安全框架Shiro学习笔记

- **Java安全框架**

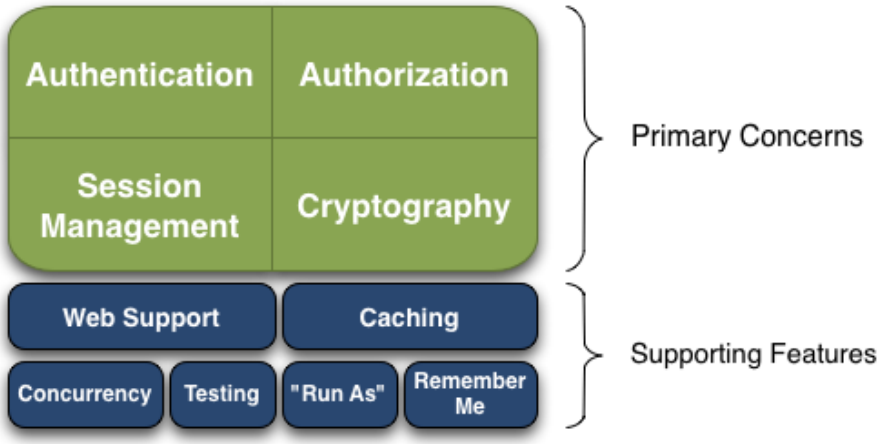
作为广泛应用的互联网时代语言之一，Java也考虑过安全上的问题，比如如何保证下载到本地的Java程序的安全性，如何对Java程序访问本地资源做有限授权。而Java安全框架就是实现这些特定目标的途径之一，相比Java本身的安全机制包括安全模型，像Shrio这样的干净、容器无关的安全框架更好的解决了某些特定问题。

- **Apache Shiro**

Shiro 前身是JSecurity，实现身份认证、授权、加密和会话管理的操作，较为简单、灵活但也强大。Shiro希望能在所有的应用环境中实现自己的功能，不借助第三方框架、容器、服务器等，它主要对用户权限和会话控制进行封装，并且支持跨平台的登录权限认证，可以在非 web 或 EJB 容器的环境下可以任意使用Session API。

安全的四大基石：
身份认证---用户身份识别。
授权---访问控制，比如确定某个用户是否拥有某个操作的使用权限。
加密---基于各种强大的加密算法，保护或隐藏数据防止被偷窥。
会话管理---用户相关的状态，可以在非web或EJB上用的。

Shiro正是强调了这四个功能，在此基础上提供其他的功能支持，比如web支持，缓存，并发，测试，Run As的身份假设以及Remember Me的跨session 记录用户的身份。Shiro功能框架：



- **Shiro特点：**

1. 易于理解的 Java Security API，易用性让人还算开心；
2. 简单的身份认证（登录），支持多种数据源
   （LDAP，JDBC，Kerberos，ActiveDirectory 等）；
3. 简单的授权（访问控制），支持细粒度的授权；

4. 支持一级缓存，以提升应用程序的性能；
5. 内置的基于 POJO 企业会话管理，适用于 Web 以及非 Web 的环境；
6. 异构客户端会话访问；
7. 非常简单的加密 API；
8. 不跟任何的框架或者容器捆绑，可以独立运行，即在不同容器中可以以同样方式工作（比较实用了，简单又强大吧）；
9. 可以工作在任何应用环境中，不强加任何规范，无需过多依赖。

- **API----三个核心组件：**

Subject：相当于user但意义超过user，代表了当前对象的安全操作。这里也反映了Shrio的思考是基于用户角度。

```
Subject currentUser = SecurityUtils.getSubject();
```

这样就可以获取当前subject，是位于应用程序中的用户信息。

SecurityManager：管理所有用户的安全操作。核心中的核心。

这是开发时的重点，但应该是一个程序一般一个SecurityManager即可，而且一旦做好之后就不用对它进行更多操作。

Realms：充当Shiro与应用安全数据间的“桥梁”或者“连接器”，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。

鉴于Shiro是定义在一个web程序中的过滤器框架，我没有具体的做一个程序并将Shiro嵌入，而是简单学习了Shiro的各种使用方法。

- **web支持：**

首先是在web中安装使用Shiro，只需要在web.xml中定义一个Shiro Servlet过滤器，Shiro默认配置是INI配置（一个不太习惯的）。

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

<filter>  <!--加入过滤器-->
    <filter-name>ShiroFilter</filter-name>
    <filter-class>org.apache.shiro.web.servlet.IniShiroFilter</filter-class>
    <!-- 没有init-param属性就默认从classpath:shiro.ini装入INI配置 -->
</filter>

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

在Shiro中，之前的代码有提到获得subject和他的当前session，我们要对他们进行角色和权限的验证。要验证要先登录。

- **认证：**

```
if ( !currentUser.isAuthenticated() ) {
    UsernamePasswordToken token = new UsernamePasswordToken("用户名称", "password");
    token.setRememberMe(true);
    currentUser.login(token);
}
```

在调用了login方法后，SecurityManager会发送给给已配置的Realm，执行必须的认证检查。对于登录失败的情况我们可以捕获这些异常并用try-chatch的语句进行处理。比如：

```
try {
  currentUser.login( token );   //无异常
} catch ( UnknownAccountException uae ) {
  ...//username 不存在
} catch ( IncorrectCredentialsException ice ) {
  ...//password 不匹配
} catch ( LockedAccountException lae ) {
  ...//账号锁住了，不能登入
}
  ... 更多类型异常 ...
} catch ( AuthenticationException ae ) {
  ...//未考虑到的问题
}
```

登陆了之后我们就可以对他们的角色或者权限进行检查，然后以便我们对他们的访问，或者应用程序的功能进行控制。

- **授权：**

我们可以检查他们的角色，如下是检查是否拥有角色1：

```
if ( currentUser.hasRole( "角色1" ) ) {
  ...//提供角色1对应的功能
} else {
 ...//sorry。。。
}
```

对权限的检查如下：

```
if ( currentUser.isPermitted( "user:create" ) ) {   //创建用户的权限
  ...//比如可以显示创建用户的按钮或者其他的形式来表现“创建用户”的功能
} else {
  ...//sorry!
}
```

我们可以用Shiro对用户的状态进行管理来保证安全。

- **会话管理：**

```
Session session = currentUser.getSession(); //获取用户当前session的一些参数。
session.setAttribute( "someKey", "aValue" );//
```

我们要对用户的数据进行隐藏加密，可以在任何情况下调用Shiro的加密API。

- **加密：**

对于加密支持，Shiro关注加密Hash（消息摘要）和加密密码两个领域。

首先Hash一个文件：

```
String hex = new Md5Hash(myFile).toHex();
```

然后可以直接实例化一个CipherService创建秘钥进行加密算法。

Shiro还有缓存、并发等功能支持，感觉目前PJ应该用不到，就没写。

最后，不使用程序的时候，要退出登录：

```
currentUser.logout(); //清除识别信息，设置session失效.
```

总结

Apache Shiro是一个功能齐全、健壮、通用的Java安全框架，你可以用其为你的应用护航。通过简化应用安全的四个领域，即认证、授权、会话管理和加密，在真实应用中，应用安全能更容易被理解和实现。Shiro的简单架构和兼容JavaBean使其几乎能够在任何环境下配置和使用。附加的Web支持和辅助功能，比如多线程和测试支持，让这个框架为应用安全提供了"一站式"服务。Apache Shiro开发团队将继续前进，精炼代码库和支持社区。随着持续被开源和商业应用采纳，可以预期Shiro会继续发展壮大。

别人的总结，就我看来，当你开发的web应用对认证、授权、会话管理和加密四个领域的安全有更进一步的要求时，使用Shiro框架就是一个比较好的选择了。