存在缓存，第一次如果成功，下次即使失败也能成功。
* 解决方案：在spring-shiro.xml中



* 在jsp页面中加入：



# Shiro加密

登录的密码是不会采用明文的方式进行存储的，都经过严格的加密方式来存储密码。

## 加密概述

加密技术是对信息进行编码和解码的技术，编码是把原来可读信息（又称明文）译成代码形式（又称密文），其逆过程就是解码（解密）。加密技术的要点是加密算法，加密算法可以分为对称加密、非对称加密和不可逆加密三类算法。

加密常用概念分为：**对称与非对称和不可逆**

**对称加密:**

数据加密和解密采用的都是同一个密钥，因而其安全性依赖于所持有密钥的安全性。  
对称加密算法的主要优点是加密和解密速度快，加密强度高，且算法公开.

 常用算法名称：DES （Data Encryption Standard）、 AES（Advanced Encryption Standard）

**非对称加密:**

使用两把完全不同但又是完全匹配的一对钥匙（即一把公开密钥或加密密钥和专用密钥或解密密钥）—公钥和私钥。在使用不对称加密算法加密文件时，只有使用匹配的一对公钥和私钥，才能完成对明文的加密和解密过程。加密明文时采用公钥加密，解密密文时使用私钥才能完成

**常用算法名称： SHS  RSA**

**不可逆加密:**

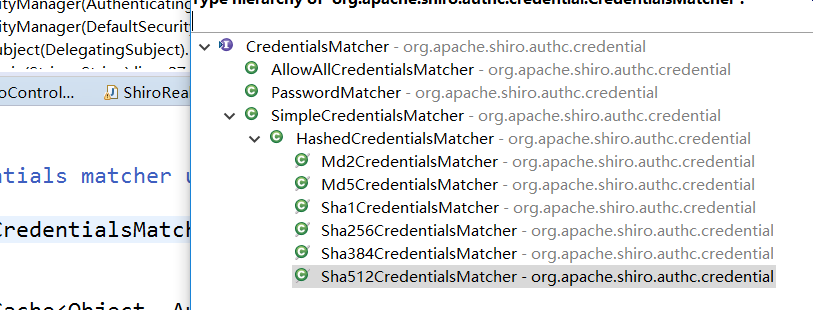
　不可逆加密算法的特征是加密过程中不需要使用密钥，输入明文后由系统直接经过加密算法处理成密文，这种加密后的数据是无法被解密的，只有重新输入明文，并再次经过同样不可逆的加密算法处理，得到相同的加密密文并被系统重新识别后，才能真正解密。

不可逆加密算法不存在密钥保管和分发问题，非常适合在分布式网络系统上使用，但因加密计算复杂，工作量相当繁重，通常只在数据量有限的情形下使用，如广泛应用在计算机系统中的口令加密，利用的就是不可逆加密算法。

**常用算法名称：MD5(32bit) SHA256(64bit) SHA512（生成128bit）**

## 断点进入，通过查看，可以知道是通过

 AuthenticatingRealm的 credentialsMatcher 属性来进行的密码的比对



## Shrio密码比对

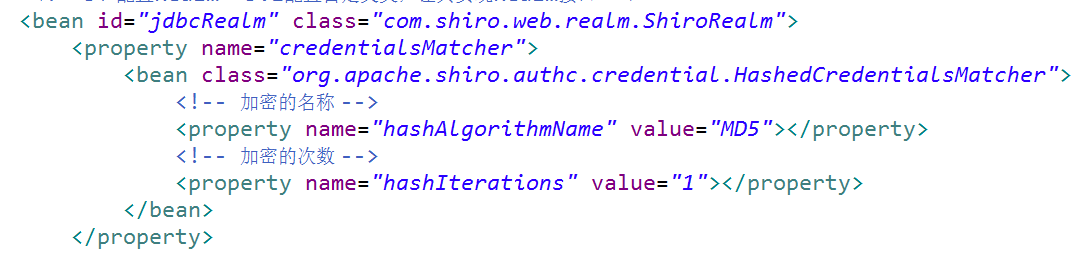
* **密码的比对由Shiro完成**
* **1、前台传递过来的进行MD5加密。**
* **2、查询出来的进行MD5加密。**

****

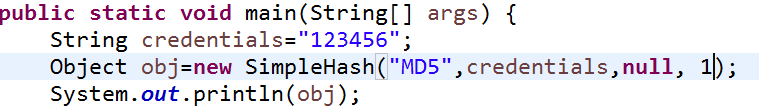
****

**（传递的进行MD5加密）解决方案：**

替换当前Realm的 AuthenticatingRealm的 credentialsMatcher 属性,改为MD5 CredentialsMatcher



**查询出来的进行MD5加密（才能一一匹配）**



## Shiro盐值加密

加盐加密是一种对系统登录口令的加密方式，它实现的方式是将每一个口令同一个叫做”盐“（salt）的n位随机数相关联

