什么是JAAS,以及灵活的Java安全机制Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/571/2021_2022__E4_BB_80_E4_B9_88_E6_98_AFJ_c104_571327.htm Java Authentication Authorization Service(JAAS，Java验证和授权API)提供了灵活和可伸缩的机制来保证客户端或服务器端的Java程序。Java早期的安全框架强调的是通过验证代码的来源和作者，保护用户避免受到下载下来的代码的攻击。JAAS强调的是通过验证谁在运行代码以及他/她的权限来保护系统面受用户的攻击。它让你能够将一些标准的安全机制，例如Solaris NIS(网络信息服务)、Windows NT、LDAP(轻量目录存取协议)，Kerberos等通过一种通用的，可配置的方式集成到系统中。 你是否曾经需要为一个应用程序实现登录模块呢?如果你是一个比较有经验的程序员，相信你这样的工作做过很多次，而且每次都不完全一样。你有可能把你的登录模块建立在Oracle数据库的基础上，也有可能使用的是NT的用户验证，或者使用的是LDAP目录。如果有一种方法可以在不改变应用程序级的代码的基础上支持上面提到的所有这一些安全机制，对于程序员来说一定是一件幸运的事。 现在你可以使用JAAS实现上面的目标。JAAS是一个比较新的的Java API。在J2SE 1.3中，它是一个扩展包.在J2SE 1.4中变成了一个核心包。在本文中，我们将介绍JAAS的一些核心概念，然后通过例子说明如何将JAAS应用到实际的程序中。本文的例子是根据我们一个基于Web的Java应用程序进行改编的，在这个例子中，我们使用了关系数据库保存用户的登录信息。由于使用了JAAS，我们实现了一个健壮而灵活的登录和身份验证模块。 客户端和服

务器端的JAAS 开发人员可以将JAAS应用到客户端和服务器端。在客户端使用JAAS很简单。在服务器端使用JAAS时情况要复杂一些。目前在应用服务器市场中的JAAS产品还不是很一致，使用JAAS的J2EE应用服务器有一些细微的差别。例如JBossSx使用自己的结构，将JAAS集成到了一个更大的安全框架中.而虽然WebLogic 6.x也使用了JAAS，安全框架却完全不一样。 现在你能够理解为什么我们需要从客户端和服务器端的角度来看JAAS了。我们将在后面列出两种情况下的例子。为了使服务器端的例子程序更加简单，我们使用了Resin应用服务器。 核心JAAS类 在使用JAAS之前，你首先需要安装JAAS。在J2SE 1.4中已经包括了JAAS，但是在J2SE 1.3中没有。如果你希望使用J2SE 1.3，你可以从SUN的官方站点上下载JAAS。当正确安装了JAAS后，你会在安装目录的lib目录下找到jaas.jar。你需要将该路径加入Classpath中。(注：如果你安装了应用服务器，其中就已经包括了JAAS，请阅读应用服务器的帮助文档以获得更详细的信息)。在Java安全属性文件java.security中，你可以改变一些与 JAAS相关的系统属性。该文件保存在/lib/security目录中。 在应用程序中使用JAAS验证通常会涉及到以下几个步骤： 1. 创建一个LoginContext的实例。 2. 为了能够获得和处理验证信息，将一个CallBackHandler对象作为参数传送给LoginContext。 3. 通过调用LoginContext的login()方法来进行验证。 4. 通过使用login()方法返回的Subject对象实现一些特殊的功能(假设登录成功)。 下面是一个简单的例子： LoginContext lc = new LoginContext("MyExample"). try { lc.login(). } catch (LoginException) { // Authentication failed. } // Authentication

successful, we can now continue. // We can use the returned Subject if we like. Subject sub = lc.getSubject(). Subject.doAs(sub, new MyPrivilegedAction()). 在运行这段代码时，后台进行了以下的工作。 1. 当初始化时，LoginContext对象首先在JAAS配置文件中找到MyExample项，然后更具该项的内容决定该加载哪个LoginModule对象。 2. 在登录时，LoginContext对象调用每个LoginModule对象的login()方法。 3. 每个login()方法进行验证操作或获得一个CallbackHandle对象。 4. CallbackHandle对象通过使用一个或多个CallBack方法同用户进行交互，获得用户输入。 5. 向一个新的Subject对象中填入验证信息。 我们将对代码作进一步的解释。但是在这之前，让我们先看代码中涉及到的核心JAAS类和接口。这些类可以被分为三种类型：普通类型 Subject，Principal，凭证 验证 LoginContext，LoginModule，CallBackHandler，Callback 授权 Policy，AuthPermission，PrivateCredentialPermission 上面列举的类和接口大多数都在javax.security.auth包中。在J2SE 1.4中，还有一些接口的实现类在com.sun.security.auth包中。 普通类型：Subject，Principal，凭证 Subject类代表了一个验证实体，它可以是用户、管理员、Web服务，设备或者其他的过程。该类包含了三中类型的安全信息： 身份(Identities)：由一个或多个Principal对象表示 公共凭证(Public credentials)：例如名称或公共秘钥 私有凭证(Private credentials)：例如口令或私有密钥 Principal对象代表了Subject对象的身份。它们实现了java.security.Principal和 java.io.Serializable接口。在Subject类中，最重要的方法是getName()。该方法返回一个身份名称。在Subject对象中包含了多个Principal对象，因此它可以拥有多

个名称。由于登录名称、身份证号和Email地址都可以作为用户的身份标识，可见拥有多个身份名称的情况在实际应用中是非常普遍的情况。 在上面提到的凭证并不是一个特定的类或借口，它可以是任何对象。凭证中可以包含任何特定安全系统需要的验证信息，例如标签(ticket)，密钥或口令。Subject对象中维护着一组特定的私有和公有的凭证，这些凭证可以通过getPrivateCredentials()和 getPublicCredentials()方法获得。这些方法通常在应用程序层中的安全子系统被调用。 验证：LoginContext 在应用程序层中，你可以使用LoginContext对象来验证Subject对象。LoginContext对象同时体现了JAAS的动态可插入性(Dynamic Pluggability)，因为当你创建一个LoginContext的实例时，你需要指定一个配置。LoginContext通常从一个文本文件中加载配置信息，这些配置信息告诉LoginContext对象在登录时使用哪一个LoginModule对象。 下面列出了在LoginContext中经常使用的三个方法： login () 进行登录操作。该方法激活了配置中制定的所有LoginModule对象。如果成功，它将创建一个经过了验证的Subject对象.否则抛出LoginException异常。 getSubject () 返回经过验证的Subject对象 logout () 注销Subject对象，删除与之相关的Principal对象和凭证 验证：LoginModule LoginModule是调用特定验证机制的接口。J2EE 1.4中包含了下面几种LoginModule的实现类： JndiLoginModule 用于验证在JNDI中配置的目录服务 Krb5LoginModule 使用Kerberos协议进行验证 NTLoginModul 使用当前用户在NT中的用户信息进行验证 UnixLoginModule 使用当前用户在Unix中的用户信息进行验证 同上面这些模块绑定在一起的还有对应的Principal接口的实现

类，例如NTDomainPrincipal和UnixPrincipal。这些类在com.sun.security.auth包中。 LoginModule接口中包含了五个方法： initialize () 当创建一LoginModule实例时会被构造函数调用 login () 进行验证 commit () 当LgoninContext对象接受所有LoginModule对象传回的结果后将调用该方法。该方法将Principal对象和凭证赋给Subject对象。 abort () 当任何一个LoginModule对象验证失败时都会调用该方法。此时没有任何Principal对象或凭证关联到Subject对象上。 logout () 删除与Subject对象关联的Principal对象和凭证。 在应用程序的代码中，程序员通常不会直接调用上面列出的方法，而是通过LigonContext间接调用这些方法。 验证：CallbackHandler和Callback CallbackHandler和Callback对象可以使LoginModule对象从系统和用户那里收集必要的验证信息，同时独立于实际的收集信息时发生的交互过程。 JAAS在javax.sevurity.auth.callback包中包含了七个Callback的实现类和两个CallbackHandler的实现类： ChoiceCallback、ConfirmationCallback、LogcaleCallback、NameCallback、PasswordCallback、TextInputCallback、TextOutputCallback、DialogCallbackHandler和TextCallBackHandler。Callback接口只会在客户端会被使用到。我将在后面介绍如何编写你自己的CallbackHandler类。 