**Shiro 源码解读**

# Shiro

# 什么是Shiro

Shiro是一个强大且易用的Java安全框架,执行身份验证、授权、密码和会话管理。使用Shiro的易于理解的API,您可以快速、轻松地获得任何应用程序,从最小的移动应用程序到最大的网络和企业应用程序。



## Shiro 框架主要功能

三个核心组件：**Subject, SecurityManager** 和 **Realms**.

**Subject**：即“当前操作用户”。但是，在Shiro中，Subject这一概念并不仅仅指人，也可以是第三方进程、后台帐户（Daemon Account）或其他类似事物。它仅仅意味着“当前跟软件交互的东西”。  
　　**Subject**代表了当前用户的安全操作，**SecurityManager**则管理所有用户的安全操作。  
　　**SecurityManager**：它是Shiro框架的核心，典型的Façade（门面模式）模式，Shiro通过SecurityManager来管理内部组件实例，并通过它来提供安全管理的各种服务。  
　　**Realm**： Realm充当了Shiro与应用安全数据间的“桥梁”或者“连接器”。也就是说，当对用户执行认证（登录）和授权（访问控制）验证时，Shiro会从应用配置的Realm中查找用户及其权限信息。  
　　从这个意义上讲，Realm实质上是一个安全相关的DAO：它封装了数据源的连接细节，并在需要时将相关数据提供给Shiro。当配置Shiro时，你必须至少指定一个Realm，用于认证和（或）授权。配置多个Realm是可以的，但是至少需要一个。  
　　Shiro内置了可以连接大量安全数据源（又名目录）的Realm，如LDAP、关系数据库（JDBC）、类似INI的文本配置资源以及属性文件等。如果缺省的Realm不能满足需求，你还可以插入代表自定义数据源的自己的Realm实现。

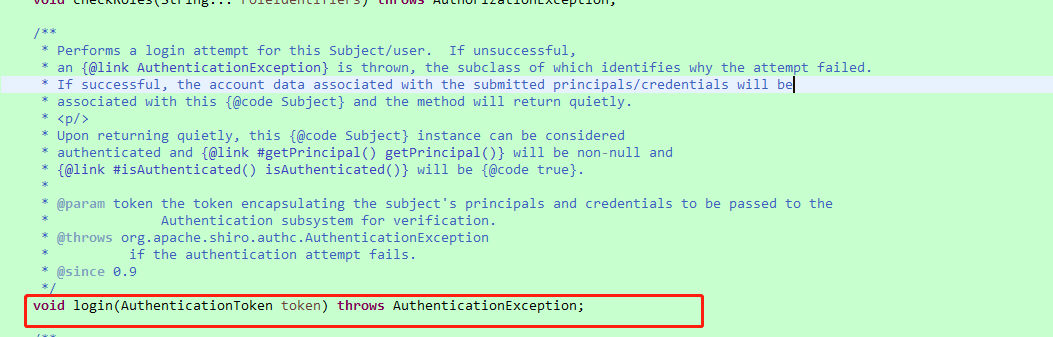
**什么是门面模式**

门面模式要求一个子系统的外部与其内部的通信必须通过一个统一的门面(Facade)对象进行。门面模式提供一个高层次的接口，使得子系统更易于使用。

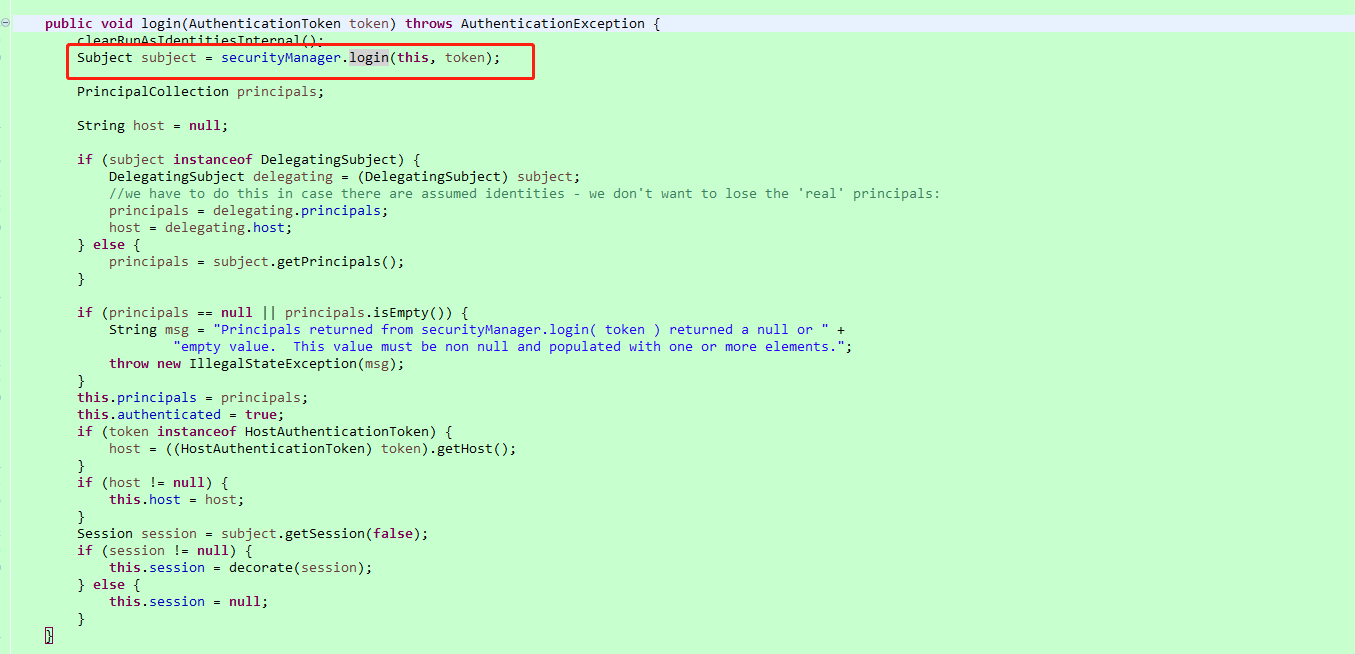
就如同医院的接待员一样，门面模式的门面类将客户端与子系统的内部复杂性分隔开，使得客户端只需要与门面对象打交道，而不需要与子系统内部的很多对象打交道。

# 获取Subject对象

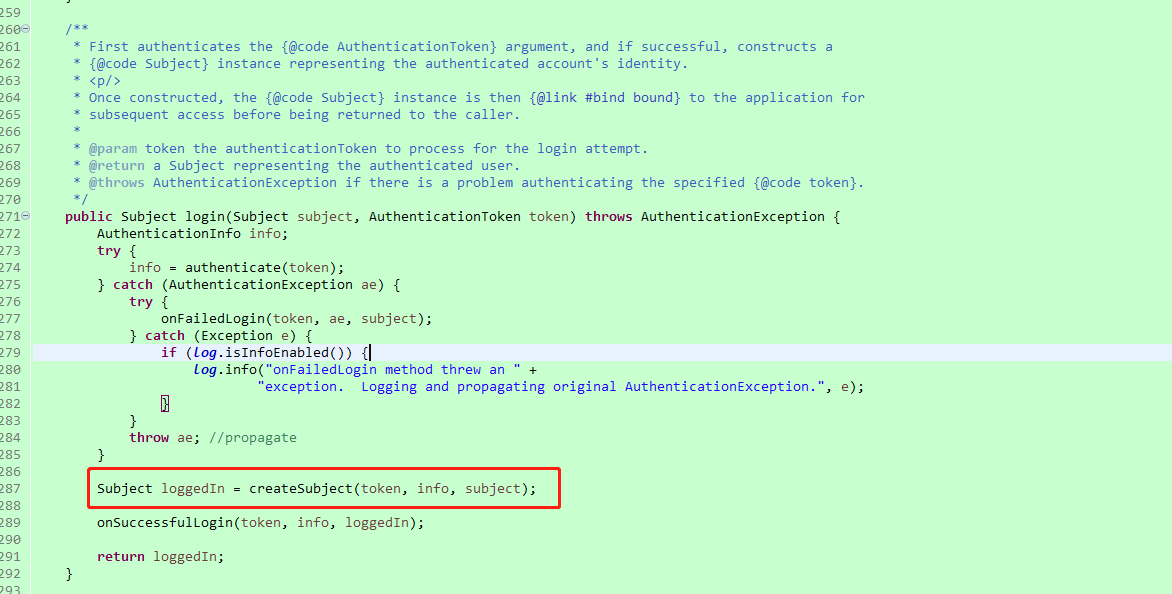
Shiro 入口函数 Subject Interface 的login 方法

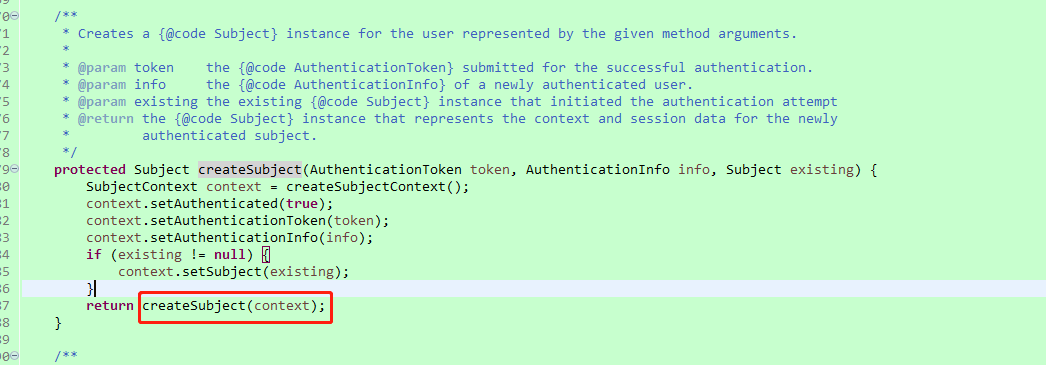


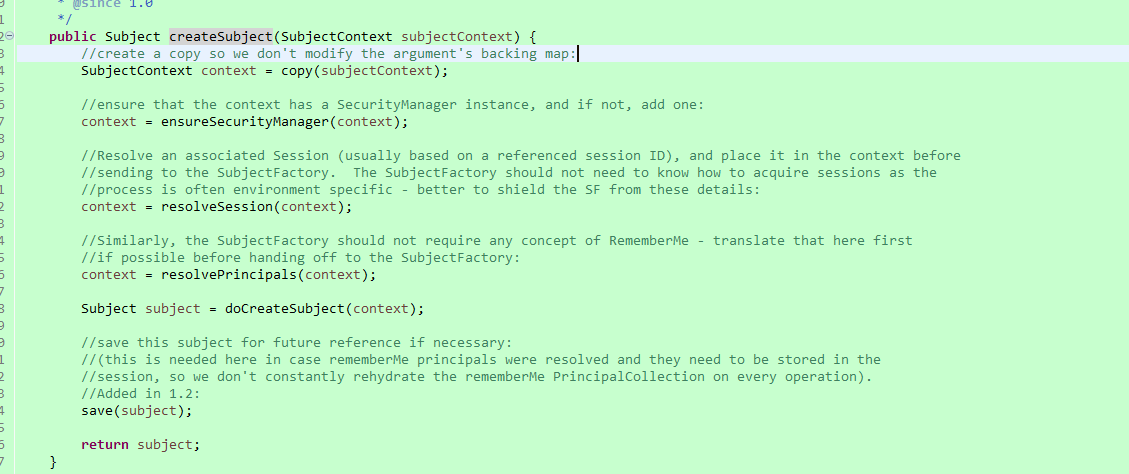
实际上调用的是Subject接口的实现类 DelegatingSubject 中的login 方法

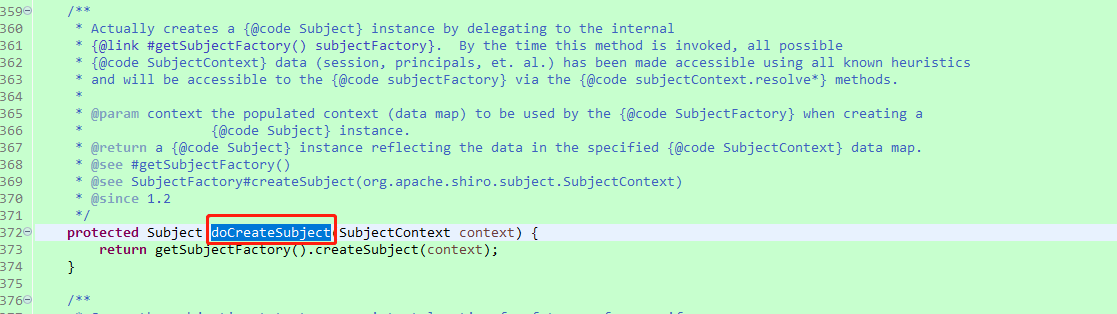


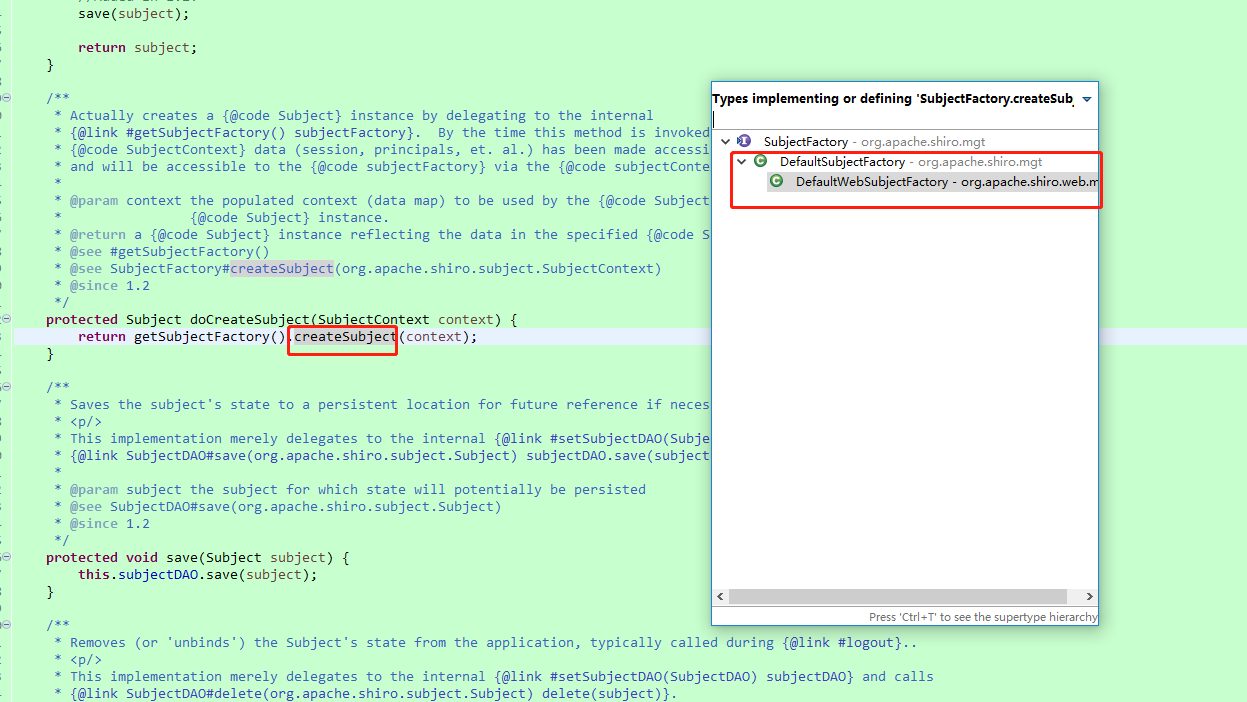
DefaultSecurityManager.Java





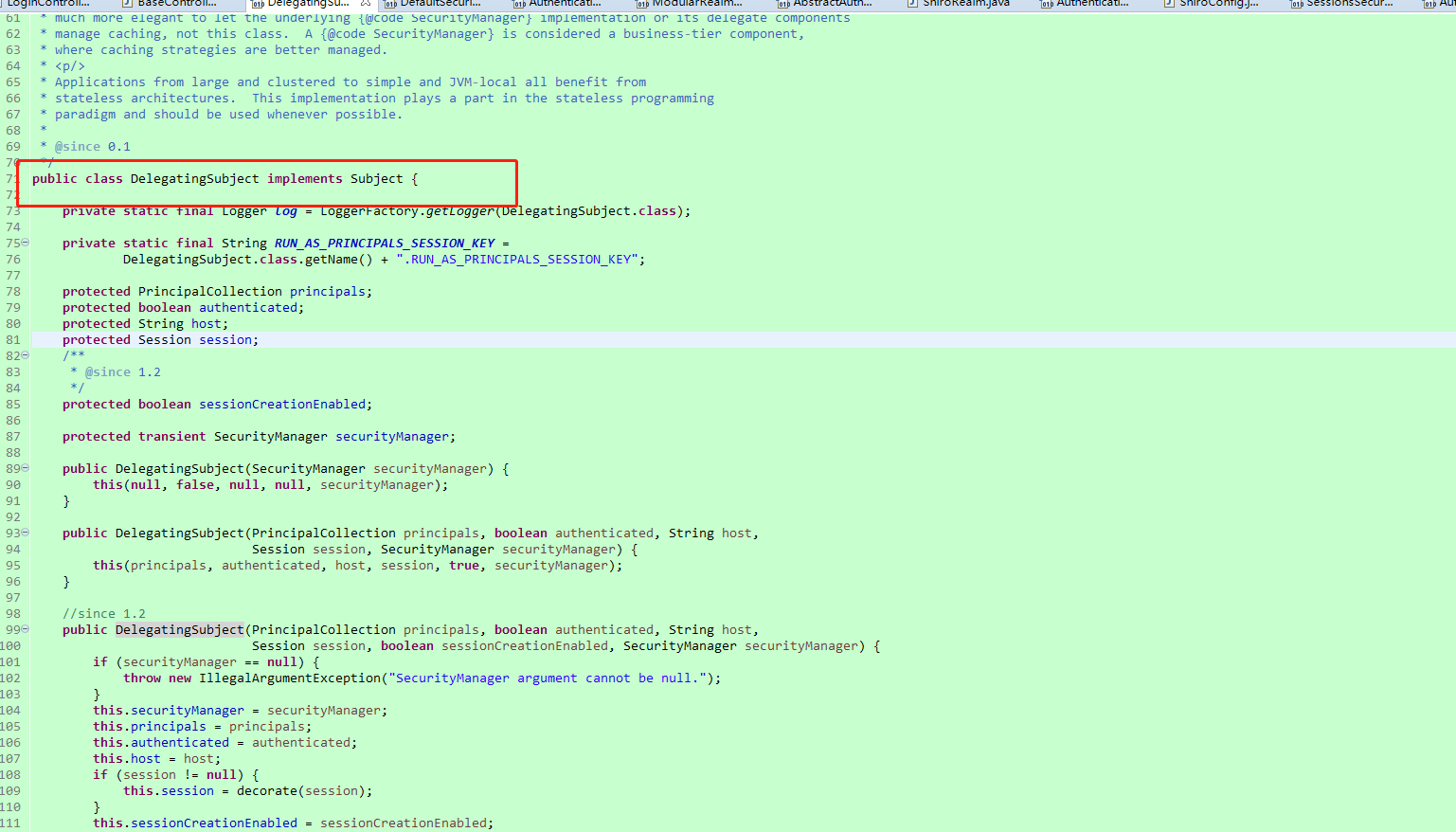








至此 Subject 的实现类 就获取到了



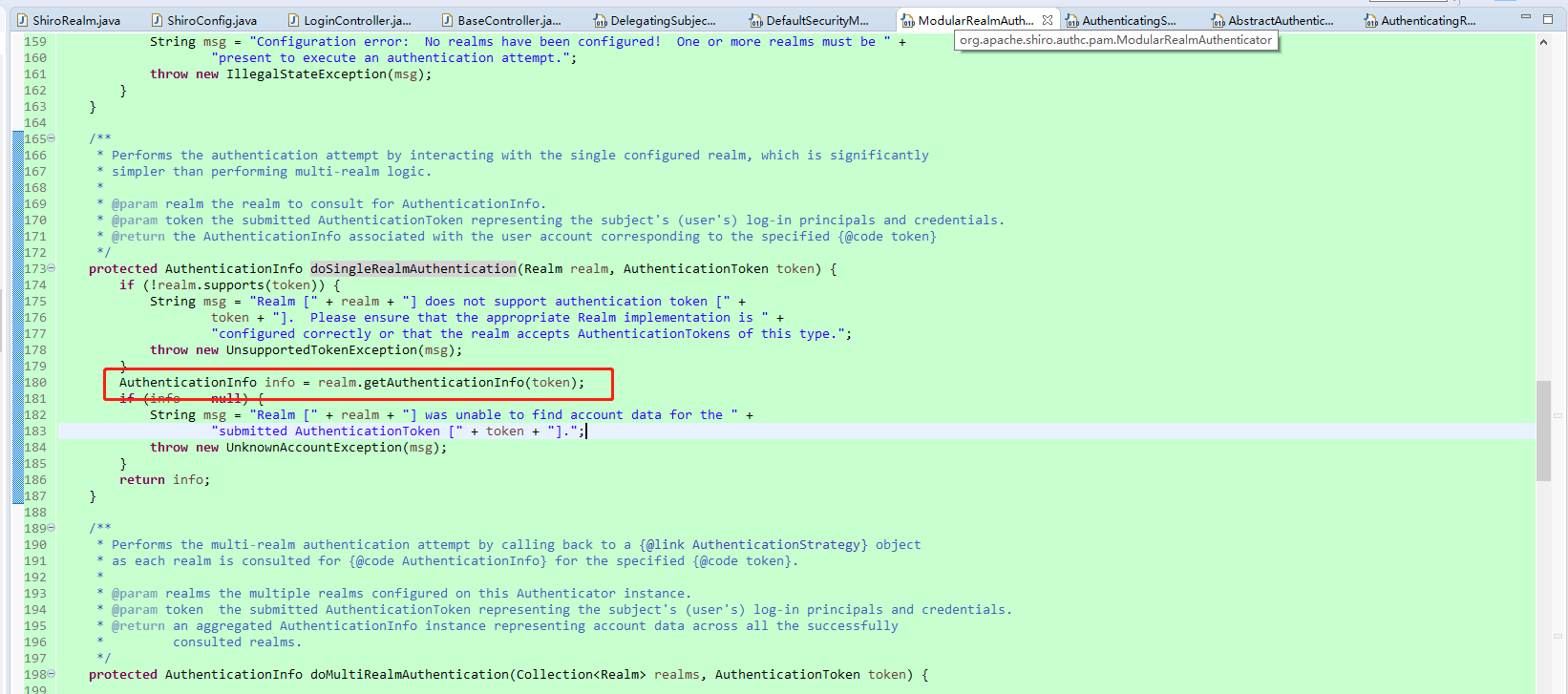
# Shiro 缓存CacheManger

### 缓存的是什么

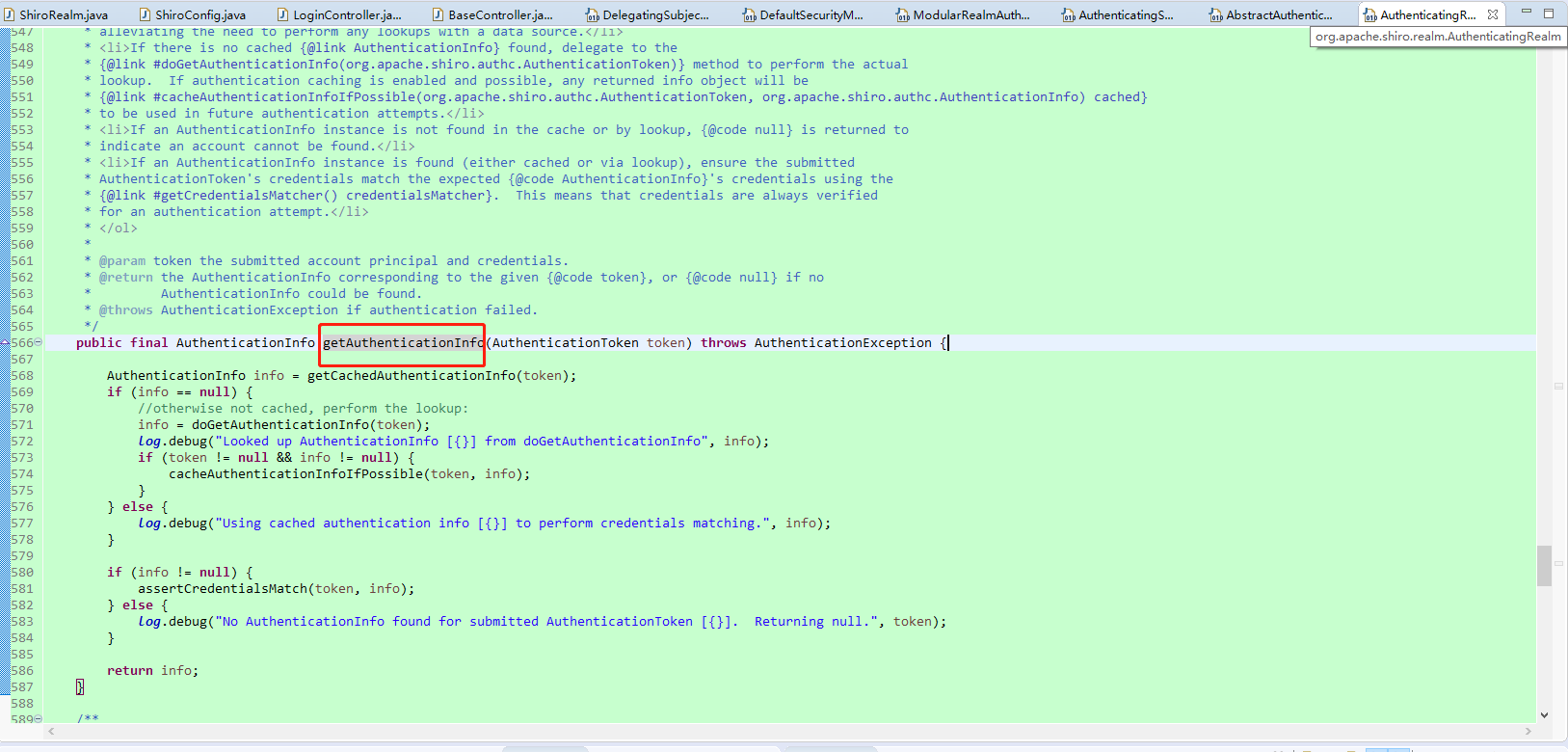
缓存自定义Realm 中的 AuthorizationInfo 角色权限对象，避免每次请求都从数据库中重新加载数据

获取AuthorizationInfo代码实现

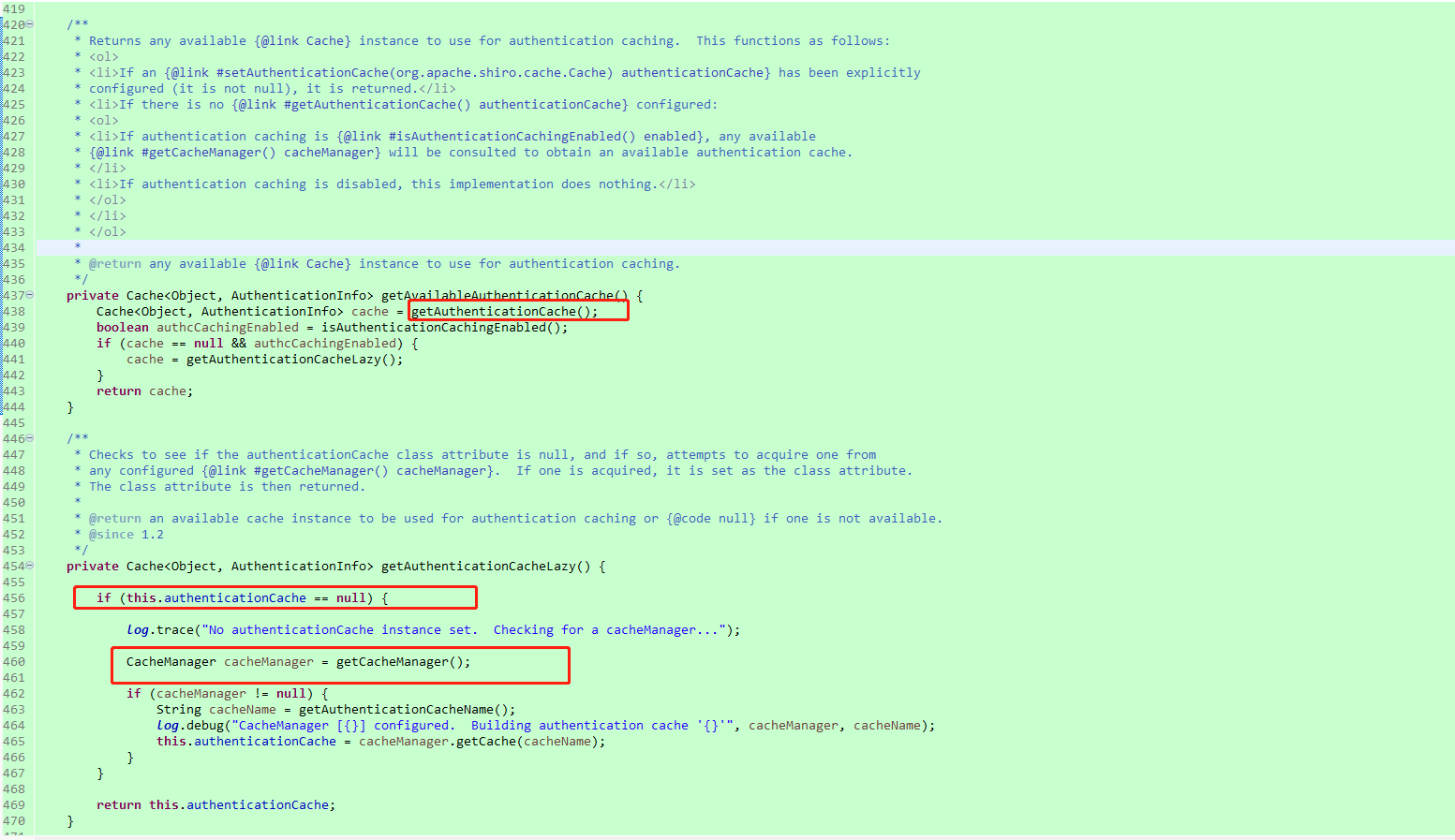
ModularRealmAuthenticator.Java 在加载自定义realm类时候



AuthenticatingRealm.java 先从缓存中查找 ，如果缓存中没有在从Realm 类中重新加载



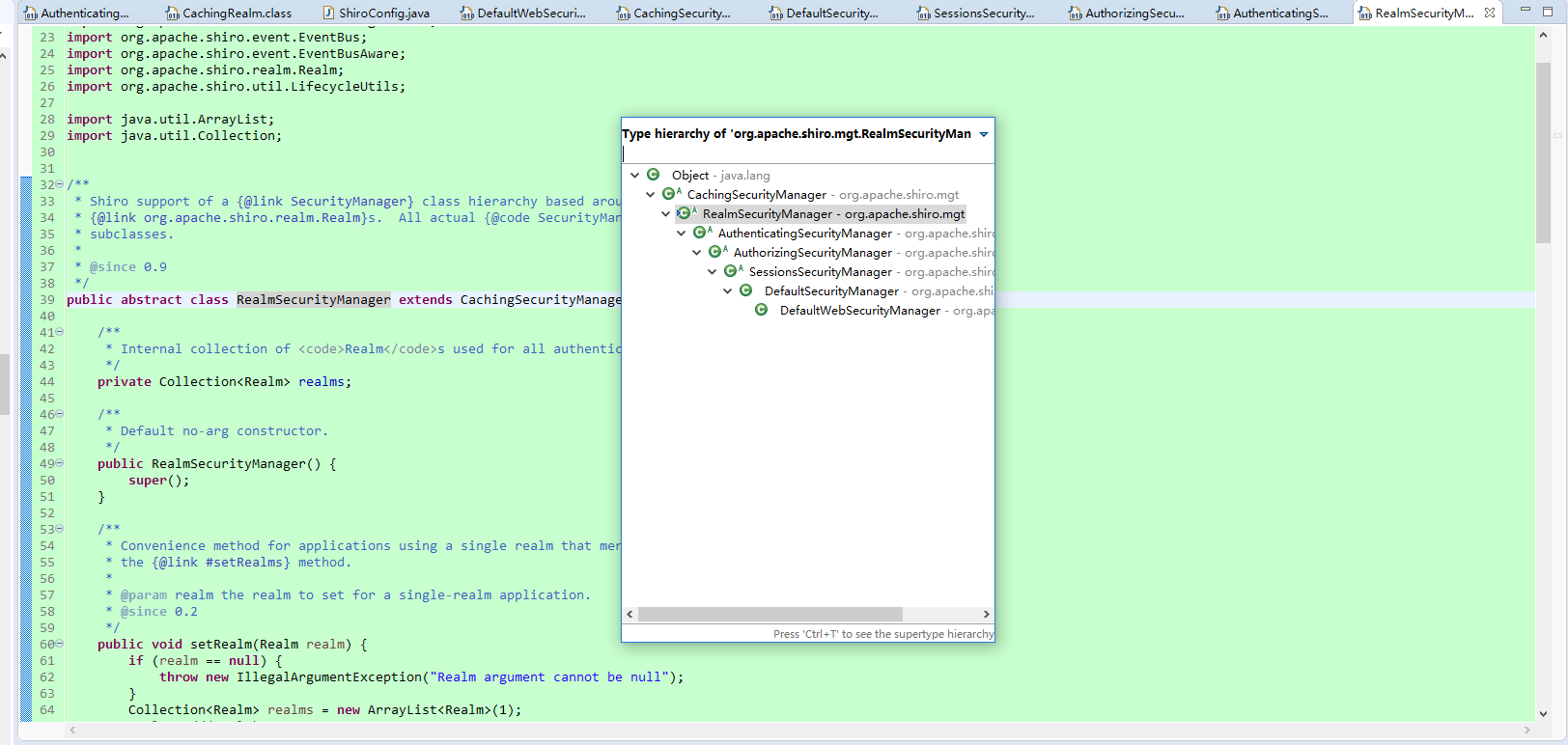
看当前缓存有没有显示设置缓存 如果没有在找设置的缓存 查找当前realm设置的缓存（设置缓存见设置缓存） 通过key 获取当前在缓存对象中的数据



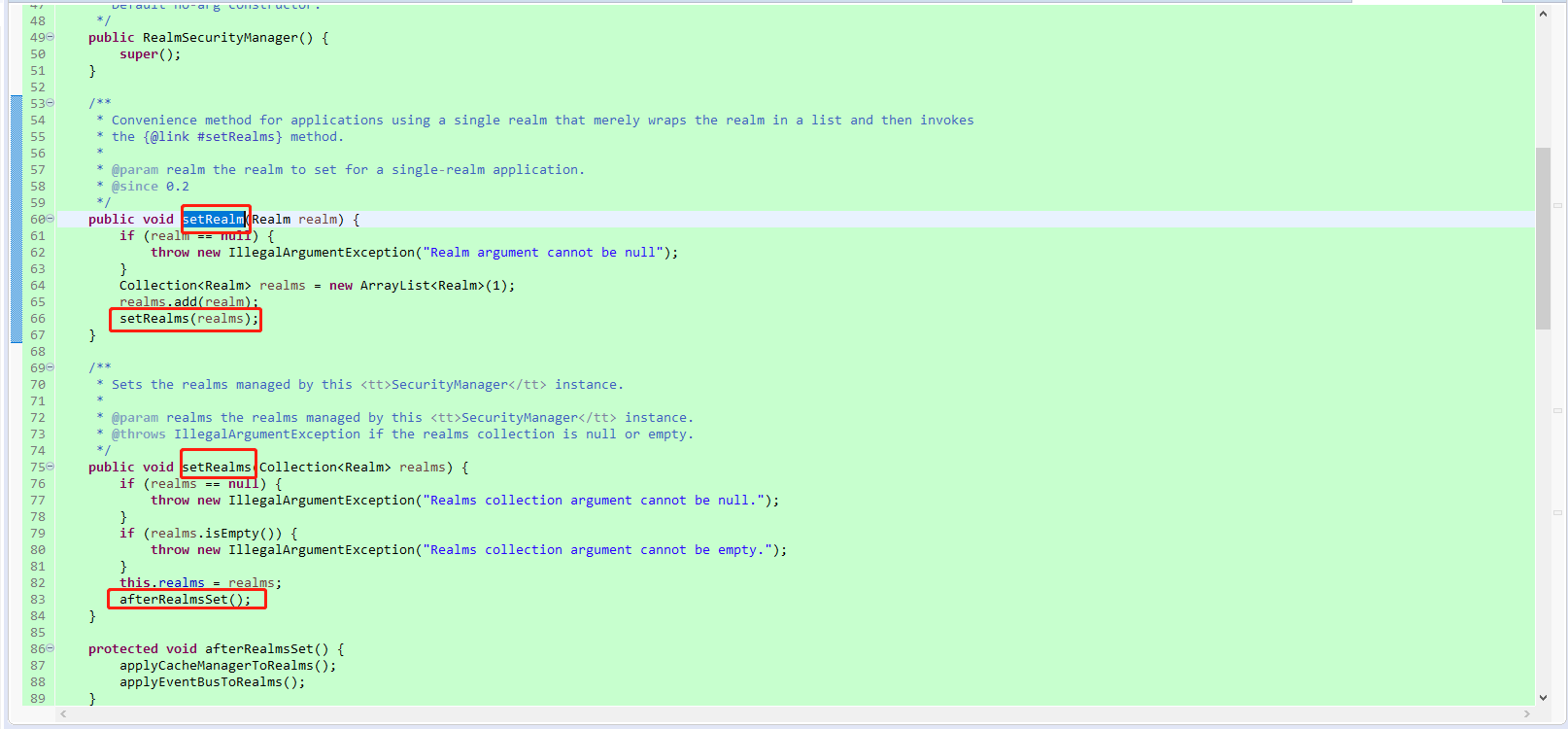
### 设置缓存

SpringBoot 中自定义SecurityManager实现对象 设置setCacheManage

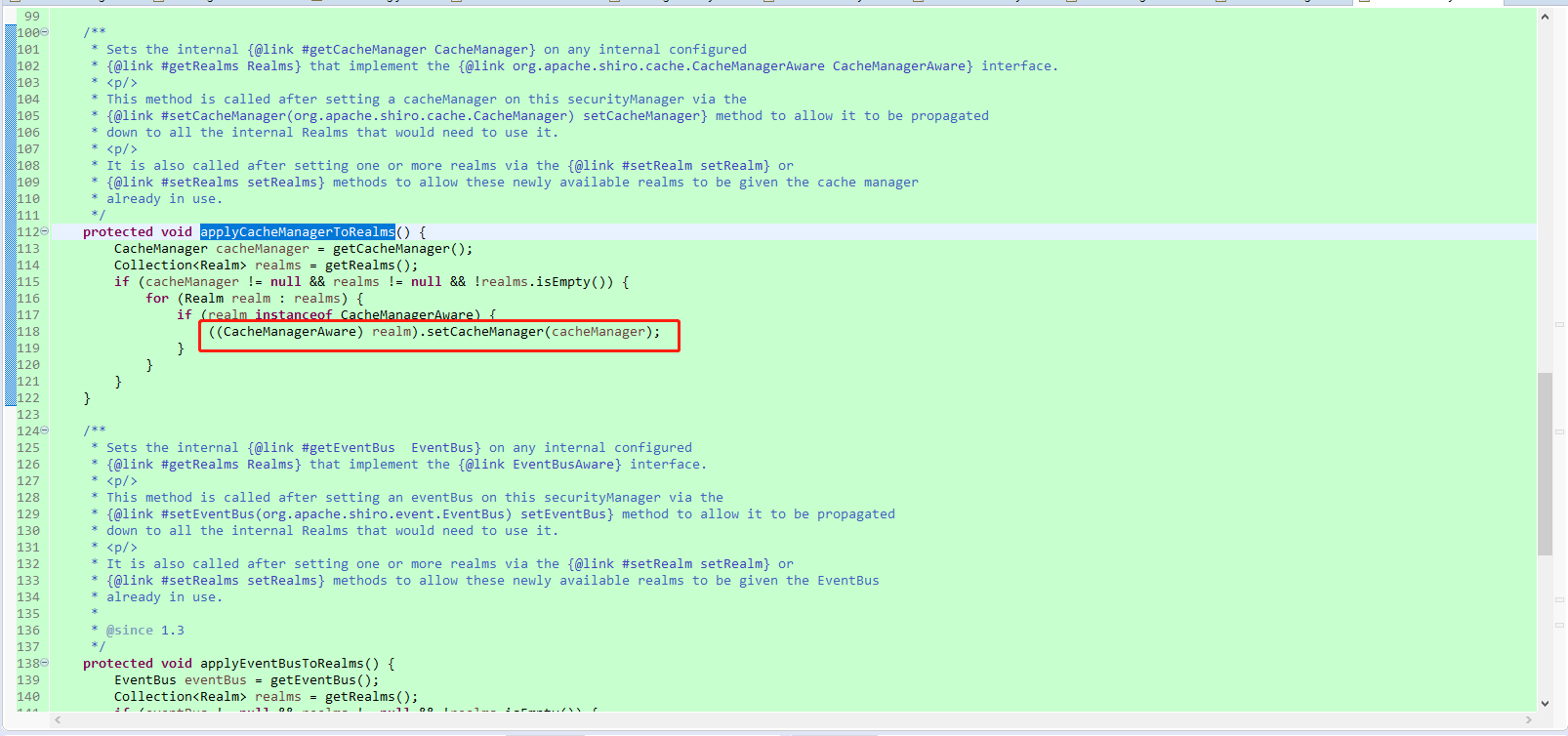




给自定义realm 设置 cacheManager



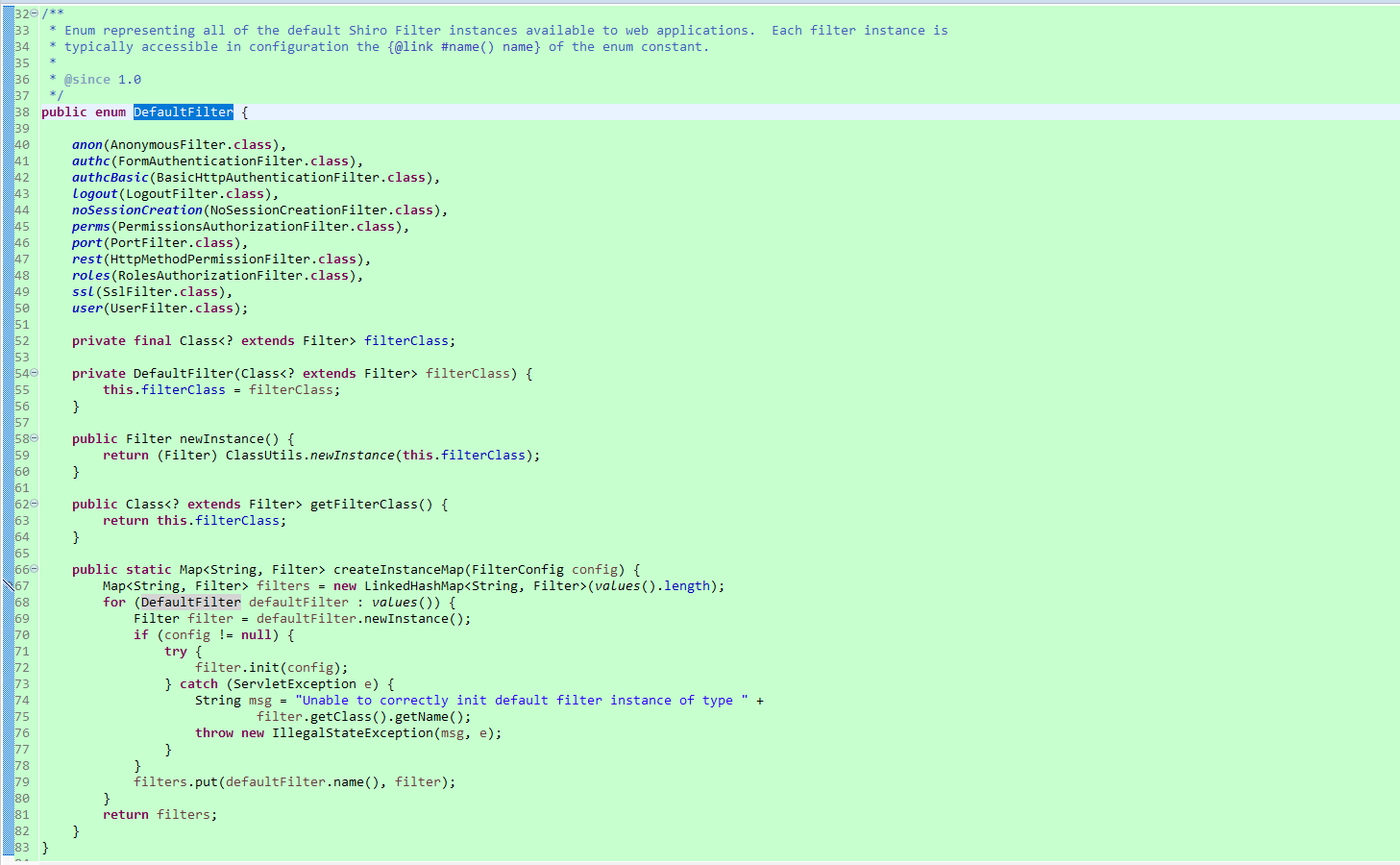
统一将 非空Realm 设置cacheManager



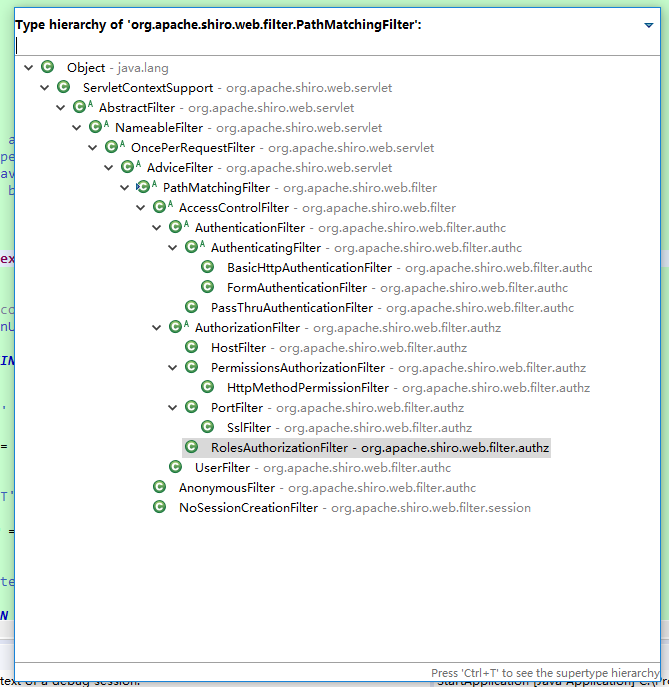
# Filter 对应

### 对应的filter

通过枚举 指定字符对应的Filter

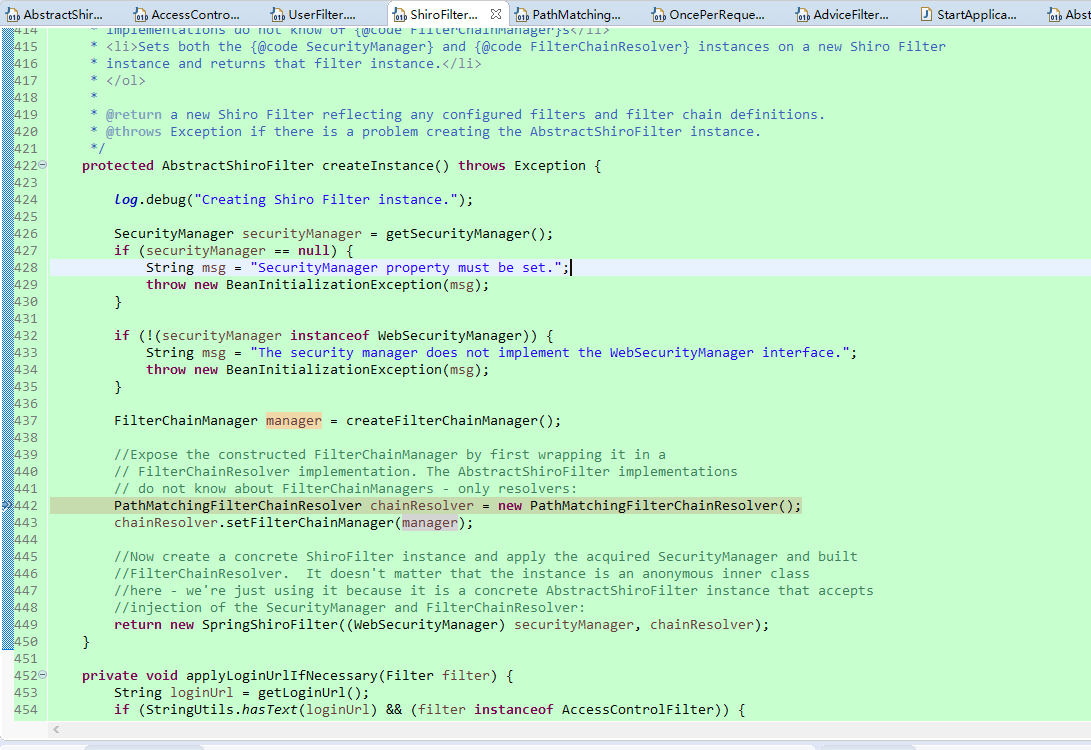


ShiroFilter 中的关系图

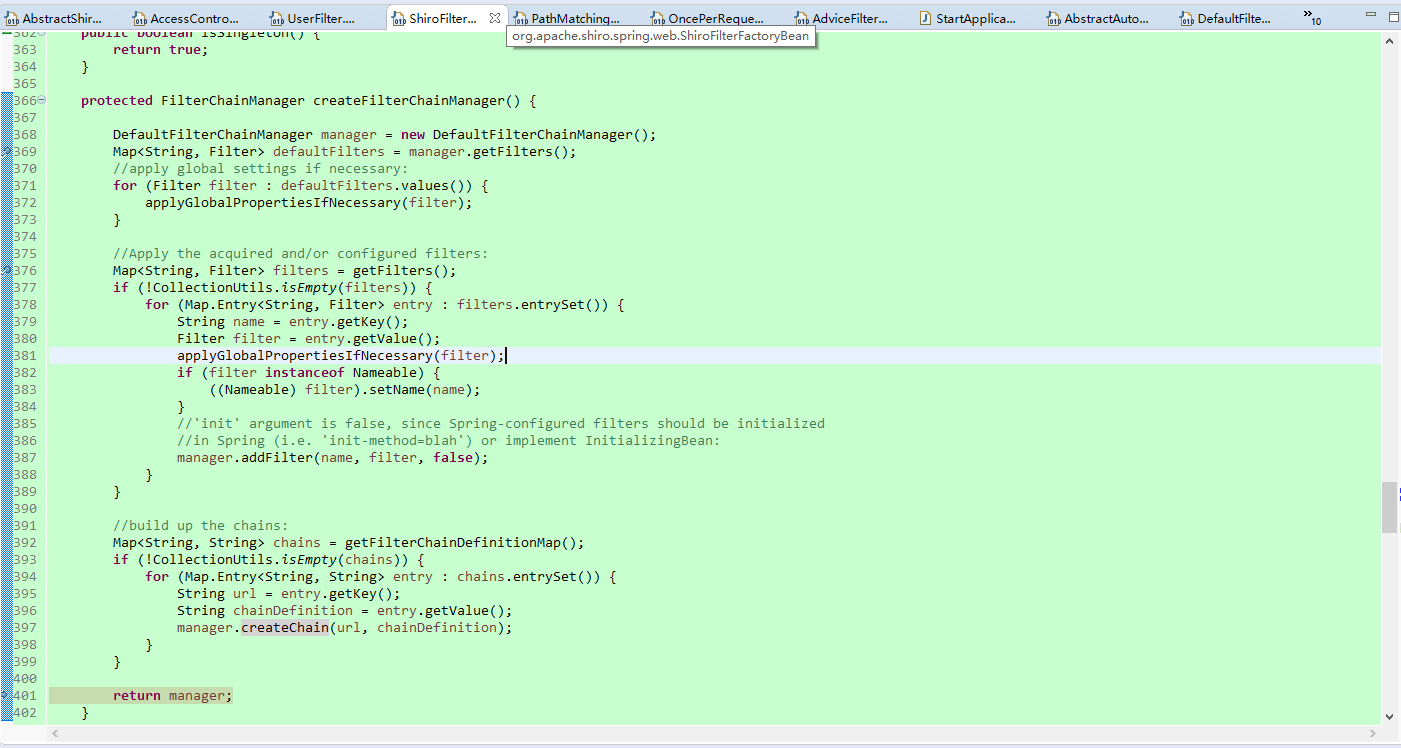


### ShiroFilterFactoryBean

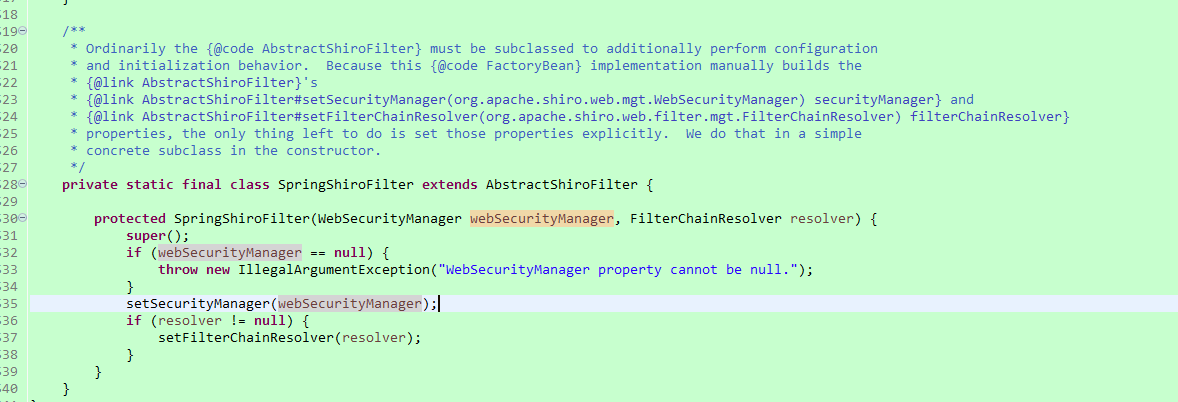
创建 AbstractFilter 实例



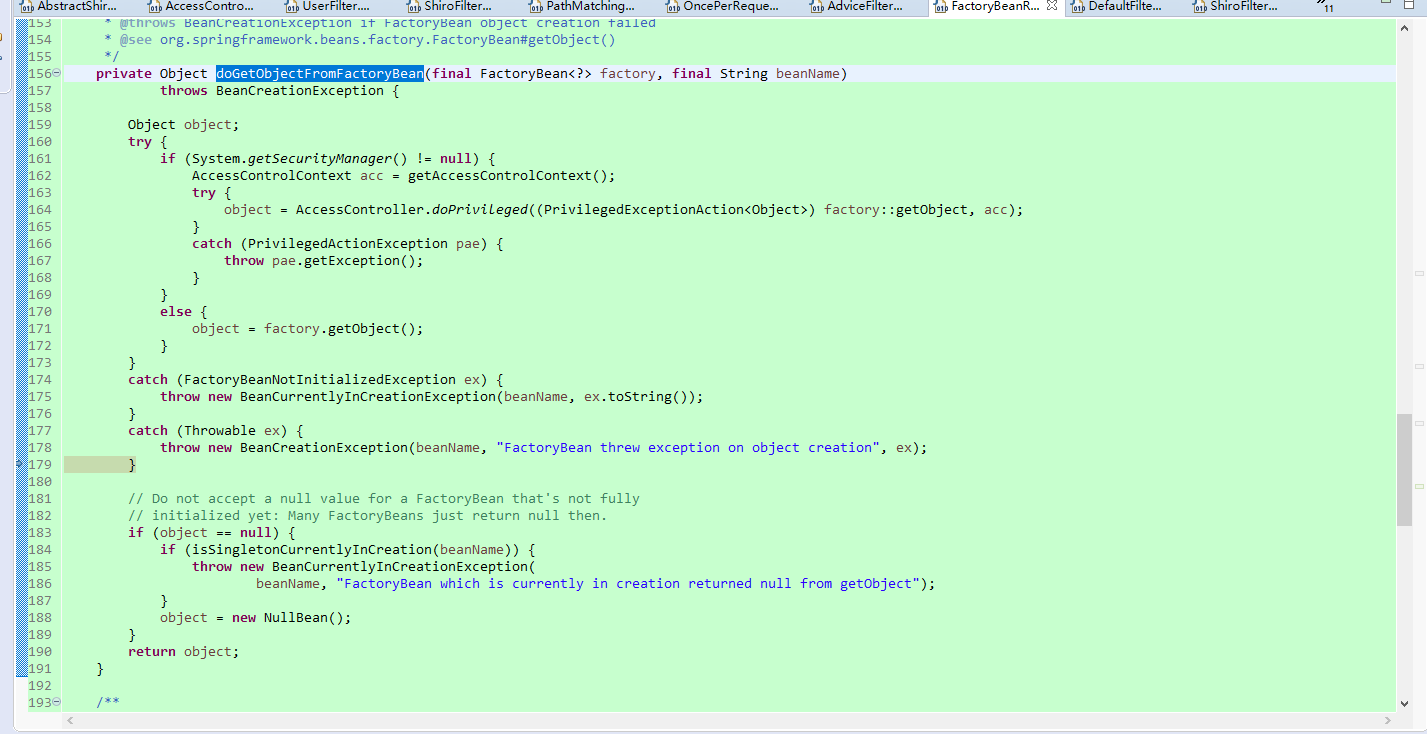
然后通过 ShiroFilterFactoryBean 的createFileterChainManager 方法 将设置的路径 与Filter对应起来



设置 ShiroFilterFactoryBean 中设置的securityManager 和 设置的路径对应的 Filter



ShiroFilterFactoryBean 通过 spring的 FactoryBeanRegistrySupport 抽象类的



将 其实例

### UserFilter

用以验证当前请求路径是否经过登录之后，在filter 中实现就是验证当前subject 是否有用户信息

### 