安全框架 shiro

概述：

简而言之，Apache Shiro 是一个强大灵活的开源安全框架，可以完全处理身份验证、授权、加密和会话管理。

三大核心组件

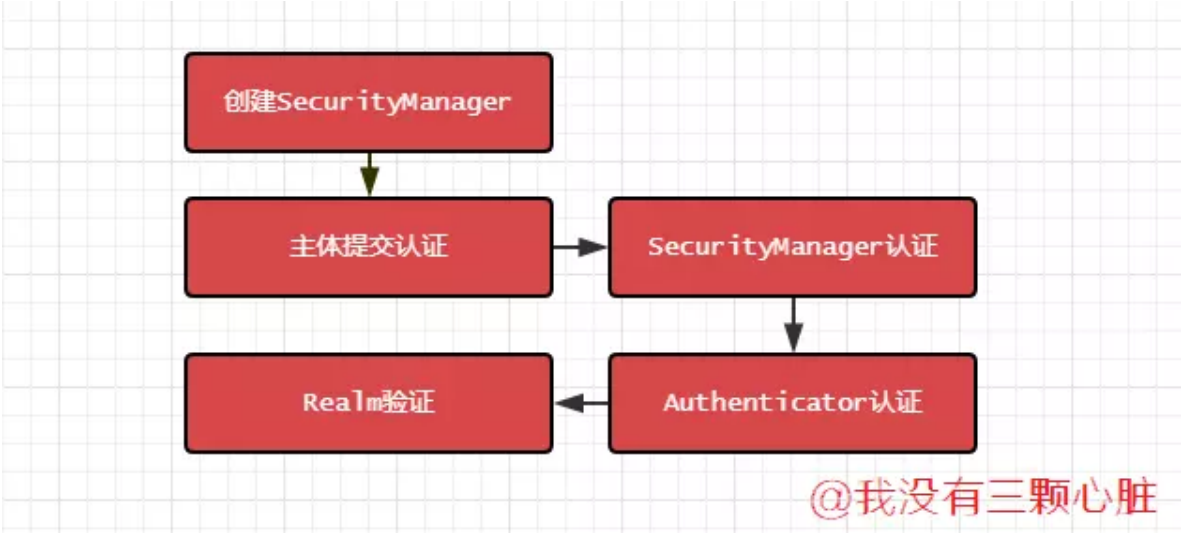
shiro 架构包含三个主要的理念：Subject,SecurityManager和 Realm

Subject：当前用户，Subject 可以是一个人，但也可以是第三方服务、守护进程帐户、时钟守护任务或者其它–当前和软件交互的任何事件。

SecurityManager：管理所有Subject，SecurityManager 是 Shiro 架构的核心，配合内部安全组件共同组成安全伞。

Realms：用于进行权限信息的验证，我们自己实现。Realm 本质上是一个特定的安全 DAO：它封装与数据源连接的细节，得到Shiro 所需的相关的数据。在配置 Shiro 的时候，你必须指定至少一个Realm 来实现认证（authentication）和/或授权（authorization）。

shiro认证过程



1.首先调用 Subject.login(token) 进行登录，其会自动委托给 Security Manager，调用之前必须通过 SecurityUtils.setSecurityManager() 设置；

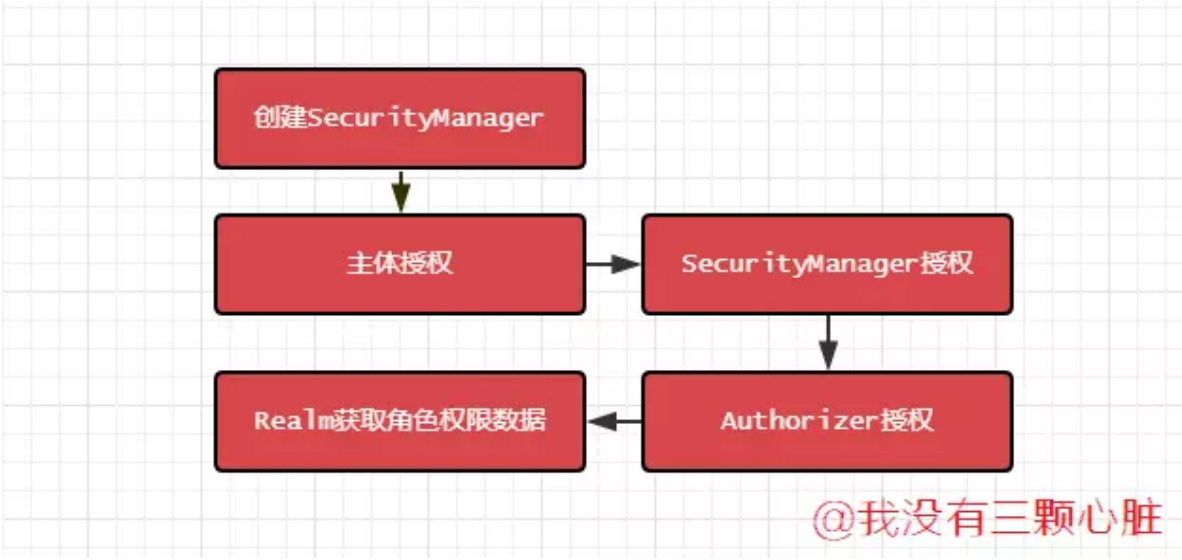
2.SecurityManager 负责真正的身份验证逻辑；它会委托给 Authenticator 进行身份验证；

3.Authenticator 才是真正的身份验证者，Shiro API 中核心的身份认证入口点，此处可以自定义插入自己的实现；

4.Authenticator 可能会委托给相应的 AuthenticationStrategy 进行多 Realm 身份验证，默认 ModularRealmAuthenticator 会调用 AuthenticationStrategy 进行多 Realm 身份验证；

5.Authenticator 会把相应的 token 传入 Realm，从 Realm 获取身份验证信息，如果没有返回 / 抛出异常表示身份验证失败了。此处可以配置多个 Realm，将按照相应的顺序及策略进行访问。

shiro授权过程



加盐 + 多次加密

既然相同的密码 md5 一样，那么我们就让我们的原始密码再**加一个随机数**，然后再进行 md5 加密，这个随机数就是我们说的**盐(salt)**，这样处理下来就能得到不同的 Md5 值，当然我们需要把这个随机数盐也保存进数据库中，以便我们进行验证。

另外我们可以通过**多次加密**的方法，即使黑客通过一定的技术手段拿到了我们的密码 md5 值，但它并不知道我们到底加密了多少次，所以这也使得破解工作变得艰难。

在 Shiro 框架中，对于这样的操作提供了简单的代码实现：

```
String password = "123456";
String salt = new
SecureRandomNumberGenerator().nextBytes().toString();
int times = 2;  // 加密次数：2
String alogrithmName = "md5";    // 加密算法

String encodePassword = new SimpleHash(alogrithmName, password, salt,
times).toString();

System.out.printf("原始密码是 %s ，盐是： %s，运算次数是： %d，运算出来的密
文是：%s ",password,salt,times,encodePassword);
```

输出：

```
原始密码是 123456 ，盐是： f5GQZsuWjnL9z585JjLrbQ==，运算次数是： 2，运算出
来的密文是：55fee80f73
```