第一部分

LDAP 和 Netscape Directory SDK for Java 简介

第1章 LDAP 简介
本章介绍 LDAP 协议及其思想。

第2章 使用 Netscape Directory SDK for Java
本章描述 Lightweight Directory Access Protocol(LDAP) Java 类和 Netscape Directory SDK for Java。

第3章 快速入门
本章介绍了一个用 Netscape Directory SDK for Java 编写的简单的 LDAP 终端的客户。

第 1 章　LDAP 简介

本章介绍 LDAP协议及其思想。

LDAP(Lightweight Directory Access Protocol ，即轻便目录访问协议)是 Internet 目录协议。这个协议由位于 Ann Arbor 的 Michigan 大学和 Internet Engineering Task Force 联合开发成功，其目的就是对目录服务器进行访问和管理。

本章由下面几节组成：

- 目录服务器的的工作方式
- LDAP服务器组织目录的方式
- LDAP 客户端和服务器的工作方式
- 了解 LDAP 协议 3.0版
- 其他内容

如果对 LDAP 已经比较熟悉，可以直接跳到第 2章。

1.1　目录服务器的工作方式

目录由包含描述信息的条目 (Entry)组成。例如，可能会包含描述人或打印机、传真机等网络资源的条目。

描述信息保存在条目的属性中。每个属性描述一种特定的信息。例如，描述人的属性可能包括这个人的姓名（常用名或其他名字）、电话号码以及 Email 地址。

例如，Barbara Jensen 的条目可能包含以下一些属性：

cn:Barbara Jensen

mail:babs@ace.com

telephoneNumber:555-1212

roomNumber:3995

对应一个属性，可以有多个值，例如，一个属性可能有两个常用名（一个正式名和一个昵称）或者两个电话号码：

cn：Barbara Jensen

cn：Jenny Jensen

mail：jen@ace.com

telephoneNumber：555-1213

telephoneNumber：555-2059

roomNumber：3996

属性也可以包含二进制数据。例如，一个人的属性可能包括这个人的 JPEG 格式的照片或保存在音频格式文件中的声音。

目录服务器是一个分布式数据库软件，用来管理目录中的条目和属性。它也使得条目和属性可以被用户或其他应用软件访问。Netscape Directory Server就是目录服务器的一

个例子。

例如，用户可以使用目录服务器来查找某个人的电话号码；而另一个软件可能用它来得到一个 Email 地址的列表。

LDAP 是定义目录服务以及如何访问这些服务的协议。它基于客户一服务器模式。LDAP 服务器提供目录服务，而 LDAP 客户端使用目录服务来访问条目和属性。

LDAP 服务器的一个例子就是 Netscape Directory Server，它管理并提供用户信息和用户所在的团体和部门等组织结构信息。而 Netscape Directory Server 的 HTTP 网关、Netscape Navigator 以及 Netscape Communicator 等都称为 LDAP 客户。网关使用目录服务器来查找、更新和添加用户信息。

1.2 LDAP 服务器组织目录的方式

由于 LDAP 要提供全球目录服务，因此其数据是以分层结构组织的：从根部向下分支为各个单独的条目。

在分层机构的最顶层，各个条目代表比较大的机构。在这些机构的下面可能是比较小的机构，而最终分层结构可能以单个的人或资源作为条目。

图 1.1 展示了 LDAP 目录服务中各个条目的一个分层结构。

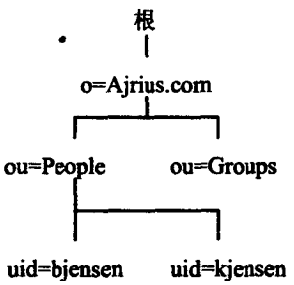

图 1.1 目录中条目的一个分层结构

每一个条目通过一个分辨名（distinguish name）相互区别。这个分辨名包括在这个层次上唯一确定这个条目的名称（例如，bjensen 和 kjensen 是不同的使用者 ID，它们用来区分在同一层次上的不同条目）和一个由名称组成的路径，沿这个路径可以从当前层返回到层级树的根部。

例如，下面是唯一标识 bjensen 这个条目的名字：

uid=bjensen，ou=People，o=Airius.com

这里，uid 表示条目的使用者 ID，ou 表示这个条目所在的机构单位，而 o 则表示这个条目所在的更高级机构。

图 1.2 展示了在目录层级机构中，分辨名如何确定每个独特的条目。

目录中保存的数据可以分散到几个 LDAP 服务器中。例如，一个在 Airius.com 的 LDAP 服务器可能包含表示 North American 组织单位和雇员的条目，而其它 LDAP 服务器可能包含表示 European 组织单位和雇员的条目。

一些 LDAP 服务器设置为可以把请示推荐到其他 LDAP 服务器。例如，如果在

Airius.com 的 LDAP服务器上收到一个雇员信息，而这个雇员在 Pacific Rim 分部，此时服务器可以把这个请求转交给 Pacific Rim 分部的 LDAP 服务器。通过这种方法，LDAP 服务器看起来就像是一个单独的目录信息源。即使一个 LDAP 服务器上并没有所需要的信息，它也会为用户查找另一个包含这个信息的服务器。

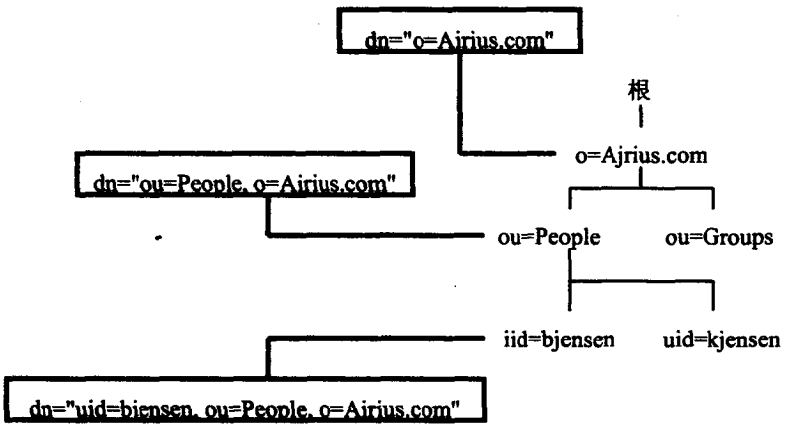

图 1.2　目录中一个分辨名

1.3　LDAP 客户端和服务器的工作方式

在 LDAP 的客户一服务器模型中，LDAP 服务器（如 Netscape Directory Server）使得 LDAP 可以访问关于个人、机构和资源的信息。LDAP 协议定义了客户端用来查找和更新目录的操作。下面列出了 LDAP 客户可以执行的部分操作：

在目录中查找并获取条目

- 向目录增加新的条目

更新目录中的条目

从目录中删除条目

- 给目录中的条目重命名

比如，要更新目录中的一个条目，LDAP 客户可以把这个条目分辨名以及更新的属性信息提交给 LDAP 服务器。LDAP 服务器使用条目的分辨名来找到这个条目并执行一个修改操作，这样就更新了目录中的条目。

要执行这些 LDAP 操作中的任何一个，LDAP 客户端都需要建立与 LDAP 服务器的连接。LDAP 协议指定使用 TCP/IP 的 389 号端口，尽管服务器可能运行在其它端口上。

LDAP 协议还定义了一个简单的鉴别方法。LDAP 服务器可以设置为拒绝对目录的访问。在 LDAP 客户端执行对 LDAP 服务器的操作之前，客户端必须通过向服务器提供一个分辨名和口令来通过鉴别。如果由分辨名识别的用户没有执行某些操作的许可，服务器就不执行这些操作。

.4 了解 LDAP 协议版本 3

许多 LDAP 服务器支持版本 2 LDAP 协议。这个版本在 RFC 1777 中有详细说明（可以在 http://www.ietf.org/rfc/rfc1777.txt 得到这个 RFC 的复件）。

最新提出 LDAP 协议的标准是版本 3，在 RFC 2251 中有详细说明（可以在 http://ww.ietf.org/rfc/rfc2251.txt 得到这个 RFC 的复件）。一些 LDAP 服务器支持这个更新版本的协议，例如 Netscape Directory Server 3.0 及以后的版本。

Netscape Directory SDK for Java 4.0支持这两个版本的协议。用这个 SDK建立的客户端可以和 LDAP 的版本 2、版本 3 服务器进行连接。

LDAP 协议的版本 3 包括以下一些新的特征：

- 可以定义控制来扩展 LDAP 操作的功能（在服务器或客户端都可以）。
- 可以请求服务器执行扩展操作（标准 LDAP 操作之外的操作）。
- 可以使用 Simple Authentication and Security Layer(简单鉴别和安全层次——SASL)机制来通过鉴别访问目录。
- 服务器具有 DSE（DSA-specific entries——DSA 描述条目，这里 DSA是一个目录服务器），它可以提供 LDAP 协议的版本信息，控制列表，扩展操作信息，服务器支持的 SASL 机制的信息，以及服务器的命名环境信息（描述目录树上由该服务器来管理的部分）。
- 服务器提供其概要给客户端（可以从根 DSE 得到一个目录服务器的概要）。
- 客户端和服务器都支持 UTF-8 格式的数据。客户端现在可以请求并得到用语言信息标记的数据。

.5 其 他 内 容

Netscape Directory Deployment Guide（Netscape Directory Deployment 指南）的第一章“Welcome to the Directory Server" 目录服务器简介）中对 LDAP 协议和目录服务器进行了更详细的介绍。

第 2 章　使用 Netscape Directory SDK for Java

本章描述了 Lightweight Directory Access Protocol(LDAP) 的 Java 类和 Netscape Directory SDK for Java。

下面是本章所包含的几个小节：

- 理解 LDAP 的 Java 类
- Netscape Directory SDK for Java 初步

2.1　理解 LDAP 的 Java 类

Netscape Directory SDK for Java 包括 LDAP Java 类，它可以用来建立 LDAP 客户端。LDAP Java 类允许编写与服务器相联系的 Java 程序和应用软件，并执行标准的 LDAP 操作（例如，可以用这些程序来查找或添加、更新、删除条目）。

这个类由以下一些组件构成：

- netscape.ldap 包括了主要的 LDAP Java 类，其中一些类允许用户与 LDAP服务器相连，对项目和属性进行操作，以及获取查询结果。
- netscape.ldap.beans 包括 LDAP JavaBeans。可以在一些开发环境中使用这些 Beans，比如 Sun 公司的 Bean Development Kit(BDK)。
- netscape.ldap.ber.stream 包括执行 Basic Encoding Rules(基本编码规则——BER)以转换句法的 LDAP Java 类。关于 BER的内容可以在 http://www.iso.ch/ 参考 ISO/IEC 8825。
- netscape.ldap.controls 包括执行特殊 LDAP 版本 3 控制的 LDAP Java 类。这些控制包括请求服务器端分类和持续搜索。
- netscape.ldap.util 包括一些工具类，例如对 LDIF 数据进行分析的类和一些过滤器，这些过滤器允许匹配某些规则的符号。
- com.netscape.sasl 包括接口以及用来使客户端在 SASL 机制下进行鉴别的类。
- com.netscape.jndi 包括 Netscape 的 LDAP Service Provider 和它所基于的类。这个 JNDI 工具将在第 15章进行详细介绍。

典型的，客户端同步执行 Netscape Directory SDK for Java 中的程序。所有 LDAP 操作在完成之前拥塞在一起（唯一不同的是查询操作，它可以在得到所有结果之前返回信息）。

同样，在需要与 LDAP 服务器进行低层次交互的环境下也提供异步的接口。在第 17 章中对异步接口有更完整的论述。

下面几章将介绍如何使用这些 LDAP Java 类。

2.2　Netscape Directory SDK for Java 初步

本节涉及到以下主题：

获取并安装 SDK
SDK 简介
准备使用 SDK
用 SDK 编写 Java 小程序
通过一个程序检测类的版本

- 使用 LDAP JavaBeans
  使用 JavaScript 中的类

2.2.1 获取并安装 SDK

可以从 Netscape 的 DevEdge 网站得到 Netscape Directory SDK for Java，网址如下：
http://developer.netscape.com/tech/directory/index.html
SDK 给 UNIX 环境提供一个以 GNU 格式压缩的档案文件，而提供给 Windows 的是一个自解压的可执行文件，在 Macintosh 下是一个 Stufflt 文档。
安装 SDK：
下载与所运行系统相匹配的文件
扩展这个文件（其中包括一些子目录）

2.2.2 SDK 简介

Netscape Directory SDK for Java 包括以下目录：

- beans
  这个目录下有 LDAP JavaBean 的类文件，这是 netscape.ldap.beans 插件的一部分。注意，这些类在 Netscape Communicator 中并没有包括。因此，如果要用这些类编写程序或软件，必须把这些类提供给用户。
  这个目录还包括一个 makejars.bat 文件和一个 makejars.sh 外壳程序。可以用这些来为 LDAP JavaBeans 创建 JAR 文件。
- examples
  这个目录包括一些 LDAP 软件的 Java 源代码，这些例子按下面的子目录分类：
  - java 包括一些添加条目和搜索条目的标准的 LDAP 操作的例子。还包括一些使用 LDAP 控制的例子。
  - java/beans 包括一些使用 LDAP JavaBeans 的例子。
    java/ldapfilt 包括一个通过 LDAP 过滤器类来使用 LDAP 过滤器配置文件的例子（注意，Netscape Communicator 中不包括 LDAP 过滤器类）。
    js 包括一个使用 LiveConnect 来创建并管理 JavaScript 中 LDAP Java 对象的例子（LiveConnect 是 Netscape 的一种技术，它可以在一个页面上实现各种元素之间的通信，包括 JavaScript，HTML，插件和 Java 小程序）。

  packages
  这个目录包括下面两个 JAR 文件：
  ldapjdk.jar——一这个 JAR 文件包括 netscape..ldap、netscape.ldap.controls、netscape.ldap.util 和 com.netscape.sasl 等组件中的类。

ldapfilt.jar——这个 JAR 文件包括 netscape.ldap.util 和 com.oroinc.text.regex 组件中的过滤器类。

com.oroinc.text.regex 是 ORO Java Software 中的 OROMatcherTM 规则表达式软件包。因此，如果要单独使用 OROMatcher 组件（不通过 Netscape Directory SDK for Java 类），必须从 ORO Java Software 获得使用 OROMatcher 组件的许可（也可以直接从 ORO 得到 OROMatcher文档）。

- tools

  这个目录包括的一些 Java类与 Netscape Direcotry Server 4.0和 Netscape Directory C SDK 中的命令行程序相似（注意，这些 Java 工具并不完全支持其它程序所用到的所有命令行参数）。

2.2.3 准备使用 SDK

在编译任何程序或应用软件之前，必须把下面的内容添加到 CLASSPATH 环境变量中：

packages/ldapjdk.jar 文件，这个文件中包括主要的 LDAP Java 类。

packages/ldapfilt.jar 文件，如果要用到任何 LDAP Java 过滤器类，就必须添加这个文件。

classes 目录，如果要用到任何 LDAP JavaBean 类，就需要添加这个目录。

2.2.4 用 SDK 编写 Java 小程序

在 Netscape Communicator 中，程序可以和任何主机上的服务器相连接，当然，其中不包括运行该程序的主机。这个功能是新签定的程序安全框架的一部分。

要利用这个功能，程序类（实现 LDAP 连接的类）必须是已注册的。程序类在与其他服务器连接之前需要获得某些特殊的权利。

本节下面的部分总结了使程序和其它 LDAP 服务器相连接所要采取的步骤。

1. 从所在机构的认证部门（如果所在机构内部发放证明）获得证明，也可以从第三方的认证部门，如 RSA 或 ATT。

   用户应该在 Communicator 证书数据库中具有认证部门的证明。

2. 用自己的类创建 JAR文件并对其进行注册。

   可以用 JAR 文件管理工具来实现这一步，这个工具在下面的站点可以找到：

   http://developer.netscape.com/software/signedobj/

   在下面的站点可以找到关于 Netscape Object Signing 技术的附加文件：

   http://developer.netscape.com/docs/manuals/signedobj/

3. 把下面一行添加到线程的程序代码中需要调用 LDAPConnection.cnnect 的位置。

   ```
   PrivilegeManager.enablePrivilege("Universalconnect");
   ```

   在代码的这个位置，程序的用户会得到一个提示对话框，其中对注册类的作者进行确认并请求得到访问 LDAP 服务器的权利。用户可以只允许这一次的访问，也可以允许永久访问。

如果希望在不使用对象注册和认证的情况下来测试程序，使之与其它主机进行连接（除去运行该程序的主机），例如想在等待发放认证的时候对程序进行测试，可以把

Netscape Communicator 设置为忽略这次测试。

首先，退出 Communicator(或确认 Communicator没有运行)。然后在 prefs.js 文件中添加以下代码：

```
user_pref("signed.applets.codebase_principal_support",true);
```

2.2.5 通过一个程序检测类的版本

Netscape Communicator 4.0 和更新的版本都包括一个 LDAP Java 类的复件。不同的 Communicator 版本具有不同版本的类。

下面的表格列出了到目前为止已经发布的 LDAP Java类的几个不同版本。

表 2.1 最近发布的几个 LDAP Java 类版本

版本	说明
3.03	和 Netscape Directory SDK for Java 3.0 一起发布
3.1	和 Netscape Directory SDK for Java 3.1 一起发布
4.0	在 http://developer.netscape.com/software/ldap 可以找到

LDAP Java类的 3.03 版以及更高的版本都包括诸如 LDAPCache 和 LDAPSchema 等新类。如果要在程序中使用这些类，就必须检测浏览器中所提供的 LDAP Java 类的版本。

要得到版本信息，需要调用 LDAPConnection.getProperty 方法，并传入常量 LDAPConnection.LDAP PROPERTY SDK 。版本号是一个 Float类型的数。例如：

```
...
Float sdkVersion=(Float)myconn.getProperty(myConn.LDAP_PROPERTY_SDK);
System.out.println("LDAP Java Classes version:"+sdkVersion);
...
```

如果需要了解当前版本和以前版本中的 LDAP Java 类之间有何不同,可以在下面的站点查看发布说明：

http://developer.netscape.com/tech/directory/index.html?content=java40relnotes.html

2.2.6 使用 LDAP JavaBeans

Netscape Driectory SDK for Java 包括一组 LDAP JavaBeans，可以在诸如 Sun Microsystem 的 BeanBox 或 Symantec 的 Visual Cafe 设计环境下使用它们。

这些 Bean 是 netscape.ldap.beans 组件的一部分，在 beans 目录下可以找到其类文件。

下面列出了 Netscape Directory SDK for Java 中所包括的 JavaBeans：

LDAPGetEntries Bean 允许搜索目录并得到一个查找到的 DN 数组。可以用这个 Bean 的这些属性来指定搜索的标准。GetEntries 方法执行搜索并设置 Result 属性为所查找到的 DN 数组。

LDAPGetProperty Bean 允许在目录中查找一个条目，并获取该条目中一个指定属性的值。可以用这个 Bean 属性来指定搜索的标准。GetProperty 方法执行搜索并设置 Result 属性为一个数组，其中有指定属性的字符串值。

LDAPIsMember Bean 确定用户是否是一个团体的成员（用户和团体可以用这个 Bean 的属性类确定）。IsMember 方法把 Result 属性设置为字符串“Y ”或“N”来指明用户是否是一个成员。

- LDAPSimpleAuth Bean 对 LDAP 服务器进行验证。Authenticate 方法执行验证并把 Result 属性设置为字符串“Y”或“N”来指明验证是否成功。

Netscape.ldap.beans 组件还包括下面一些类：

- LDAPBasePropertySupport 类是派生出其它 Bean 类的基类。其它 Bean 类继承这个类定义的存取程序。
- DisplayString 类派生出 jav.awt.TestArea 类，它的作用是显示其它一些 Bean 的结果。

在某个开发环境中使用 Bean 之前，必须确认已经把 CLASSPATH 环境变量设置为包括 LDAP Java 类。例如：

- 如果用 Sun Microsystems BeanBox 工具，需要确认在 run.bat 文件、make 描述文件、当前处理程序或控制台窗口中， CLASSPATH 环境变量包括 ldapjdk.jar 文件的路径。例如：

  set CLASSPATH=classes;D:\netscape\ldapjdk\packages\ldapjdk.jar
- 如果用 Symantec Visual Cafe 2.x， 需要确认在 VisualCafe\bin\sc.ini 文件中，CLASSPATH 条目包括 ldapjdk.jar 文件的路径。例如：

  CLASSPATH=.;...<other_paths>;
  D:\NETSCAPE\LDAPJDK\PACKAGES\LDAPJDK.JAR

下面，设置开发环境使之包括 Netscape Directory SDK for Java 的 beans 目录下的 JAR 文件。例如：

- 如果使用 BeanBox 工具，把 JAR 文件复制到 BDK/jars 目录下。则当运行 BeanBox 时，就会自动载入 LDAP JavaBeans。
- 如果使用 Visual Cafe，选择 View|Component Library 菜单命令，弹出 Component Library 窗口。在确认当前已经打开一个工程之后，选择 Insert|Component Into Library 菜单命令并选择一个包括在 LDAP JavaBean 中的 JAR 文件。这样就把该 Bean 加到了自己的组件库。

注意， LDAP JavaBean 是不可见 Bean，而不是 UI 组件。

这些 Bean 具有 Debug 属性，如果希望打印调试信息就可以设置这个属性。这样，调试信息就打印到标准输出中了。因此，如果在 BeanBox 或程序阅览器测试程序，调试信息就输出到控制台窗口或外壳程序窗口。

2.2.7 使用 JavaScript 中的类

使用 Netscape Communicator 的 LiveConnect 功能，可以利用来自于 HTML 页中 JavaScript 代码的 LDAP Java 类（LiveConnect 使得 JavaScript 和 Java 程序之间的以及 JavaScript 和页面上载入的插件之间可以进行信息传递）。

examples/js 目录下的 HTML 文件中的 JavaScript 例子展示了如何实现这一功能。要更多地了解 LiveConnect，可以查阅 Netscape JavaScript Guide 和 Tech Note TN-JSCR-03-9707 中的有关章节。这两本书的电子版可以分别在下面两个站点找到：

http://developer.netscape.com/docs/manuals/js/client/jsguide/index.htm
http://developer.netscape.com/docs/technote/javascript/liveconnect/liveconnect_rh.html

第 3 章　快速入门

本章将介绍一个用 Netscape Directory SDK for Java 编写的简单的 LDAP 客户端。分为以下两节：

- 了解该客户端样本
- 样本代码

3.1　了解该客户端样本

下面是一个程序的源代码，其功能是获取 Barbara Jensen 的全名（“cn”），姓（“sn”）、Email 地址（“mail”）和电话号码（“telephoneNumber”）。可以在 examples/java 目录下的 getAttrs.java 文件中找到这个程序。

这个例子主要实现以下几点：

1. 创建一个新的 LDAPConnection 对象，这个对象代表和 LDAP 服务器的连接。
2. 连接到服务器。
3. 执行查询操作来获得一个单个条目，这个条目由其 DN 确定。要实现这一点，需要设置查询标准，这样就可以达到以下目的：
   - 分辨名（查询的起始点）为条目“uid=bjensen，ou=People，o=Airius.com”
   - 查询的覆盖范围为 LDAPConnection.SCOPE SASE，这样就只包括基本的 DN。
   - 查询过滤器为“(objectclass=*)”，这样就查找所有相匹配的项。由于覆盖范围把查询限制为单个的条目，因此查询过滤器就没有必要使用。

   （注意，执行对单个条目的查询和调用 LDAPConnection.read 方法作用相同）
   - 字符串数组 attrName 中包含查询所返回的属性。
   - 把 attrsOnly 设置为 false，这意味着将返回所指定属性的名称和值。
4. 重复列举的查询，就可以得到并打印 cn，sn，mail 和 telephone Number 这些属性的值。这个重复操作也允许客户端获取一个属性的多个实例（例如，两个电话号码）。
5. 断开到服务器的连接。

在编译这个样本客户端之前，需要把“localhost”和“389 ’替换为主机名和所用 LDAP 服务器的端口号。同时，必须确认 packages/ldapjdk.jar 文件包含在 CLASSPATH 中。

3.2　样 本 代 码

```
import netscape.*;
import java.util.*;
public class GetAttrs{
    public static void main(String[] args)
    {
        LDAPConnection ld=null;
```

```
LDAPEntry findEntry=null;
    int status=-1;
    try{
      ld=new LDAPConnection();
      /*连接到服务器*/
      String MY_HOST="localhost";
      int MY_PORT=389;
      ld.connect(MY_HOST,MY_PORT);
      String ENTRYDN="uid=bjensen,ou=People,o=Airius.com";
      String[] attrNames={
        "cn",/*获取正式名称（全名）*/
        "sn",/*获取别名（姓）*/
        "mail",/*获取 email 地址*/
        "telephonenumber"};/*获取电话号码*/
      LDAPSearchResults res=ld.search(ENTRYDN,
            LDAPConnection.SCOPE_BASE,"objectclass=*",
            attrNames,false);
      /*在结束之前按结果循环：只有一个！ */
      while(res.hasMoreElements()){
        /*下一个目录条目，最多只有一个*/
        LDAPEntry findEntry=null;
        try{
            findEntry=res.next();
        /*如果下一个结果是一个查询参考，
          打印参考中的 LDAP URL。*/
        }catch(LDAPReferralException e){
          System.out.println("Search reference:");
          LdAPUrl refUrls[]=elgetURLs();
          for(int i=0;i<refUrls.length;i++){
            System.out.println("\t"+refUrls[i].getURL());
          }
          continue;
        }catch(LDAPExcepton e){
          System.out.println("Error:"+e.toString());
          continue;
        }
        /*获取条目的属性*/
        LDAPAttributeSet findAttrs=findEntry.getAttributeSet();
        Enumeration enumAttrs=findAttrs.getAttributes();
        /*Loop on attributes*/
        while(enumAttrs.hasMoreElements()){
          LDAPAttribute anAttr=
            (LDAPAttribute)enumAttrs.nextElement();
          String attrName=anAttr.getName();
          if(attrName.equals("cn"))
            System.out.println("Full name:");
          else if(attrName.euals("sn"))
            System.out.println("Last name(surname):");
          else if(attrName.equals("mail"))
            System.out.println("Email address:");
```

```
                else if(attrName.equals("telephonenumber"))
                  System.out.println("Telephone number:");
                /*按该属性的值循环 */
                Enumeration enumVals=anAttr.getStringValues();
                while(enumVals.hasMoreEnements()){
                    String aVal=(String)enumVals.nextElement();
                    System.out.println("\t"+aVal);
                }
            }
        }
    }
    catch(LDAPException e){
      System.out.println("Error: "+e.toString());
    }
    /*完成任务，断开连接*/
    if ( (ld!=null)&&ld.isConnected()){
      try {
          ld.disconnect();
      } catch (LDAPException e){
        System.out.println("Error: "+e.toString());
    }
    System.exit(status);
  }
}
```

第二部分

用 Netscape Directory SDK for Java 编写客户端

第4章 编写 LDAP 客户端

本章详细介绍编写 LDAP 客户端的一般过程，其中涉及到的步骤包括连接到 LDAP 服务器、验证、请求操作和断开到服务器的连接。

第5章 使用 LDAP Java 类

本章涉及到一些在编写 LDAP 客户端时经常用到的一般 LDAP Java 类。

第6章 查询目录

本章介绍如何使用 LDAP Java 类来查询目录并获取条目，同时详细介绍了如何从一个条目中得到属性和属性值。

第7章 使用过滤器配置文件

本章详细说明了如何同时采用 API 函数和过滤器配置文件进行工作。使用过滤器配置文件可以简化为一个查询请求选择合适的过滤器的过程。

第8章 添加、更新和删除条目

本章详细介绍如何使用 LDAP Java 类在目录中对条目进行添加、修改、删除和重命名。

第9章 比较条目中的值

本章介绍了如何把一个条目中的某个属性值与一个指定值做比较。

第10章 使用 LDAP URL

本章详细介绍了什么是 LDAP URL，并讲解如何用 LDAP URL 来搜索目录并从中获取数据。

第 4 章　编写 LDAP 客户端

本章详细介绍了编写 LDAP 客户端的一般过程，其中涉及到的步骤包括连接到 LDAP 服务器、绑定、请求操作和断开到服务器的连接。

这一章包括以下几节：

- 概述：设计一个 LDAP 客户端
- 创建连接并设置优先权
- 连接到 LDAP 服务器
- LDAP服务器的认证和绑定
- 执行 LDAP 操作
- 断开到服务器的连接

第 5章也涉及到用 LDAP Java 类来编写 LDAP 客户端的一些重要内容。

4.1　概述：设计一个 LDAP客户端

使用 Netscape Directory SDK for Java ，可以编写和 LDAP 服务器相互作用的程序或应用软件。下面的步骤概要描述了和 LDAP 服务器通信的典型过程。按照这些步骤就可以编写自己的 LDAP 客户端：

1. 创建一个新的 LDAPConnection 对象，为所有 LDAP 操作设置自己要应用的优先权详见 4.2节。
2. 连接到 LDAP 服务器详见 4.3节。
3. 如果需要，就绑定到 LDAP 服务器。如果要用到一些 LDAP 3 版的特征（比如控制和扩展操作），必须确认该客户端所支持的 LDAP 版本详见 4.4 节。
4. 执行操作 （例如，查询目录或修改目录中的条目详见 4.5 节。
5. 完成之后断开到 LDAP 服务器的连接详见 4.6节。

下面是一个简单的 LDAP 客户端的例子，这个程序按照上面列出的步骤来查询一个目录。客户端连接到运行在本地机器上的 LDAP 服务器，其端口号为 389，以 DN “uid=bjensen，ou=People，o=Airius.com”进行认证。这里要查找目录中姓为“Jensen”（“sn=Jensen”）的条目，并且打印所有符合条件的条目 DN。

```
import netscape.ldap.*;
import java.util.*;
public class SimpleExample{
    public static void main(String[] args)
    {
        /*第 1 步：创建一个新的连接 */
        LDAPConnection ld=new LDAPConnection();

        try{
            /*第 2 步：连接到一个 LDAP 服务器*/
            ld.connect("localhost",LDAPv2.DEFAULT_PORT);
```