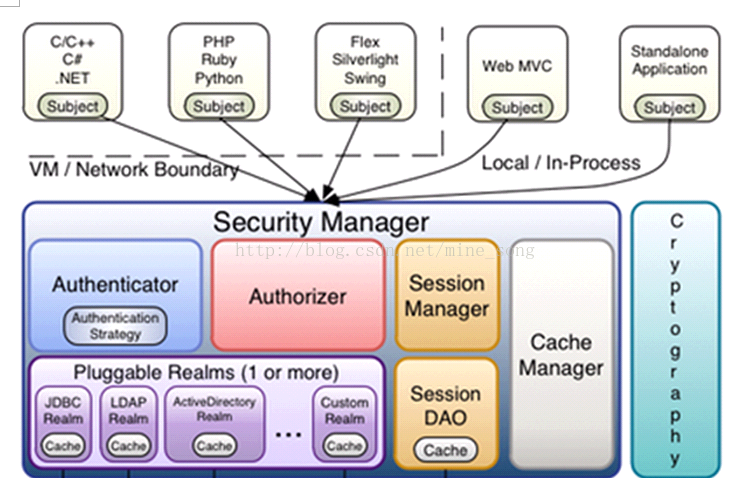
# Shiro是什么

Shiro是apache的一个开源框架，用于权限管理。实现认证、用户授权。

Shiro可以对角色以及权限进行管理。

# Shiro架构



Subject：主体，可以是用户或者程序，主体要访问系统，系统需要对主体进行认证、授权。

SecurityManager：安全管理器，主体进行认证和授权都是通过这里进行

Authenticator：认证器，主体进行认证最终通过authenticator进行的

Authorize：授权器，主体进行授权

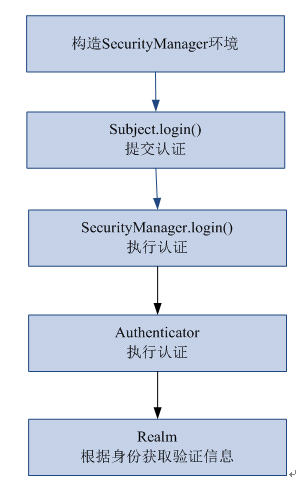
SessionManager：WEB应用中一般是用web容器对session进行管理，shiro也提供一套session管理的方式

SessionDao：通过SessionDao管理session数据

Cache Manager：缓存管理器，主要对session和授权数据进行缓存，比如将授权数据通过cacheManager进行缓存，和ehcache整合对缓存数据进行管理。

Realm：域，相当于数据源，通过realm存取认证、授权相关数据。

# 认证过程



## 认证执行流程

1、通过ini配置文件创建securityManager

2、调用subject.login方法主体提交认证，提交的token

3、securityManager进行认证，securityManager最终由ModularRealmAuthenticator进行认证。

4、ModularRealmAuthenticator调用IniRealm(给realm传入token) 去ini配置文件中查询用户信息

5、IniRealm根据输入的token（UsernamePasswordToken）从 shiro.ini查询用户信息，根据账号查询用户信息（账号和密码）

         如果查询到用户信息，就给ModularRealmAuthenticator返回用户信息（账号和密码）

         如果查询不到，就给ModularRealmAuthenticator返回null

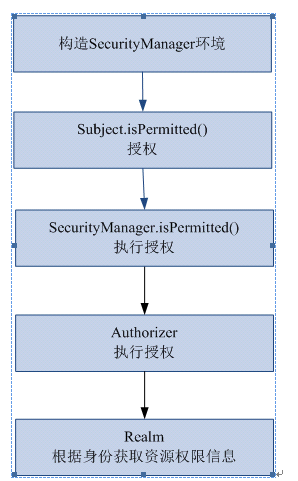
6、ModularRealmAuthenticator接收IniRealm返回Authentication认证信息

         如果返回的认证信息是null，ModularRealmAuthenticator抛出异常

         如果返回的认证信息不是null（说明inirealm找到了用户），对IniRealm返回用户密码 （在ini文件中存在）

         和 token中的密码 进行对比，如果不一致抛出异常

# 授权流程



1、对subject进行授权，调用方法isPermitted（"permission串"）

2、SecurityManager执行授权，通过ModularRealmAuthorizer执行授权

3、ModularRealmAuthorizer执行realm（自定义的Realm）从数据库查询权限数据

调用realm的授权方法：doGetAuthorizationInfo

4、realm从数据库查询权限数据，返回ModularRealmAuthorizer

5、ModularRealmAuthorizer调用PermissionResolver进行权限串比对

6、如果比对后，isPermitted中"permission串"在realm查询到权限数据中，说明用户访问permission串有权限，否则 没有权限，抛出异常。

来源自：https://blog.csdn.net/mine\_song/article/details/61616259