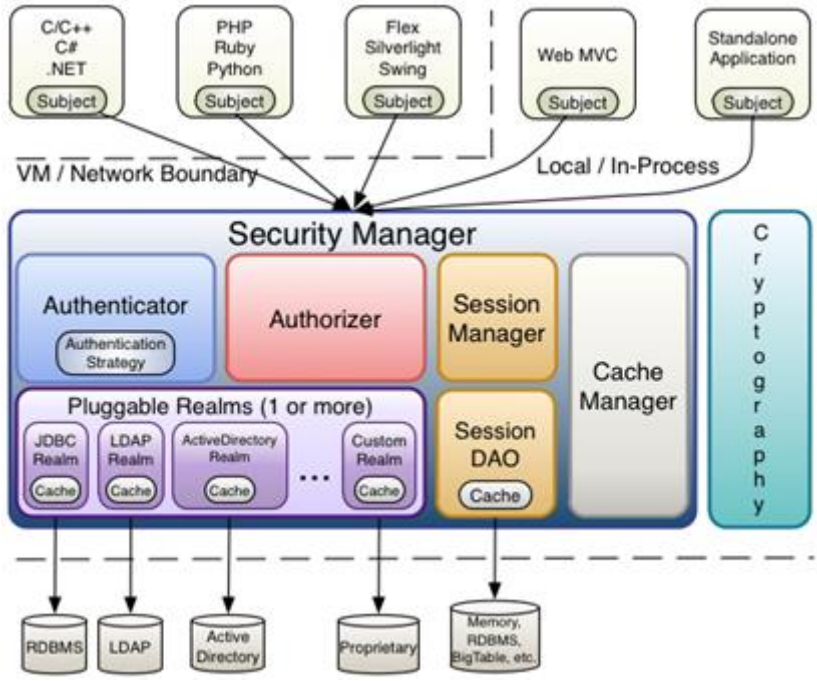
一、简介

Shiro 架构如下图：



Subject：不单单指人还可以是任何与应用交互的“用户”，例如另外一个应用；
SecurityManager：典型的 Facade，Shiro 通过它对外提供安全管理的各种服务。
Authenticator：认证器，负责用户认证的，当用户进行登录时会执行此组件
Authrizer：授权器，或者访问控制器，用来决定用户是否有权限进行相应的操作；即控制着用户能访问应用中的哪些功能；
Realm：Shiro 与安全数据之间的“桥梁”或“连接器”。你可以根据你的需求配置多个 Realm（通常一个数据源就是一个 Realm）。当进行身份认证或授权时可以配置好的 Realms 中获取相关数据，并且访问 Realms 的策略是可以自定义的。可以基于 jdbc，ldap，text，activeDirectory，jndi 等多种方式
SessionManager：这个组件保证了异构客户端的访问，配置简单。它是基于 POJO/J2SE 的，不跟任何的客户端或者协议绑定。
SessionDAO：DAO 大家都用过，数据访问对象，用于会话的 CRUD，比如我们想把 Session 保存到数据库，那么可以实现自己的 SessionDAO，通过如 JDBC 写到数据库；比如想把 Session 放到 Memcached 中，可以实现自己的 Memcached SessionDAO；另外 SessionDAO 中可以使用 Cache 进行缓存，以提高性能；
CacheManager：缓存控制器，来管理如用户、角色、权限等的缓存的；因为这些数据基本上很少去改变，放到缓存中后可以提高访问的性能
Cryptography：密码模块，Shiro 提高了一些常见的加密组件用于如密码加密/解密的。

二、实践及感想

上传的 demo 是 shiro 与 web 的集成，简单的写了一个用户登陆并查看权限、角色、身份是否验证。Demo 使用了 jetty-maven-plugin 和 tomcat7-maven-plugin 插件；这样可以直接使用“mvn jetty:run”或“mvn tomcat7:run”直接运行 webapp 了。这个主要内容在于 web.xml 和

shiro.ini 和 shiro-basicfilterlogin.ini 的配置，具体设置如图。

```
<listener>
    <listener-class>org.apache.shiro.web.env.EnvironmentLoaderListener</listener-class>
</listener>

<filter>
    <filter-name>ShiroFilter</filter-name>
    <filter-class>org.apache.shiro.web.servlet.ShiroFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>ShiroFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

shiro.ini - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
[main]
authc.loginUrl=/login
roles.unauthorizedUrl=/unauthorized
perms.unauthorizedUrl=/unauthorized

logout.redirectUrl=/login

[users]
yt=yt,admin
ty=ty

[roles]
admin=user:*,menu:*

[urls]
/logout2=logout
/login=anon
/logout=anon
/unauthorized=anon
/static/**=anon
/authenticated=authc
/role=authc,roles[admin]
/permission=authc,perms["user:create"]
```

shiro-basicfilterlogin.ini - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
[main]
authcBasic.applicationName=please login

perms.unauthorizedUrl=/unauthorized
roles.unauthorizedUrl=/unauthorized
[users]
yt=yt,admin
ty=ty

[roles]
admin=user:*,menu:*

[urls]
/role=authcBasic,roles[admin]
```

这样一来，输入 http://localhost:8080/TestShiro/login 进行登录，登录成功后接着可以访问 http://localhost:8080/TestShiro /authenticated 来显示当前登录的用户：

输入 http://localhost:8080/TestShiro/role，会弹出 Basic 验证对话框输入“zhang/123”即可登录成功进行访问。

使用帐号“yt/yt”进行登录，再访问/role 或/permission 时会跳转到成功页面（因为其授权成功了）；如果使用帐号“ty/ty”登录成功后访问这两个地址会跳转到“/unauthorized”即没有授权页面。

通过 logout.redirectUrl 指定退出后重定向的地址；通过/logout2=logout 指定退出 url 是/logout2。这样当我们登录成功后然后访问/logout2 即可退出。

个人对于 shiro 的看法：较之 JAAS 和 Spring Security，Shiro 在保持强大功能的同时，还在简单性和灵活性方面拥有巨大优势，shiro 的配置简单易懂，上手快，除了自己写以外对新手还提供了 tag，而且 shiro 对许多其他框架兼容性更好，并且可以运行在任何环境中（这点在我看来是 shiro 能够被广泛推崇并使用的主要原因），还有重要的一点是 shiro 在集群会话时是独立于容器的。