LDAP 使用手册

一、 LDAP 介绍

LDAP 是轻量级目录访问协议的简称(Lightweight Directory Access Protocol).用于访问目录服务。它是 X.500 目录访问协议的移植，但是简化了实现方法。

二、 目录服务与关系数据库之间的区别

a) 目录查询操作比关系数据库有更高的效率，但是更新效率比关系数据库低
b) 目录不支持关系数据库那样的复杂查询，比如两个表的连接。
c) 目录不支持多操作的事物完整性，没有方式确认一些操作是全部成功还是全部失败
d) 目录能够能好和更灵活的支持子查询和匹配查询
e) 目录协议更适合应用于广域网，比如因特网或者大型公司的网络
f) 目录的管理，配置，和调试比关系型数据库更简单
g) 在使用关系数据库之前，必须首先定义表结构(模式)才可以进行操作。而目录中所使用的模式是由 LDAP 定义好的一系列类组成的。对于目录中的每条记录中必须属于其中的一个类或者多个类。这些类定义了该记录中可以存储的信息。
h) 目录以对象的形式存储数据。信息被组织成树型结构。
i) 目录服务支持分布式存储结构，容易实现数据的扩展，能满足大容量存储的要求。

三、 LDAP 的优点

1:可以存储在其它条件下很难存储的管理信息
2:数据安全可靠，访问控制粒度细腻。
3:LDAP 是一个标准的，开放的协议，具有平台无关性。
4:数据分布广，规模可灵活扩充。
5:LDAP 目录服务器可以使任何一种开放源代码或商用的 LDAP 目录服务器。

四、 LDAP 模型

LDAP 模型是从 X.500 协议中继承过来的。是 LDAP 的一个组成部分，用于指导客户如何使用目录服务

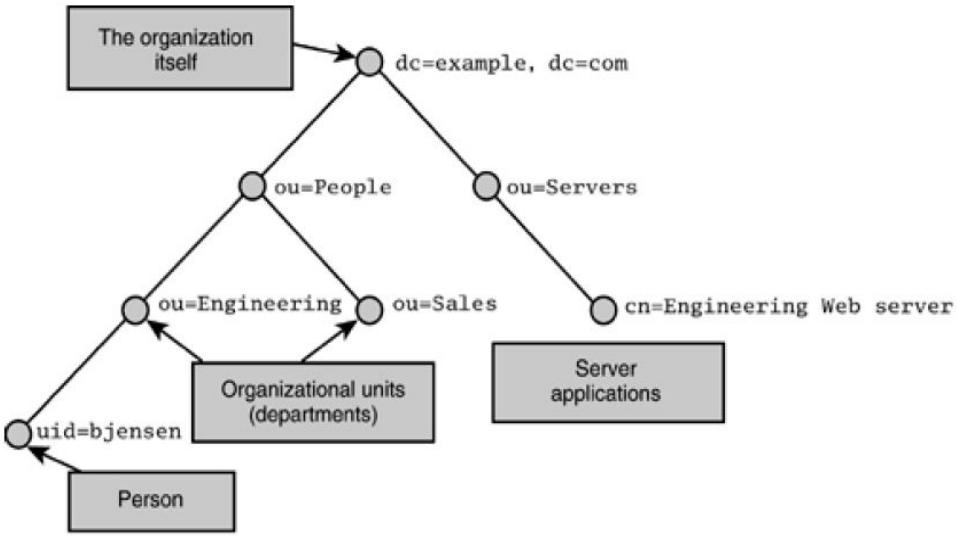
LDAP 定义了四个模型，包括信息模型，命名模型，功能模型，安全模型。

1.LDAP 信息模型(LDAP information model)

LDAP 信息模型用于描述 LDAP 中信息的表达方式。

LDAP 信息模型包含三部分 Entries Attributes Values

Entry:Directry 中最基本的信息单元，Entry 中所包含的信息描述了现实世界中的一个真实的对象，在目录系统中它可以理解为，目录树中的一个节点 。在目录中添加一个 Entry 时，该 Entry 必须属于一个或多个 object class ,每一个 object class 规定了该 Entry 中必须要包含的属性，以及允许使用的属性。Entry 所属的类型由属性 objectclass 规定。每一个 Entry 都有一个 DN(distinguished name) 用于唯一的标志 Entry 在 directory 中的位置。如下图所示：



根节点 DN 的命名有多种方法，其中之一就是域名命名法。例如：我们要以公司的网址作为公司目录树的根节点。如 sohu.com 那么根节点的 DN 应该为 DN:dc=sohu，dc=com

上图中根节点的 DN ：dc=example,dc=com 而该根节点有两个子节点，ou=people，和 ou=servers。

People 节点的 DN：ou=People,dc=example,dc=com

RDN: 是目录树中节点的相对分辨名。如：People 节点的 DN: ou=People,dc=example,dc=com 而该节点的 RDN：ou=People

Attribute: 每个 Entry 都是由许多 Attribute 组成的。每一个属性(Attribute)描述的是对象的一个特征。每一个属性(Attribute)由一个类型(type)和一个或多个值组成(Value) 如下图所示：

Attribute type	Attribute values
cn:	Barbara Jensen Babs Jensen
sn:	Jensen
telephoneNumber:	+1 408 555 1212
mail:	babs@example.com

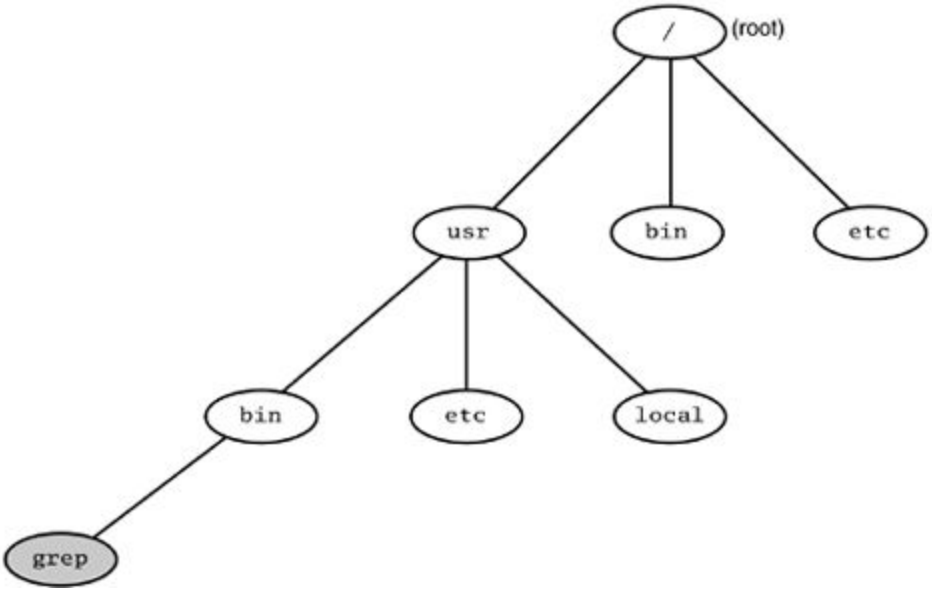
2.LDAP 命名模型(LDAP Naming Model)

LDAP 命名模型定义了如何在目录系统中组织数据以及如何从目录系统中查找数据

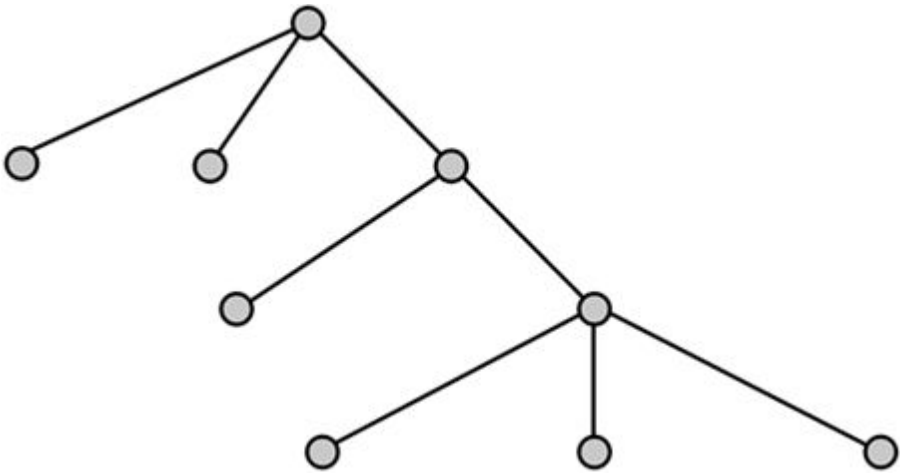
LDAP 命名模型指定将 Entry 按类似倒立的树形结构进行规划。非常类似于 UNIX 系统得文件系

统如下图所示：

Unix File System



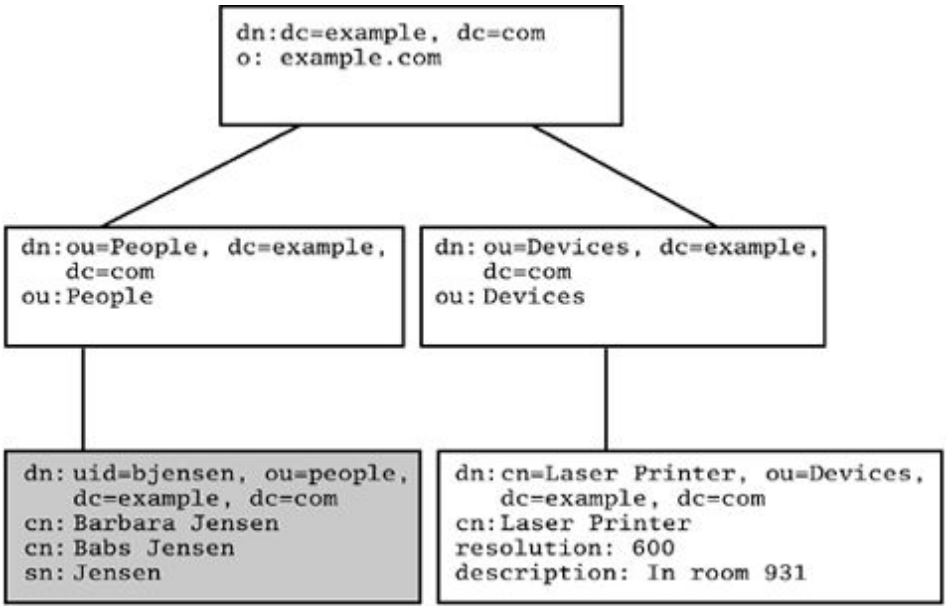
A Directory Tree



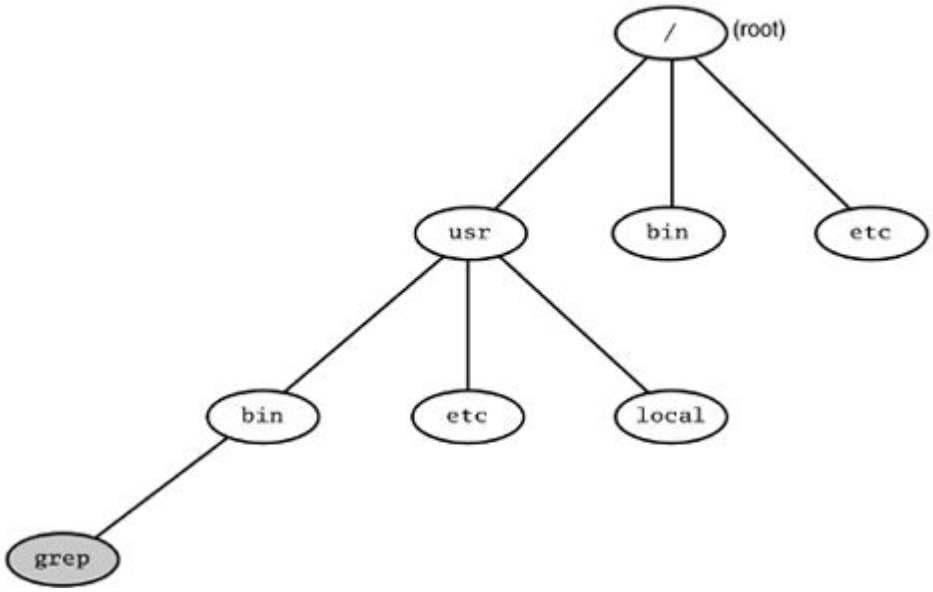
LDAP 目录结构与 Unix 系统的文件系统主要有三点不同

1. UNIX 文件系统有一个根路径，作为访问所有文件和目录的入口。而 LDAP 目录结构中的 root Entry 只是一个特殊的 Entry，它包含了目录服务器的配置信息，通常情况下，并不用来存储信息
2. 在 LDAP 目录中任何一个节点都可以包含信息，同时也可以是一个容器，也就是说任何一个 LDAP Entry 都可以有子节点。而 UNIX 文件系统中的节点要么是一个文件，要么是一个目录。而不能同时是这两种情况。只有目录才可以拥有子节点。下图表示了 LDAP 是一个典型的目录结构

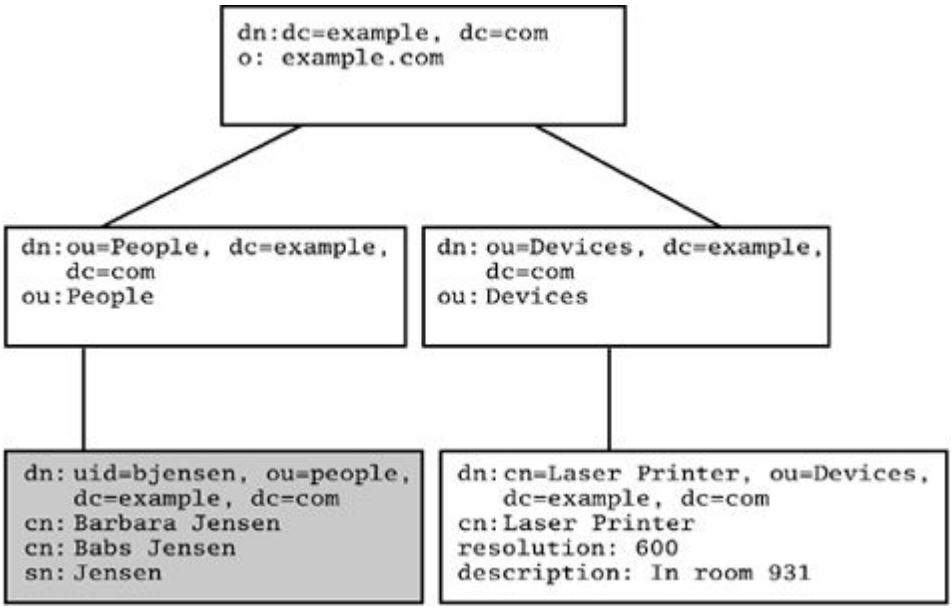
LDAP Directory



3. UNIX 文件系统目录结构与 LDAP 目录的第三个区别在于他们的每一个节点的命名不同。LDAP 目录中节点的命名和 UNIX 文件系统目录结构中的节点的命名是相反的。



上图示 UNIX 文件系统结构，如果要定位到 grep 节点的话，命名如下
/user/bin/grep



上图是一个典型的目录结构

第一个节点 DN 命名为：dn:dc=example,dc=com

第二个节点 DN 命名为：dn:ou=People,dc=example,dc=com

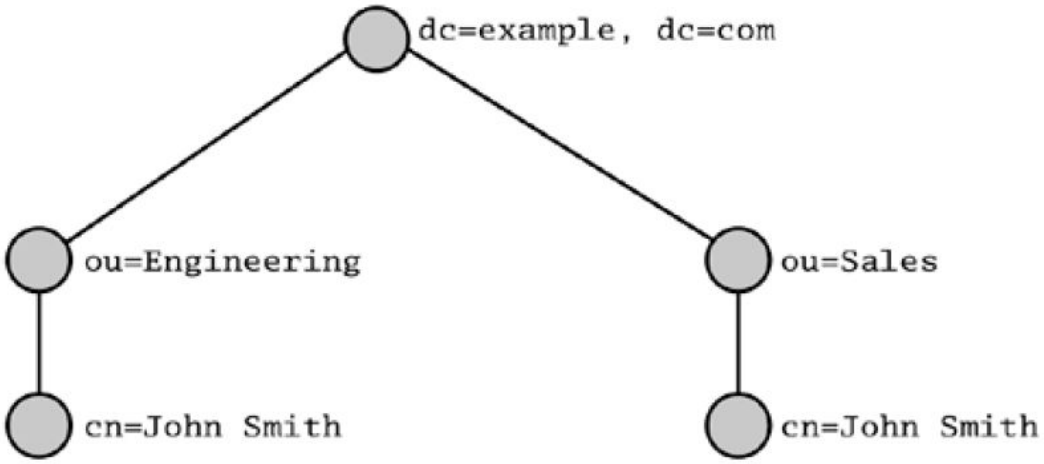
第三个节点 DN 命名为：dn:uid=bjensen,ou=people,dc=example,dc=com

我们说每一个 Entry 的 DN 是唯一的，就是因为这种数形结构决定了，从根节点到其它任何一个节点的路径是唯一的。

RDN：在 DN 中最左边的内容称为相对域名。如 ou=People,dc=example,dc=com

其 RDN 为 ou=People

对于共享同一个父节点的所有节点的 RDN 必须是唯一的。如果不属于同一个节点则节点的 RDN 可以相同。



特殊字符：

以下字符如果出现在 Entry 中的属性值，必须进行转义

 # --------\#

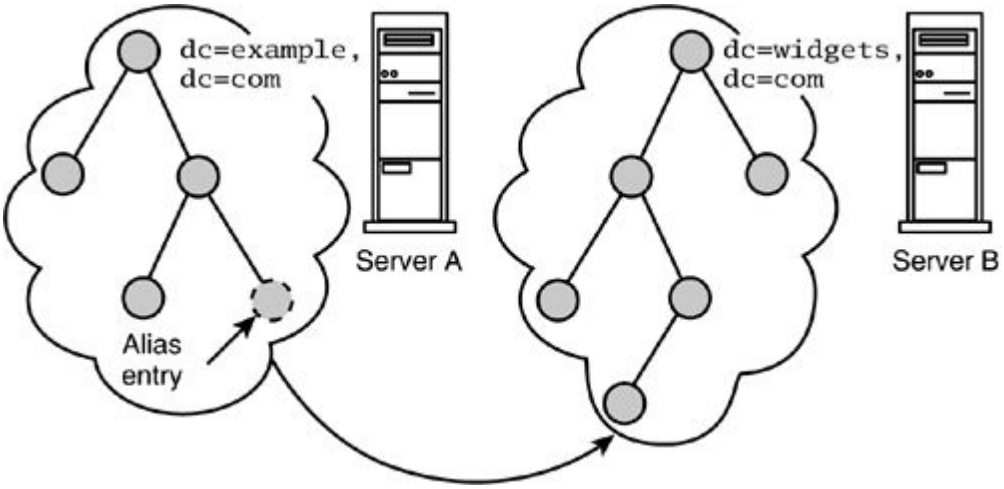
， -------\，

+---------\+
“---------\”
\--------\\
< ----- \<
> ----- \>
; -------- \;

如：o=United Widgets\, Ltd., c=GB

别名

在 LDAP 中可以定义一个别名 Entry，指向另外一个 Entry。如下图所示



如何创建别名 Entry

要创建别名 Entry，该 Entry 的 object class 必须是 alias。而且其属性 aliasedObjectName 的值必须是该 Entry 所指向的 Entry 的 DN。不过一般情况下应该避免使用别名 Entry。会影响性能。而且如果被引用的 Entry 被删除的话，该 Entry 就会指向一个错误的结果。

LDAP URL

由于使用 Alias Entry 有许多缺点，可以使用 LDAP URL 或 referral 代替 Alias Entry。

3.LDAP 功能模型(LDAP Functional Model)

LDAP 功能模型描述了 LDAP 协议可以采用的相关操作，来访问存储在目录树中的数据。

LDAP 功能模型包含一系列的操作，这些操作被分为三组。

1. 更新操作 包括添加，删除，重命名，修改 Entry
2. Interrogation Operation 用于数据的查询
3. 认证和控制 (bind unbind abandon)

Interrogation Operation

1. The LDAP Search Operation

   该操作需要八个参数

   a. base object

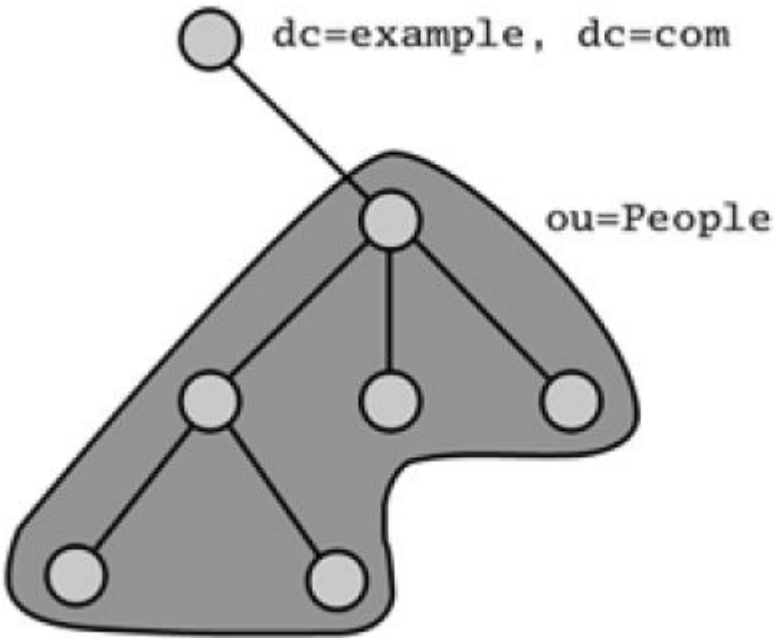
      也可以表示为 DN。表明你想要查询 direcoty 中树的顶点。

   b. search scope

      DN 与 search scope 两个参数限定了要搜索数据的范围

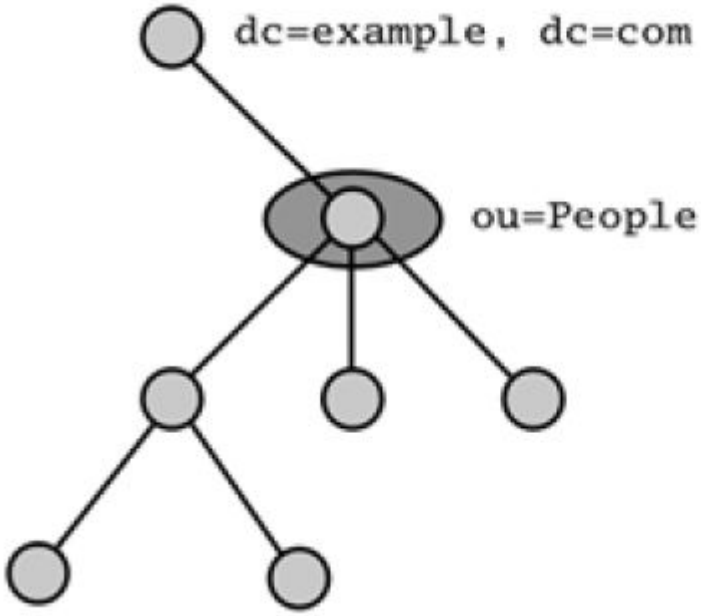
      共有三个 scope

      Sub 搜索范围是包含顶节点在内的一棵子树 如下图



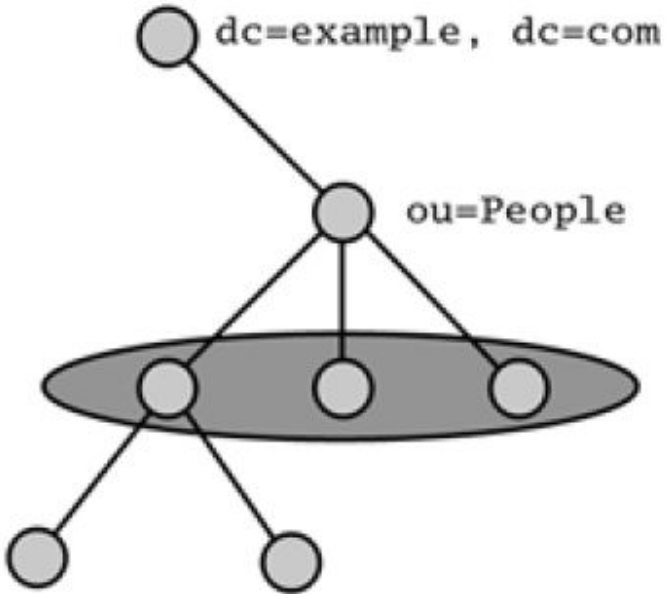
其中 DN =”ou=People,dc=example,dc=com”

Base 搜索范围只包含一个节点 如下图



其中 DN =”ou=People,dc=example,dc=com”

Onelevel 其搜索范围是 DN 所表示的节点下的直接子节点。如下图



其中 DN =”ou=People,dc=example,dc=com”

c. alias

d. size limit

表示返回的符合条件的 Entry 的数目，0 表示返回所有符合条件的 Entry。目录服务器端返回一个 LDAP_SIZELIMIT_ EXCEEDED。

e. time limit

表示搜索一次所需要的时间，超过时间将停止搜索。服务器端返回一个 LDAP_TIMELIMIT_EXCEEDED

f. attribute-only

该属性是一个 boolean 值，如果为 true ，表示服务器端之返回所搜索的 Entry 的属性名称，不返回属性值。

g. filter

通过该属性可以更精确的搜索结果。就像 SQL 语句中的条件查询。

Filter 分类

1.(sn=smith) 匹配属性 sn 的值包含 smith 的 Entry

2 (sn=smith*) 匹配属性 sn 的值以 smith 开始的所有 Entry 如 smithers，smithsonain 等。其中“*”表示通配符，代表任意字符。

3 (sn~=jensen) 匹配属性 sn 的值听起来像 jensen 的 Entry。不同的目录服务器，有不同的实现方法。

4 (age>21)或者(!(age<=21)) 匹配属性 age 的值大于 21 的 Entry
如果是字符的话，如(sn<=Smith)则按字典顺序进行比较。

5 (telephoneNumber=*) 匹配所有属性 telephoneNumber 的值不为空的 Entry

6 (&(sn=smith)(age>21))匹配属性 sn 的值包含 smith 而且属性 age 的值大于 21 的 Entry

(|(sn=smith)(age>21))匹配属性 sn 的值包含 smith 或者属性 age 的值大于 21 的 Entry

(&(mail=*)(!(telephoneNumber=*)))匹配属性 mail 的值不为空，而且属性 telephoneNumber 为空的 Entry

7 如果属性的值包含以下五个特殊字符的话必须进行转义

如：(cn=A*Star) 则必须改为(cn=A\2AStar)

Character Sequence	Decimal Value	Hex Value	Escape Sequence
* (asterisk)	42	0x2A	\2A
((left parenthesis)	40	0x28	\28
) (right parenthesis)	41	0x29	\29
\ (backslash)	92	0x5c	\5C
NUL (the null byte)	0	0x00	\00

h. return attributes

该属性表示客户的搜索结果中需要返回的和用户相关的属性列表，如果为空表示返回所有属性。

Attribute List	Attributes Returned
cn, sn, givenName	cn, sn, and givenName only
*	All user attributes
1.1	No attributes
modifiersName	modifiersName only (an operational attribute)
*, modifiersName	All user attributes plus modifiersName

4.LDAP 安全模型

安全模型的作用：是提供一个框架，保护目录中的信息不被非法访问。

LDAP 中的安全模型主要通过身份认证、安全通道和访问控制(ACL)来实现

LDAP 是一个面向连接的协议，在能够对 LDAP 目录进行任何操作之前，LDAP 客户端必须获得一个到 LDAP 服务端的一个连接，在这个过程中需要对 LDAP 客户端的身份进行验证，这一过程可以理解为用户绑定。

LDAPV2 只支持简单的密码验证。

LDAPV3 实现了 SASL 安全框架，SASL 为多种验证协议提供了一种标准的验证方法，对于不同的验证系统，可以实现特定的 SASL 机制。SASL 机制代表了一种验证协议。

在用户通过验证之后，可以为该用户分配附加的权限，比如一些用户只能查看特定的 Entry，而不能修改。一些用户可以查看并且修改所有的 Entry 等。这一过程可以理解为访问控制。

五、 LDIF

LDIF 通过一个文本文件，用来描述目录数据，可以将目录服务器中的数据导出到一个 LDIF 文件中，并且可以将 LDIF 文件中的数据导入到另一个目录服务器。即使这两个目录服务器内部使用的是不同的数据库格式。

有两种类型的 LDIF 文件，第一种用来描述 Directory 目录数据的，第二种包含更新语句，用于更新现有的 Directory 条目数据。

第一种文件，内容包含两部分：第一部分是 DN，第二部分是一系列的属性-值对

如下图：

```
dn: uid=bjensen, ou=people, dc=example, dc=com

objectclass: top

objectclass: person

objectclass: organizationalPerson

objectclass: inetOrgPerson

cn: Barbara Jensen

cn: Babs Jensen

givenName: Barbara

sn: Jensen

uid: bjensen

mail: bjensen@example.com

telephoneNumber: +1 408 555 1212

description: Manager, Switching Products Division
```

以上的 LDIF 文件只包含了一个 Directory Entry。可以在一个 LDIF 文件中包含多个 Entry

第二种文件，包含更新语句。第一行同样是 DN。第二行是更新类型，后边是要更新的属性及值。当然也可以用来添加新的 Entry。

如下图：添加一条 Entry(注意第二行:changetype：add)

```
dn: uid=bjensen, ou=people, dc=example, dc=com

changetype: add

objectclass: top

objectclass: person

objectclass: organizationalPerson

objectclass: inetOrgPerson

cn: Barbara Jensen

cn: Babs Jensen

givenName: Barbara

sn: Jensen

uid: bjensen

mail: bjensen@example.com

telephoneNumber: +1 408 555 1212
```

如下图：删除一条 Entry

```
dn: uid=bjensen, ou=people, dc=example, dc=com

changetype: delete
```

如下图：修改一条 Entry

语法如下：

```
dn: dn of entry to be modified
changetype: modify
modifytype: attribute type
[attribute type: attribute value]
```

添加属性示例

```
dn: uid=bjensen, ou=people, dc=example, dc=com

changetype: modify

add: mail

mail: bjensen@example.com
```

删除属性示例

```
dn: uid=bjensen, ou=people, dc=example, dc=com

changetype: modify

delete: telephoneNumber

telephoneNumber: +1 216 555 1212
```

或者

```
dn: uid=bjensen, ou=people, dc=example, dc=com

changetype: modify

delete: telephoneNumber
```

修改属性值

```
dn: uid=bjensen, ou=people, dc=example, dc=com

changetype: modify

replace: telephoneNumber

telephoneNumber: +1 216 555 1212

telephoneNumber: +1 405 555 1212
```

也可以将多条更新语句放在一个文件里，各语句之间用“-”分开

```
dn: uid=bjensen, ou=people, dc=example, dc=com

changetype: modify

add: mail

mail: bjensen@example.com

-

delete: telephoneNumber

telephoneNumber: +1 216 555 1212

-

delete: description

-

replace: givenName

givenName: Barbara
```

重新命名 Entry 或移动 Entry

语法：

dn: Ｅｎｔｒｙ的ＤＮ名称

changetype: moddn //changetype类型必须为ｍｏｄｄｎ

[newsuperior:如果要移动一条Ｅｎｔｒｙ则该项表示一个新的节点的ＤＮ]

[deleteoldrdn: (0 | 1)该项表示是否要删除修改以前的ＲＤＮ 0 不删除]

[newrdn:Ｅｎｔｒｙ的新ＲＤＮ]

由于一条 Entry 的 DN 是由该 Entry 的 RDN 和它的父节点的 DN 组成的。

一条 Entry 的 RDN 是该 Entry 中的一个属性。就像关系数据库中，主键值可以唯一区区分一条记录一样，在同一个节点下的所有 Entry 中的 RDN 必须由唯一标识该 Entry 属性组成。

所以重新命名 Entry，也就是重新命名该 Entry 的 RDN。

示例如下：

```
dn: uid=bjensen, ou=People, dc=example, dc=com

changetype: moddn

newrdn: uid=babsj

deleteoldrdn: 0
```

执行以上语句后该Ｅｎｔｒｙ　如下所示：

```
dn: uid=babsj, ou=People, dc=example, dc=com
uid: babsj
uid: bjensen
```

移动一条Ｅｎｔｒｙ示例如下：

```
dn: uid=bjensen, ou=People, dc=example, dc=com

changetype: moddn

newsuperior: ou=Terminated Employees, dc=example, dc=com
```

六、　LDAP 模式（schema）

如果你使用过关系型数据库，那么对模式应该不会陌生。关系数据库系统都是通过表格的形式进行数据存储的。在这之前，我们首先要定义表结构，也即是模式。表结构由一些字段组成，每个字段都有一个类型，以及一些约束条件。这就规定了我们可以存储的信息。

上面我们介绍过 LDAP 目录服务器中存储的信息是被组织成树型结构进行存储的。和存储信息之前也要定义模式，不过，与关系型数据库系统不同的是，作为 LDAP 目录服务器的用户而言，一般不需要自己定义模式，所有实现 LDAP 协议的目录服务器，都已经定义好了许多模式，这些模式可以解决我们大部分的信息存储的问题。

LDAP 模式是由以下一些元素组成的

1：Attribute types 属性类型，也就是属性名称。我们已经介绍过，每个属性名称其实也代表着一种属性类型。表示该属性可以存储什么样的信息。

2：Attribute syntaxes 属性语法，该元素表示每个属性名称所存储的信息如何组织。

3：匹配规则，每一个属性都有匹配规则，用于数据的比较。

4：object classes，对象类.上面已经介绍过，每个 Entry 都必须至少属于一个 object class。规定了该 Entry 可以存储那些属性。

下面介绍一下 LDAP 协议中定义的一些常用属性极其含义（具体信息看 RFC 2252 文档）

属性	中文名称	描述
c	国家名称	值为两位国家代码 如：中国：CN 美国：US
cn	通用名称	
dc	域名组件	如：sohu.com dc=sohu,dc=com
co	国家名称	国家的全名
gn	gavenName	
homephone	家庭电话号码	
mail	邮件地址	
mobile	移动电话号码	
o	组织名称	
ou	部门名称	通常为组织机构下的一个部门或者一个大

		型实体下的一个子实体
postalCode	邮政编码	
sn	姓，别名	
st	州或者省的名称	
street	街道地址	
userPassword	用户密码	
uid	用户 ID	
departmentNumber	部门编号	
displayName	显示名称	
description	描述	
employeeNumber	员工编号	
manager	经理	

下面是一些 LDAP 协议中定义的 object class（具体信息查看 RFC 2252 文档）

Object class	必须属性	可选属性
account	userid	description $ seeAlso $ localityName $ organizationName $ organizationalUnitName $ host
country	c	searchGuide $ description
dcObject	dc	
device	cn	serialNumber $ seeAlso $ owner $ ou $ o $ l $ description
inetOrgPerson->person 继承 person		audio $ businessCategory $ carLicense $ departmentNumber $ displayName $ employeeNumber $ employeeType $ givenName $ homePhone $ homePostalAddress $ initials $ jpegPhoto $ labeledURI $ mail $ manager $ mobile $ o $ pager $ photo $ roomNumber $ secretary $ uid $ userCertificate $ x500uniqueIdentifier $ preferredLanguage

		$ userSMIMECertificate $ userPKCS12
organizationalPerson 继承 Person		title $ x121Address $ registeredAddress $ destinationIndicator $ preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $ telephoneNumber $ internationaliSDNNumber $ facsimileTelephoneNumber $ street $ postOfficeBox $ postalCode $ postalAddress $ physicalDeliveryOfficeName $ ou $ st $ l
organization	o	userPassword $ searchGuide $ seeAlso $ businessCategory $ x121Address $ registeredAddress $ destinationIndicator $ preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $ telephoneNumber $ internationaliSDNNumber $ facsimileTelephoneNumber $ street $ postOfficeBox $ postalCode $ postalAddress $ physicalDeliveryOfficeName $ st $ l $ description
organizationalRole	cn	x121Address $ registeredAddress $ destinationIndicator $ preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $ telephoneNumber $ internationaliSDNNumber $ facsimileTelephoneNumber $ seeAlso $ roleOccupant $ preferredDeliveryMethod

		$ street $ postOfficeBox $ postalCode $ postalAddress $ physicalDeliveryOfficeName $ ou $ st $ l $ description
organizationalUnit	ou	userPassword $ searchGuide $ seeAlso $ businessCategory $ x121Address $ registeredAddress $ destinationIndicator $ preferredDeliveryMethod $ telexNumber $ teletexTerminalIdentifier $ telephoneNumber $ internationaliSDNNumber $ facsimileTelephoneNumber $ street $ postOfficeBox $ postalCode $ postalAddress $ physicalDeliveryOfficeName $ st $ l $ description
person	cn sn	userPassword $ telephoneNumber $ seeAlso $ description
Top(所有类的基类)		

七、 常用目录服务器

1:Apache directory server

2:Sun directory server

3:openDS 一个开源的，基于 LDAP 和 DSML 标准的 Directory service。Directory service 不仅包括 Directory server，还有其它与 directory 相关的基本 service：directory proxy 、virtual dirctory、namespace distribution 和数据同步 Directory server 是一个可以通过网络访问，信息分级存储的数据库。OpenDS 只能用在 linux 操作系统。该项目的地址为：http://www.opends.org/

4: Netscape Directory Server

5: Window AD

八、 Apache directory server 安装与使用

1:下载 ApacheDS 地址为: http://directory.apache.org/apacheds/1.5/

2:ApacheDS 的安装比较简单，没有什么特殊的设置。在 ApacheDS 安装完成后要启动 ApacheDS 服务。路径如下：控制面板---→ 管理工具---→服务--→Apache Directory server

ApacheDS 的监听端口默认为 10389

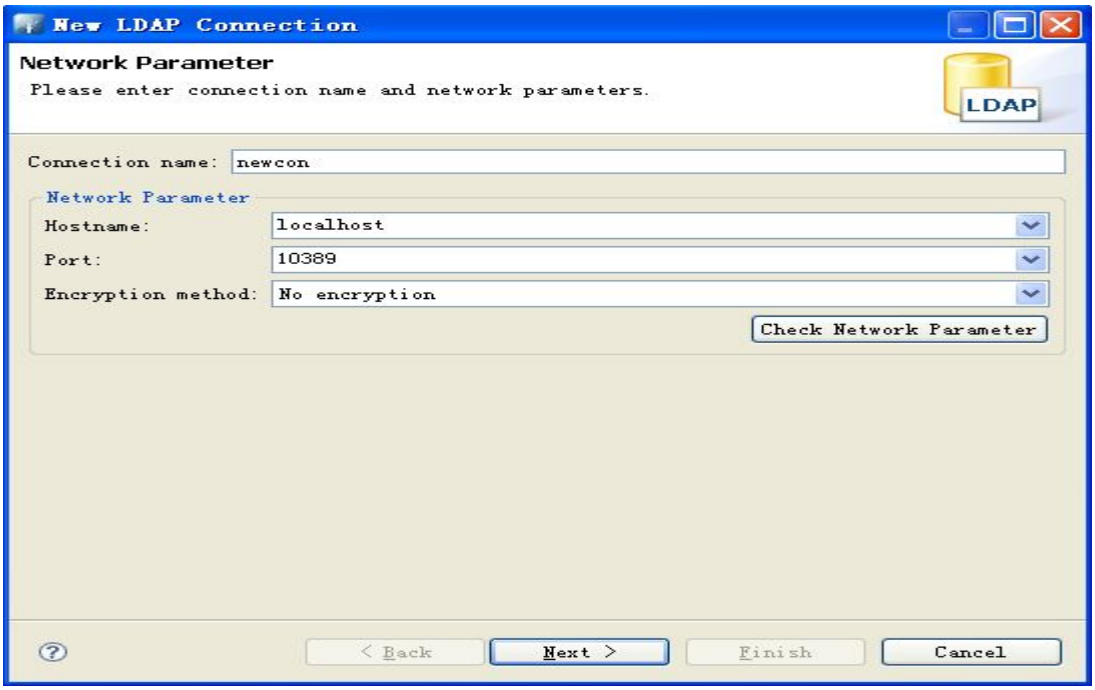
3:安装 Apache directory studio。下载地址同上。该软件是 ApacheDS 的一个客户端工具。用于连接

ApacheDS，搜索，更新，删除，添加数据。安装时也没有特别设置，请按默认设置安装该软件。

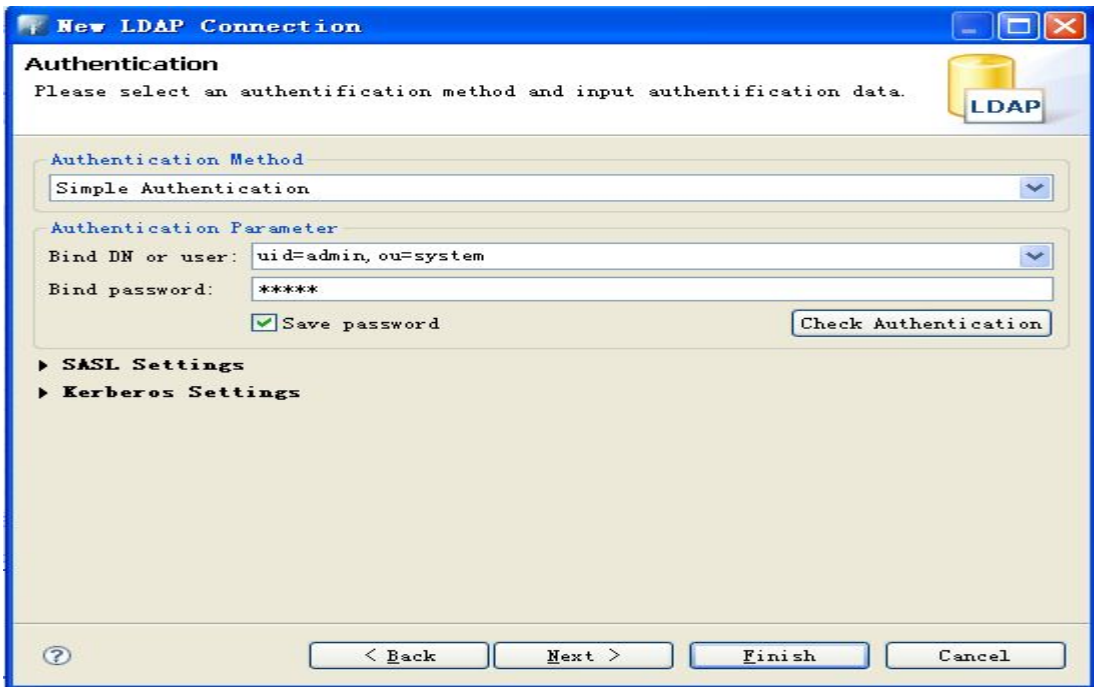
4:使用 Apache directory studio 连接 ApacheDS 步骤如下：

第一步：在菜单栏中选择 LDAP 菜单下的 New Connection 选项

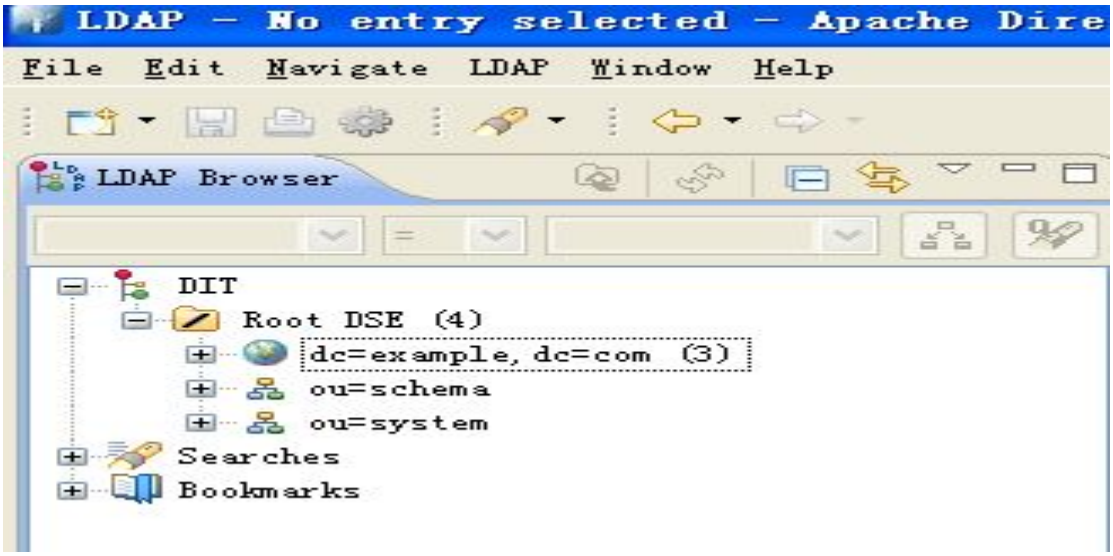
第二步：在下面的对话框中输入相应的值。



第三步：在一下对话框中输入要连接的根 DN 和密码。在 ApacheDS 安装完成后，已经创建了几个 DIT（目录信息树），第一次登录 ApacheDS 的话一般会连接如下输入的根节点的 DN 密码为：secret。当然可以在第一个选项框中选择 No Authentication。这是不需要输入下面的验证信息。这种登录术语匿名登录。权限受限制。



第四步。连接完成后，如下图：图中出现了 apahceDS 安装默认创建的三个目录信息树



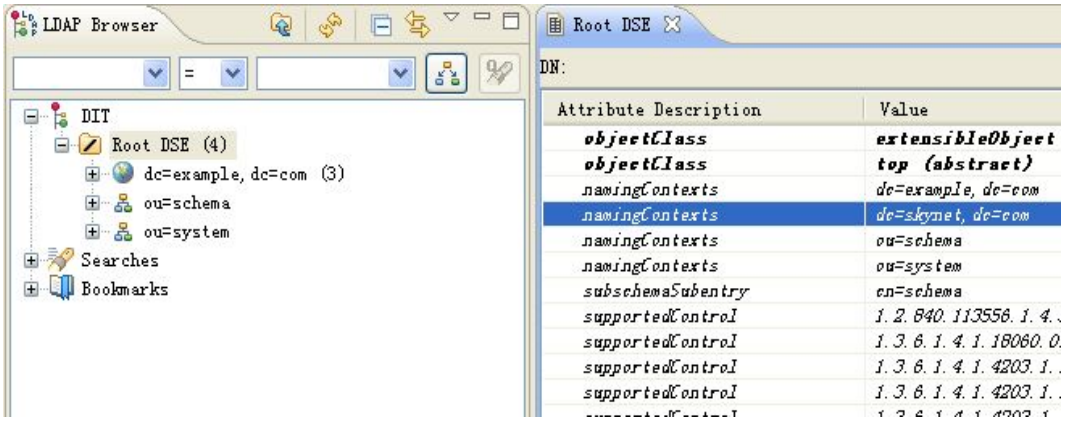
第五步：创建 Partition。ApacheDS 中的目录信息是保存在 Partition 中的。所以在创建新的目录树之前首先要创建 Partition。创建 Partition 的一个方法是修改 server.xml 配置文件。该文件路径如下：C:\Program Files\Apache Directory Server\instances\default\conf\

在该文件中找到 Partitions 元素。添加<jdbmPartition>元素。

```
<partitions>
  ......
  <jdbmPartition id="skynet" cacheSize="100" suffix="dc=skynet,dc=com" optimizerEnabled="true"
                 syncOnWrite="true"/>

</partitions>
```

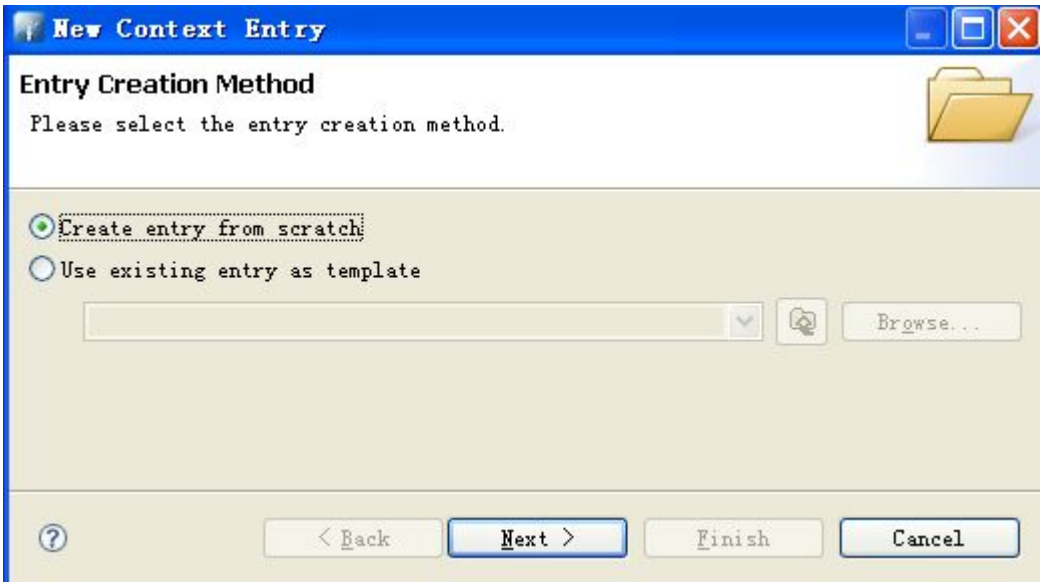
第六步：创建完成之后，重启 Apache direcory server 服务。在 Apache directory studio 视图中看到结果如下，在右侧栏中我们可以看到我们刚刚添加的 Partition。但是在左侧栏中，我们并没有看到该目录分支存在。因为，必须要创建一个 Context Entry。



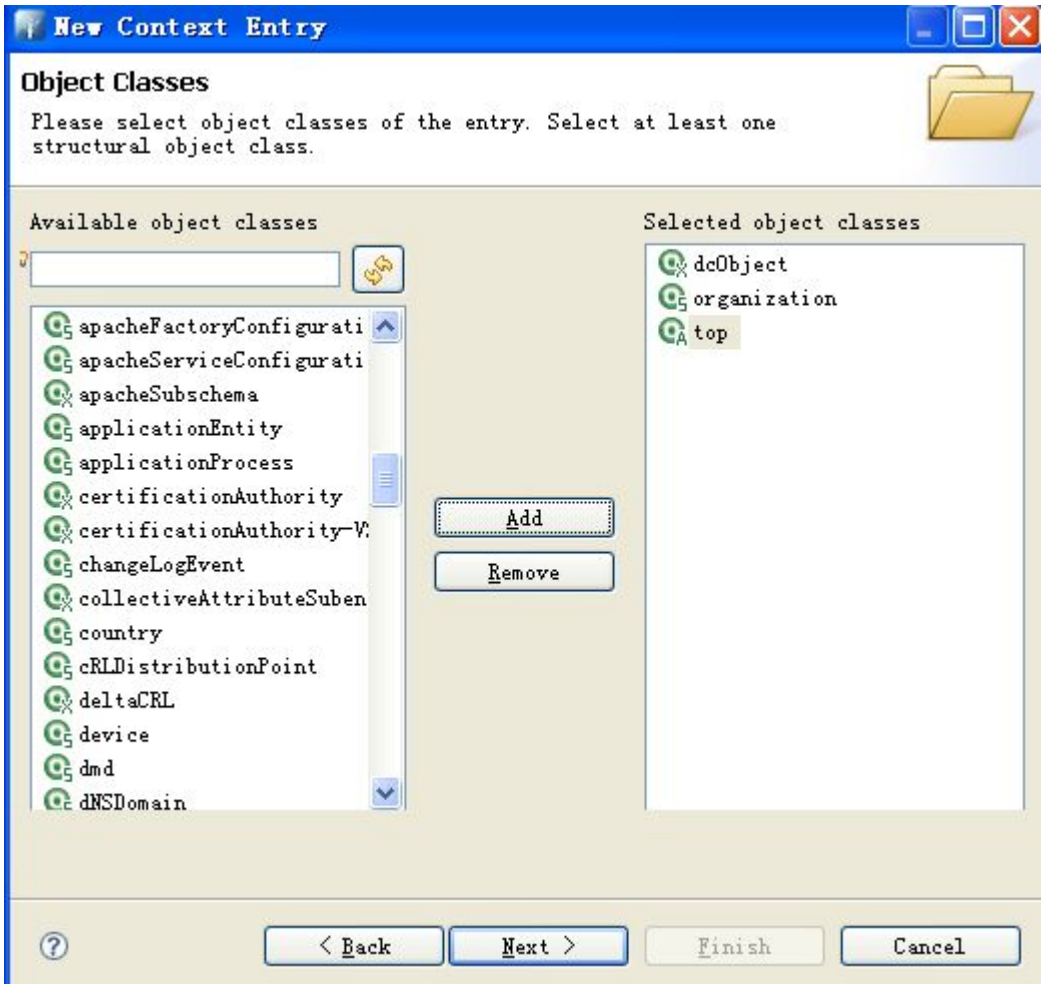
第七步：创建 Context Entry

1

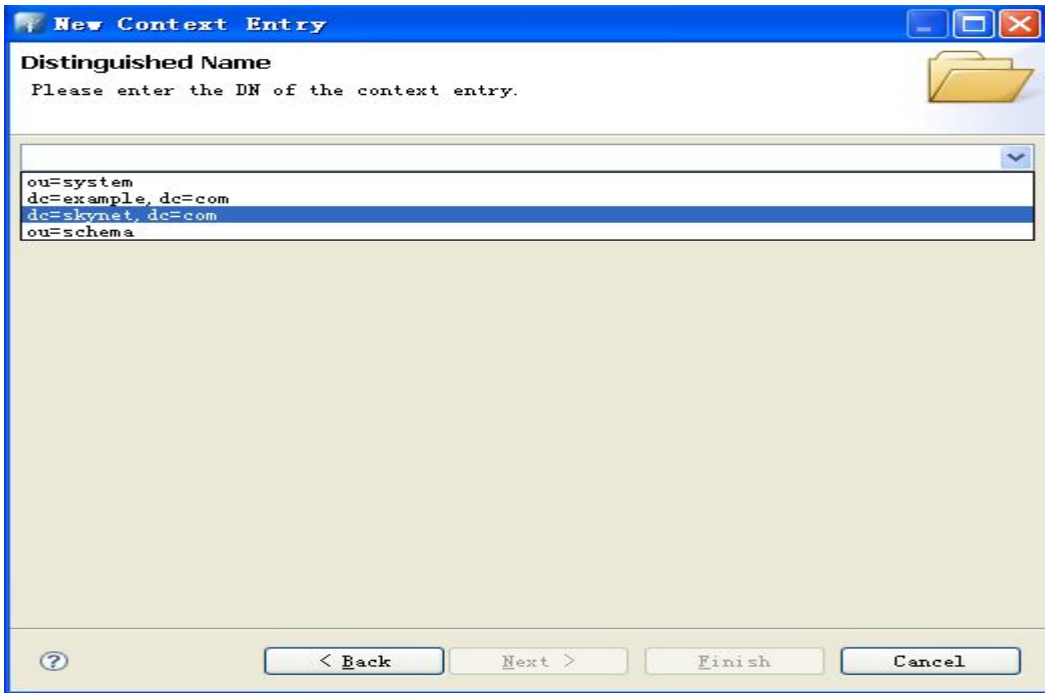
2



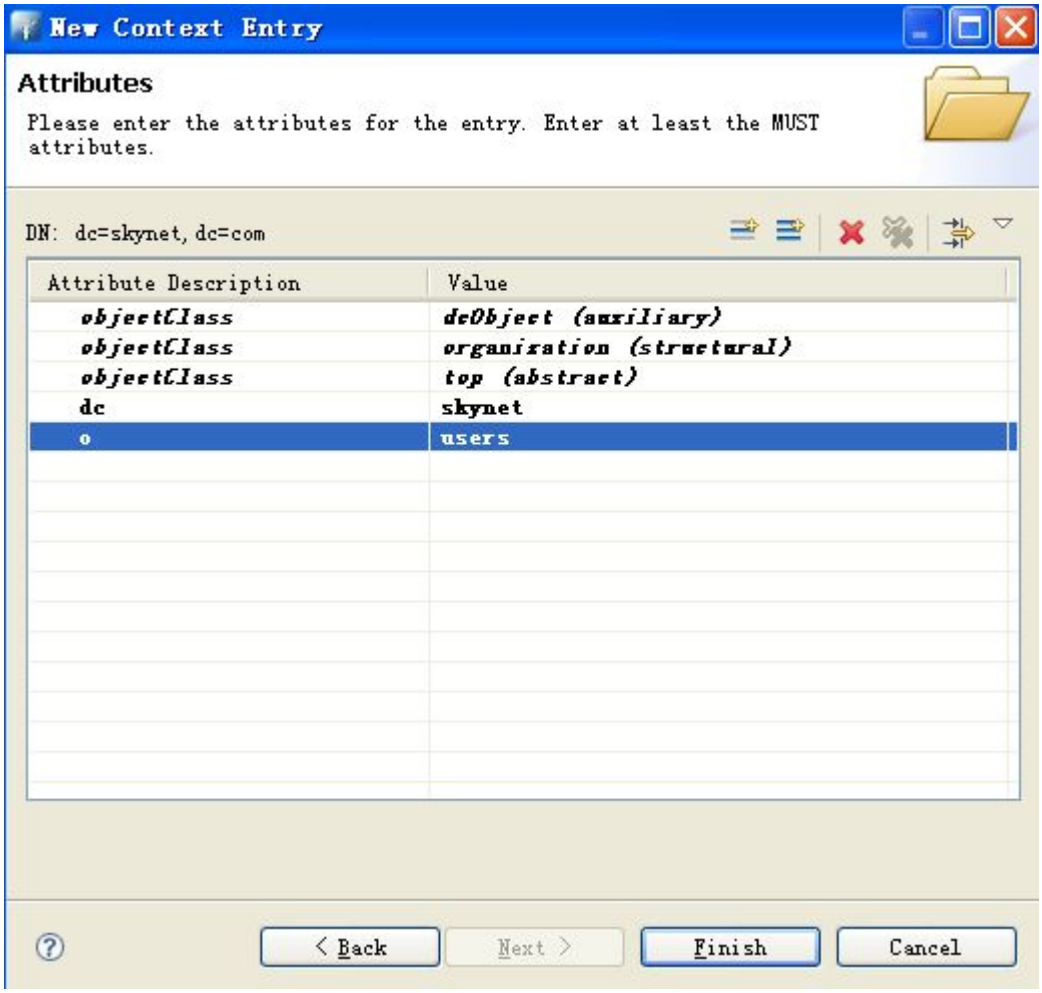
3


New Context Entry
Object Classes
Please select object classes of the entry. Select at least one structural object class.
Available object classes
Selected object classes
apacheFactoryConfigurati
apacheServiceConfigurati
apacheSubschema
applicationEntity
applicationProcess
certificationAuthority
certificationAuthority-V
changeLogEvent
collectiveAttributeSuben
country
cRLDistributionPoint
deltaCRL
device
dmd
dNSDomain
dcObject
organization
top
Add
Remove
< Back
Next >
Finish
Cancel

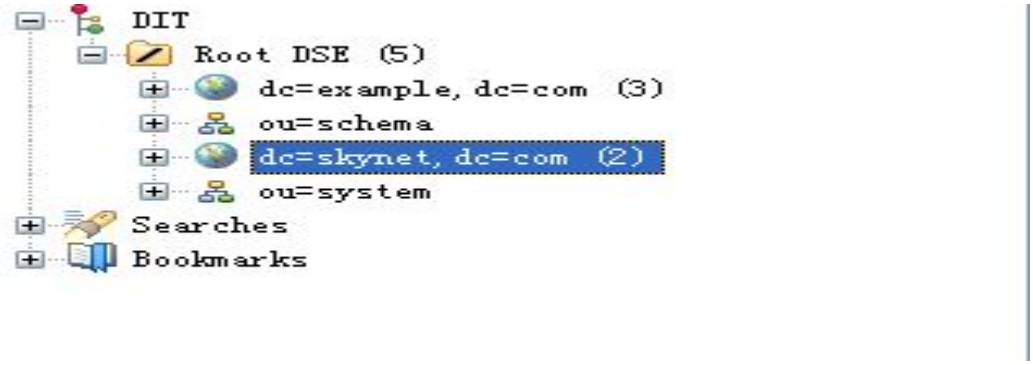
4


New Context Entry
Distinguished Name
Please enter the DN of the context entry.
ou=system
dc=example, dc=com
dc=skynet, dc=com
ou=schema
< Back
Next >
Finish
Cancel

5



6



九、 LDAP 应用场景

使用 JNDI 操作目录服务

1. **准备连接目录服务器的相应配置文件（以 ApacheDS 为例）**

   apacheds.properties

```
java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
java.naming.provider.url=ldap://localhost:10389/dc=example,dc=com
java.naming.security.authentication=simple
java.naming.security.principal=uid=liujz,dc=example,dc=com
java.naming.security.credentials=123456
```

2. **创建测试类**

```
public class LDAPTest {
   private static DirContext ctx =null;
   private Properties    ldapProps ;
   public LDAPTest(Properties    ldapProp){
     This. ldapProp = ldapProp;
   }
   public DirContext getDirContext(){
     If(ctx=null){
       ctx = new InitialDirContext(ldapProps);
     }
     return ctx ;
   }
   //从 LDAP 服务器中查询符合条件的 Entry
   public void queryEntry(){
     SearchControls sc = new SearchControls();
     //用于设置查询范围
     //有三种查询范围
      SUBTREE_SCOPE：表示在以指定对象为根的子树中查找，可以返回多个元素
      ONLEVEL_SCOPE:表示指定对象极其直接子实体
      OBJECT_SCOPE:表示返回指定对象。
     sc.setSearchScope(SearchControls.SUBTREE_SCOPE);
     //该方法有单个参数，第一个参数是 DN 是一种相对的，因为根据配置文件，我们已经连接
      到根 DN 为 dc=example，dc=com 的目录树上，所以一下方法查询的 Entry 的 DN 为
      ou=Account，dc=example，dc=com。第二个参数是过滤器，相当于 SQL 中的 where 子句。
      第三个参数为查询控制
     String dn = "ou=Account";
     String filter = "ou=Account";
     NamingEnumeration result = getDirContext().search(dn,filter,sc);
     While(result.hasMore()){
       SearchResult entry = (SearchResult)result.next();
       Attributes attrs = entry.getAttributes();
       Attribute attr = attrs.get("ou");
       System.out.println("ou="+attr.get());
     }
   }
   //向 LDAP 服务器中添加 Entry
   Public void addEntry(){
     // 在 ou=account， dc=example， dc=com 节点下添加一个子节点。其 RDN 为
      cn=liujianzhong
     Attributes attrs = new BasicAttributes();
     attrs.put("cn","liujianzhong");
     attrs.put("sn","liu");
     attrs.put("userpassword","123456");
     BasicAttribute objectClassSet = new BasicAtrribute("objectclass");
```

```
  objectClassSet.add("top"):
  objectClassSet.add("person");
  objectClassSet.add("organizationalPerson");
  objectClassSet.add("inetOrgPerson");
  attrs.put(objectClassSet);
  ctx.createSubContext("cn=liujianzhong,ou=Account",attrs);
}
//删除 LDAP 目录服务器中指定的 Entry，如果该节点为叶子节点则直接删除，否则要先得到
 该节点下的所有叶子节点，然后从最底层的叶子节点删除，直到删除所有叶子节点为止
Public void delEntry(){
  //删除叶子节点
  String DN = "cn=liujianzhong,ou=Account" ;
  getDirContext().destroySubcontext(DN);
  //删除非叶子节点
  String DN = "ou=Account,dc=example,dc=com" ;
}
//删除非叶子节点
Public void delDN(String dn,DirContext ctx){
  String root ="dc=example,dc=com";
  SearchControls sc = new SearchControls();
  String filter = "(objectclass=*)";
  sc.setSearchScope(SearchControls.ONELEVEL_SCOPE);
  NamingEnumeration results = ctx.search(dn, filter, sc);
  while(results.hasMore()){
    SearchResult entry = (SearchResult)results.nextElement();
    String name = entry.getNameInNamespace();
    int rin = name.length()-root.length()-1;
    String rdn = name.substring(0, rin);
    delDn(rdn,ctx);
  }
  ctx.destroySubcontext(dn);
}
//修改 LDAP 服务器中 Entry 的属性
public void modifyEntry(String dn,DirContext ctx) throws NamingException {
  BasicAttribute attr = new BasicAttribute("sn");
  attr.add("jianzhong");
  //修改属性
  ModificationItem[] mods = new ModificationItem[1];
  mods[0] = new ModificationItem(LdapContext.REPLACE_ATTRIBUTE,attr);
  ctx.modifyAttributes(dn, mods);
  //添加属性
  BasicAttribute psw = new BasicAttribute("userPassword");
  Psw.add("123456");
  ModificationItem[] add=new ModificationItem[1];
```

```
            Add[0] = new ModificationItem(LdapContext.ADD_ATTRIBUTE,psw);
            ctx.modifyAttributes(dn, add);
            //删除属性
            BasicAttribute psw1= new BasicAttribute("userPassword");
            ModificationItem[] rem=new ModificationItem[1];
            Rem[0] = new ModificationItem(LdapContext.ADD_ATTRIBUTE,psw1);
            ctx.modifyAttributes(dn, rem);
          }
        }
```

3.

十、