第 11 章　PHP 访问 LDAP

LDAP 的全称是“轻量级目录访问协议（Lightweight Directory Access Protocol）”，是一种简单的目录协议。所谓目录，是一种专门的数据库，可以用来服务于任何应用程序。在企业应用中使用 LDAP 可以让企业范围内的所有应用程序从 LDAP 目录中获取信息，应用程序可以从网络上直接从 LDAP 目录获取信息，而不局限于操作系统与服务器的类型。本章将主要介绍如何使用 PHP 来访问 LDAP。

11.1　LDAP 简介

LDAP 协议是一种跨平台的标准协议，当应用程序通过 LDAP 协议访问 LDAP 目录时，不需要考虑 LDAP 所在的服务器类型和操作系统类型，均可以直接对 LDAP 目录中的内容进行读写。

LDAP 由于得到了业界的广泛认证，越来越多的 LDAP 产品被开发出来。很多企业用户也将 LDAP 目录用于企业内部的信息管理。如图 11-1 所示的结构是一个企业的信息系统构架图，所有的子系统均使用放置在 LDAP 服务器上的 LDAP 目录信息进行用户的身份验证，并向系统返回用户的信息。

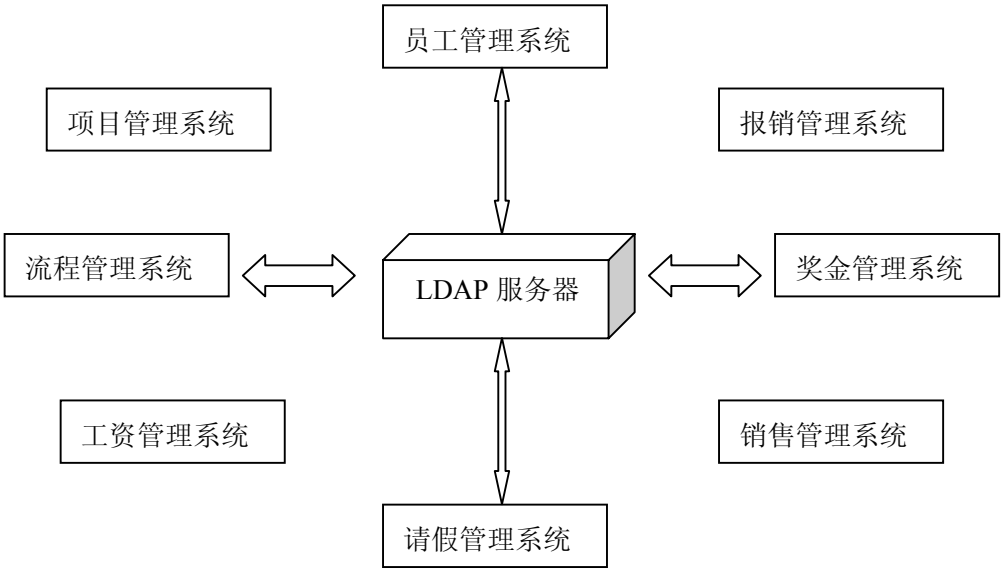

图 11-1　企业信息系统构架

在 Windows 下可以直接通过在浏览器上输入 LDAP 的域名来访问 LDAP 服务器，例如在浏览器上输入 ldap://192.168.3.1/可以看到如图 11-2 所示的窗口。

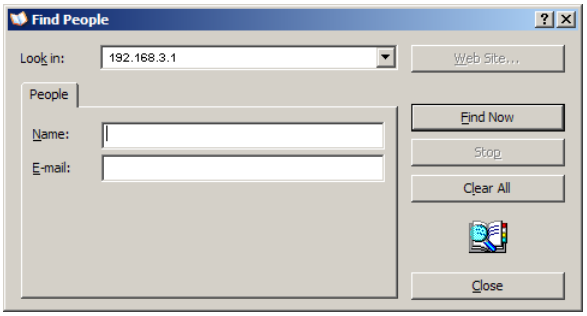

图 11-2　直接访问 LDAP 服务器

11.2　LDAP 服务器的安装与配置

OpenLDAP 是一款免费的 LDAP 服务器软件，在服务器上安装 OpenLDAP 可以方便、简单的实现 LDAP 服务的配置。本节将以 OpenLDAP 为例介绍如何在 Windows 平台上安装和配置 LDAP 服务器。

11.2.1　OpenLDAP 的安装

OpenLDAP 可以从其官方网站 http://www.openldap.org/免费下载到。与其他 Windows 安装程序类似，OpenLDAP 的安装步骤如下所示。

（1）双击安装文件图标启动安装程序，弹出【Setup – OpenLDAP】对话框。

（2）单击【Next】按钮，弹出【License Agreement】对话框。

（3）阅读协议后，选中【I accept the agreement】接受协议，单击【Next】按钮，弹出【Select Destination Location】对话框。

（4）选择好安装路径后，单击【Next】按钮，弹出【Select Components】对话框，如图 11-3 所示。

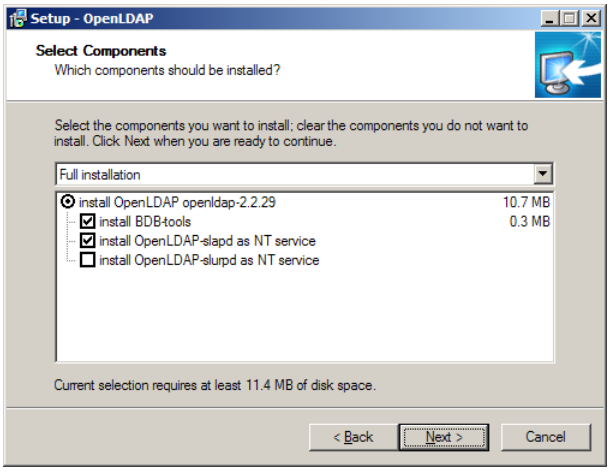

图 11-3　选择安装组件

（5）选择了是否安装客户端工具以及是否将 OpenLDAP 安装成 Windows 服务后，单击【Next】按钮，弹出【Select Start Menu Folder】对话框。

（6）单击【Next】按钮，弹出【Select Addition Tasks】对话框，如图 11-4 所示。

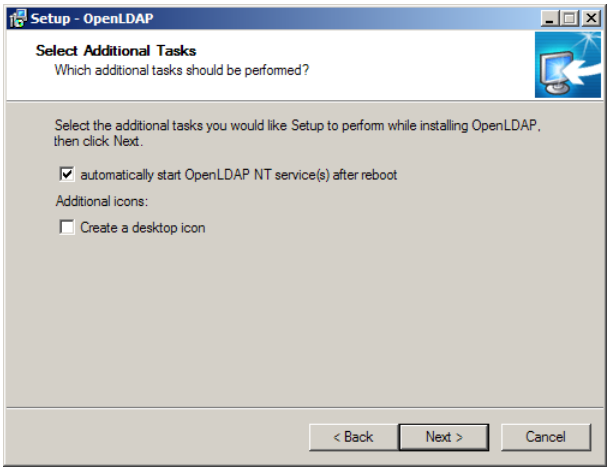

图 11-4　选择附加任务

（7）选择了是否在 Windows 启动时启动 LDAP 服务以及是否创建桌面图标后，单击【Next】按钮，弹出【Ready to Install】对话框。

（8）单击【Install】按钮，开始安装。

（9）安装完成后，单击【Finish】按钮，结束安装程序。

11.2.2　OpenLDAP 的配置

OpenLDAP 的安装目录下的 slapd.conf 文件是用于 OpenLDAP 服务器的配置文件。在启动 OpenLDAP 服务之前，需要首先修改该文件对 OpenLDAP 进行配置。主要的改动包括以下几点。

- ❑ 包含更多的 schema 文件。schema 文件是 OpenLDAP 提供的存储模式，在这里包含相应的模式文件可以使 OpenLDAP 使用相应的模式进行数据存储和管理，在 slapd.conf 文件的相应位置填写如下代码可以使 OpenLDAP 具有可以使用自带的各种模式进行数据存储。

```
include         ./schema/core.schema
include         ./schema/cosine.schema
include         ./schema/inetorgperson.schema
include         ./schema/nis.schema
include         ./schema/misc.schema
```

- ❑ 配置服务器根目录信息。修改 slapd.conf 文件中的 suffix 可以修改服务器的根目录信息，所有的其他数据都将建立在这个根目录之上，如下所示。

```
suffix          "dc=simon,dc=com"
```

- ❑ 设置管理员用户名和密码。修改 rootdn 中的 cn 值可以修改管理员账户，修改 rootpw 的值可以修改管理员账户的密码，如下所示。

```
rootdn          "cn=admin,dc=simon,dc=com"
rootpw          pass
```

配置后的文件如下所示。

```
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
ucdata-path     ./ucdata
include         ./schema/core.schema
```

```
include         ./schema/cosine.schema
include         ./schema/inetorgperson.schema
include         ./schema/nis.schema
include         ./schema/misc.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
#referral   ldap:/root.openldap.org

pidfile         ./run/slapd.pid
argsfile        ./run/slapd.args

# Load dynamic backend modules:
# modulepath   ./libexec/openldap
# moduleload   back_bdb.la
# moduleload   back_ldap.la
# moduleload   back_ldbm.la
# moduleload   back_passwd.la
# moduleload   back_shell.la

# Sample security restrictions
#      Require integrity protection (prevent hijacking)
#      Require 112-bit (3DES or better) encryption for updates
#      Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
#      Root DSE: allow anyone to read it
#      Subschema (sub)entry DSE: allow anyone to read it
#      Other DSEs:
#            Allow self write access
#            Allow authenticated users read access
#            Allow anonymous users to authenticate
#      Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
# access to *
#      by self write
#      by users read
#      by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn.    (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

#######################################################################
```

```
# BDB database definitions
#######################################################################

database  bdb
suffix          "dc=simon,dc=com"
rootdn          "cn=admin,dc=simon,dc=com"
# Cleartext passwords, especially for the rootdn, should
# be avoid.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw          pass
# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory   ./data
# Indices to maintain
index       objectClass      eq
```

11.2.3　OpenLDAP 的启动与关闭

OpenLDAP 服务器可以通过 Windows 控制面板中的服务进行启动和关闭，如图 11-5 所示。

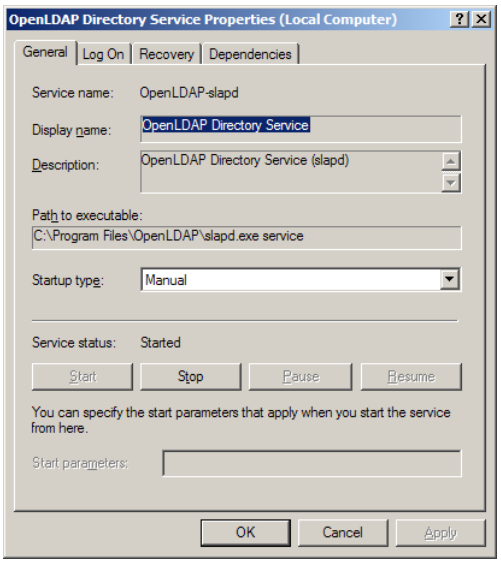

图 11-5　OpenLDAP 服务的启动和关闭

单击【OpenLDAP Directory Service Properties】对话框中的【Start】和【Stop】按钮可以实现 LDAP 服务器的启动与关闭。除此之外，也可以使用 OpenLDAP 安装目录下的 slapd 命令来完成。

启动 OpenLDAP 的命令如下所示。

```
slapd start
```

关闭 OpenLDAP 的命令如下所示。

```
slapd stop
```

11.2.4　OpenLDAP 的数据操作

使用 OpenLDAP 安装目录下的 ldapadd 命令可以实现向 LDAP 目录中增加数据，具体命令如下所示。

```
ldapadd -x -D "cn=admin,dc=simon,dc=com" –W
```

其中，-x 表示使用简单验证方式，-D 及其后面的参数表示当前的根目录和账户信息，-W 表示当前管理员登录需要使用密码进行身份验证。命令运行后，将提示输入密码。输入密码后，就可以向 LDAP 目录中输入数据了。输入的数据如下所示。

```
dn:dc=simon,dc=com
objectClass:dcObject
objectClass:organization
dc:simon
o:company
description:this is a test
```

其中，dn 表示根目录信息。objectClass 表示对象的类别，即在 OpenLDAP 模式中指定的属性。后面的 dc、o 和 description 分别表示数据的属性。数据输入后，按 Ctrl+Z 键完成输入。

在添加了数据后，可以使用 ldapsearch 命令查看数据，具体命令如下所示。

```
ldapsearch -x -b "dc=simon,dc=com"
```

其中，-x 表示使用简单验证方式，-b 及其后面的参数表示要进行查询的根目录信息。对于上面输入的数据，该命令返回以下结果。

```
# extended LDIF
#
# LDAPv3
# base <dc=simon,dc=com> with scope sub
# filter: (objectclass=*)
# requesting: ALL
#

# simon.com
dn: dc=simon,dc=com
objectClass: dcObject
objectClass: organization
dc: simon
o: company
description: this is a test

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

可以看到，前面输入的数据被成功输出了。

11.2.5　phpLDAPadmin 简介

前面介绍了使用命令行方式对 LDAP 服务器进行数据的添加和查找。除此之外，使用图形工具

phpLDAPadmin 可以更方便的进行 LDAP 服务器的管理。phpLDAPadmin 是一款使用 PHP 编写的 LDAP 服务器管理系统，该系统可以从其官方网站 http://sourceforge.net/projects/phpldapadmin/免费下载到。下载后解压缩到 Web 服务器的文档文件夹即可运行。登录后，可以从 phpLDAPadmin 中查看到 LDAP 服务器中的数据，如图 11-6 所示。

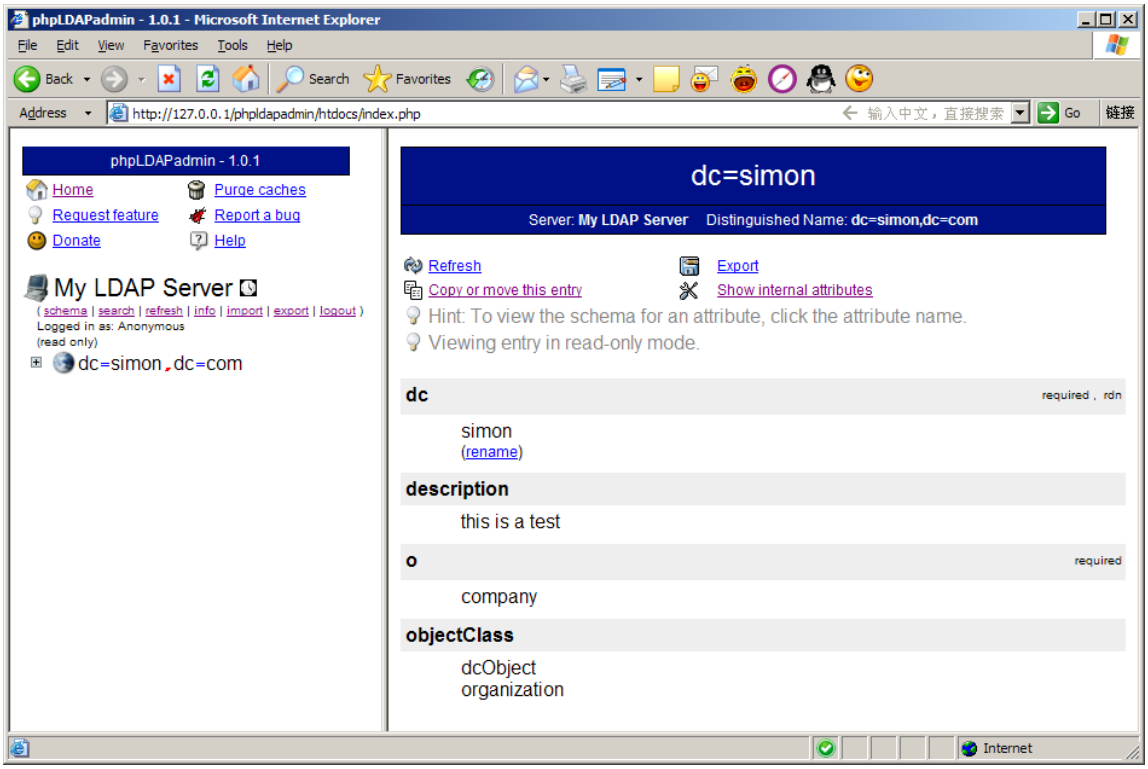

图 11-6　phpLDAPadmin

11.3　PHP 中 LDAP 扩展的配置

前面简要介绍了如何进行 LDAP 服务器的安装与配置，本节将介绍如何使用 PHP 实现对 LDAP 服务器的连接。PHP 提供了专门用于访问 LDAP 的扩展，但是，在默认情况下，这个扩展并没有被激活。激活 LDAP 的扩展的方法是找到 PHP 安装目录下的 php.ini 文件，找到如下所示的代码。

```
;extension=php_ldap.dll
```

去掉前面的分号，然后重新启动 Apache 就可以完成配置了。验证 LDAP 是否已经可用可以通过在浏览器上运行以下代码来实现。

```
<?php
phpinfo();
?>
```

如果在浏览器中可以看到如图 11-7 所示的配置信息，就说明 LDAP 的扩展已经成功完成了。

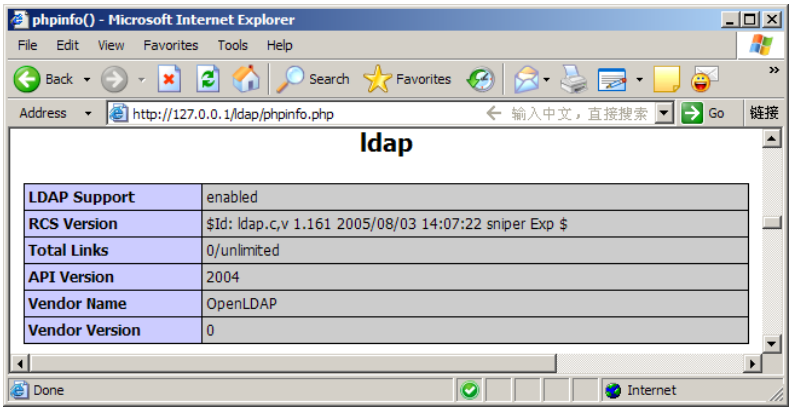

图 11-7　LDAP 配置信息

11.4　PHP 与 LDAP 的相关操作

本节将介绍 PHP 中专门用于操作 LDAP 的函数。正如 11.1 节中介绍的那样，LDAP 目录与关系数据库很相像，因此，用于访问 LDAP 的操作也与数据库操作很相像。

11.4.1　连接 LDAP 服务器

PHP 中用于连接 LDAP 服务器的函数是 ldap_connect，其语法格式如下所示。

```
ldap_connect([string hostname [, int port]])
```

其中，hostname 是 LDAP 服务器所在的主机地址，port 是 LDAP 服务器的端口号。

以下代码实现了对位于 192.168.3.1 地址的 LDAP 服务器的连接。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                          //LDAP 服务器地址
    $ldap_port = "389";                                         //LDAP 服务器端口号
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                or die("Can't connect to LDAP server");         //建立与 LDAP 服务器的连接
?>
```

与前面介绍过的方法类似，上面的代码使用了“or die”来美化错误信息。

11.4.2　绑定 LDAP 服务器

绑定 LDAP 服务器的含义是使用特定的用户名或密码来登陆 LDAP 服务器。PHP 中用于绑定 LDAP 服务器的函数是 ldap_bind，其语法格式如下所示。

```
ldap_bind(ldap_conn [, string username [, string password]])
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，username 是登陆 LDAP 服务器时使用的用户名，password 是登陆时所用的密码。以下代码实现了对位于 192.168.3.1 地址的 LDAP 服务器的绑定。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                          //LDAP 服务器地址
    $ldap_port = "389";                                         //LDAP 服务器端口号
```

```
    $ldap_user = "";                                                    //设定服务器用户名
    $ldap_pwd   = "";                                                   //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                or die("Can't connect to LDAP server");                 //建立与 LDAP 服务器的连接
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                or die("Can't bind to LDAP server.");                   //与服务器绑定
?>
```

11.4.3　断开 LDAP 服务器

与 LDAP 服务器断开的过程与绑定 LDAP 服务器相反，PHP 中用于绑定 LDAP 服务器的函数是 ldap_unbind，其语法格式如下所示。

```
ldap_unbind(ldap_conn)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象。以下代码在绑定了对位于 192.168.3.1 地址的 LDAP 服务器后与其断开连接。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                                  //LDAP 服务器地址
    $ldap_port = "389";                                                 //LDAP 服务器端口号
    $ldap_user = "";                                                    //设定服务器用户名
    $ldap_pwd   = "";                                                   //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                or die("Can't connect to LDAP server");                 //建立与 LDAP 服务器的连接
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                or die("Can't bind to LDAP server.");                   //与服务器绑定
    ldap_unbind($ldap_conn)
                or die("Can't unbind from LDAP server.");               //与服务器断开连接
?>
```

11.4.4　查询 LDAP 目录内容

查询 LDAP 目录使用 ldap_search 函数来实现，其语法格式如下所示。

```
ldap_search(ldap_conn, base_dn, conditions)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象。base_dn 是 LDAP 服务器的查询主键。conditions 是用于 LDAP 目录查询所用的条件。该函数返回一个结果对象，该结果对象保存查询到的所有记录。

对于这个结果对象，可以使用 ldap_get_entries 函数进行简单的读取，其语法格式如下所示。

```
ldap_get_entries(ldap_conn, result)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，result 是前面查询 LDAP 目录时返回的对象。该函数返回一个数组，包含所有的结果记录。以下代码实现了对服务器上的内容进行查询。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                                  //LDAP 服务器地址
    $ldap_port = "389";                                                 //LDAP 服务器端口号
    $ldap_user = "";                                                    //设定服务器用户名
    $ldap_pwd   = "";                                                   //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                or die("Can't connect to LDAP server");                 //建立与 LDAP 服务器的连接
```

```
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                    or die("Can't bind to LDAP server.");                    //与服务器绑定
    $base_dn     = "ou=company,o=depart";                                    //定义要进行查询的目录主键
    $filter_col = "mail";                                                    //定义用于查询的列
    $filter_val = "phptester@163.com";                                       //定义用于匹配的值
    $result       = ldap_search($ldap_conn, $base_dn, "($filter_col=$filter_val)");       //执行查询
    $entry        = ldap_get_entries($ldap_conn, $result);                   //获得查询结果
    print_r($entry);
      //输出查询结果
    ldap_unbind($ldap_conn)
                    or die("Can't unbind from LDAP server.");                //与服务器断开连接
?>
```

运行结果如下所示。

```
Array
(
    [count] => 1
    [0] => Array
        (
            [objectclass] => Array
                (
                    [count] => 5
                    [0] => person
                    [1] => organizationalPerson
                    [2] => companyPerson
                    [3] => departPerson
                    [4] => top
                )

            [0] => objectclass
            [ou] => Array
                (
                    [count] => 1
                    [0] => company
                )

            [1] => ou
            [o] => Array
                (
                    [count] => 1
                    [0] => depart
                )

            [2] => o
            [employeeserialnumber] => Array
                (
                    [count] => 1
                    [0] => 100001
                )

            [3] => employeeserialnumber
```

```
            [givenname] => Array
                (
                    [count] => 2
                    [0] => Peng Cheng
                    [1] => Peng
                )

            [4] => givenname

            [mail] => Array
                (
                    [count] => 1
                    [0] => phptester@163.com
                )

            [5] => mail
            [count] => 6
            [dn] => uid=672100001,c=cn,ou=company,o=depart
        )

)
```

可以看出，访问 LDAP 服务器与查询数据库中的记录很相似。事实上，在实际应用中，LDAP 服务器也与数据库服务器有着相似的作用。

除了上面的用法之外，ldap_search 函数还支持通配符的使用。例如，将前面的用于 LDAP 服务器查询的代码修改如下。

```
<?php
    $filter_val = "*@163.com";                                                  //定义用于匹配的值
    $result      = ldap_search($ldap_conn, $base_dn, "($filter_col=$filter_val)");      //执行查询
?>
```

这里，就会将所有包含以“@163.com”结尾的邮件地址的记录返回。

11.4.5　获得查询结果中的值

上面的例子完整的获得了 LDAP 服务器查询结果的信息，这样，根据数组中的值就可以进行其他操作了。除此之外，PHP 还提供了专门的用于获得查询结果值的方法。

首先介绍一种与 ldap_get_entries 相似的函数——ldap_first_entry，该函数仅获得结果对象中的第一条记录，其语法格式如下所示。

```
ldap_first_entry(ldap_conn, result)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，result 是前面查询 LDAP 目录时返回的对象。

获取结果中的值的函数为 ldap_get_values，该函数的语法格式如下所示。

```
ldap_get_values(ldap_conn, entry, column)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，entry 是前面查询查询结果时返回的对象，column 是要返回的值所在的列的名称。该函数返回一个仅包含该列信息的数组。以下代码返回了前面数组的 givenname 列。

```
<?php
```

```
    $ldap_host = "ldap://192.168.3.1";                                    //LDAP 服务器地址
    $ldap_port = "389";                                                   //LDAP 服务器端口号
    $ldap_user = "";                                                      //设定服务器用户名
    $ldap_pwd   = "";                                                     //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                    or die("Can't connect to LDAP server");               //建立与 LDAP 服务器的连接
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                    or die("Can't bind to LDAP server.");                 //与服务器绑定
    $base_dn     = "ou=company,o=depart";                                 //定义要进行查询的目录
    $filter_col = "mail";                                                 //定义用于查询的列
    $filter_val = "phptester@163.com";                                    //定义用于匹配的值
    $result      = ldap_search($ldap_conn, $base_dn, "($filter_col=$filter_val)");       //执行查询
    $entry = ldap_first_entry($ldap_conn, $result);                       //获得第一个查询结果
    $firstname = ldap_get_values($ldap_conn, $entry, "givenname");        //获得查询结果中的值
    print_r($firstname);                                                  //输出
    ldap_unbind($ldap_conn)
                    or die("Can't unbind from LDAP server.");             //与服务器断开连接
?>
```

运行结果如下所示。

```
Array
(
    [0] => Peng Cheng
    [1] => Peng
    [count] => 2
)
```

11.4.6　计算查询结果中的记录数

计算查询结果中的记录数 ldap_count_entries 函数来实现，该函数的语法格式如下所示。

```
ldap_count_entries(ldap_conn, result)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，result 是前面查询 LDAP 目录时返回的对象。以下代码计算出了查询结果中的记录数。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                                    //LDAP 服务器地址
    $ldap_port = "389";                                                   //LDAP 服务器端口号
    $ldap_user = "";                                                      //设定服务器用户名
    $ldap_pwd   = "";                                                     //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                    or die("Can't connect to LDAP server");               //建立与 LDAP 服务器的连接
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                    or die("Can't bind to LDAP server.");                 //与服务器绑定
    $base_dn     = "ou=company,o=depart";                                 //定义要进行查询的目录
    $filter_col = "mail";                                                 //定义用于查询的列
    $filter_val = "*@163.com";                                            //定义用于匹配的值
    $result      = ldap_search($ldap_conn, $base_dn, "($filter_col=$filter_val)");       //执行查询
    $count       = ldap_count_entries($ldap_conn, $result);               //计算查询结果中的记录数
    echo "Total records count: ".$count;                                  //输出查询结果中的记录数
    ldap_unbind($ldap_conn)
                    or die("Can't unbind from LDAP server.");             //与服务器断开连接
```

```
?>
```

需要注意的是这里使用了通配符进行 LDAP 数据库的查询，因此，可能会有多条记录被返回。

11.4.7　向 LDAP 添加一条新记录

向 LDAP 添加一条新记录使用 ldap_add 函数来完成。

```
ldap_add(ldap_conn, base_dn, entry)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，base_dn 是 LDAP 服务器的查询主键，entry 是储存新记录的数组。以下代码实现了向 LDAP 服务器添加一条新记录的功能。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                          //LDAP 服务器地址
    $ldap_port = "389";                                         //LDAP 服务器端口号
    $ldap_user = "";                                            //设定服务器用户名
    $ldap_pwd    = "";                                          //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                or die("Can't connect to LDAP server");         //建立与 LDAP 服务器的连接
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                or die("Can't bind to LDAP server.");           //与服务器绑定
    $base_dn     = "ou=company,o=depart";                       //定义要进行查询的目录
    $entry["givenname"] = "Simon";                              //定义新记录数组
    $entry["company"] = "PHP workshop";
    $entry["mail"] = "pch1982cn@cn.yahoo.com";
    $entry["serial_no"] = "100001";
    ldap_add($ldap_conn, $base_dn, $entry)
                or die("Can't add new entry!");
    ldap_unbind($ldap_conn)
                or die("Can't unbind from LDAP server.");       //与服务器断开连接
?>
```

11.4.8　更新 LDAP 中的一条记录

更新 LDAP 中的一条记录使用 ldap_modify 函数来完成。

```
ldap_modify(ldap_conn, base_dn, entry)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，base_dn 是 LDAP 服务器的查询主键，entry 是储存更新后的记录的数组。以下代码实现了更新 LDAP 记录的功能。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                          //LDAP 服务器地址
    $ldap_port = "389";                                         //LDAP 服务器端口号
    $ldap_user = "";                                            //设定服务器用户名
    $ldap_pwd    = "";                                          //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                or die("Can't connect to LDAP server");         //建立与 LDAP 服务器的连接
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                or die("Can't bind to LDAP server.");           //与服务器绑定
    $base_dn     = "ou=company,o=depart";                       //定义要进行查询的目录
    $entry = array("company" => "PHP Workshop", "mail" => "pengcheng.sun@yahoo.com");
    //设定要修改的记录属性
```

```
    ldap_modify($ldap_conn, $base_dn, $entry)
                or die("Can't modify entry.");                    //修改记录
    ldap_unbind($ldap_conn)
                or die("Can't unbind from LDAP server.");         //与服务器断开连接
?>
```

11.4.9　从 LDAP 中删除一条新记录

删除 LDAP 中的一条记录使用 ldap_delete 函数来完成。

```
ldap_modify(ldap_conn, base_dn)
```

其中，ldap_conn 是前面连接 LDAP 服务器时创建的连接对象，base_dn 是 LDAP 服务器的查询主键。以下代码实现了删除 LDAP 记录的功能。

```
<?php
    $ldap_host = "ldap://192.168.3.1";                            //LDAP 服务器地址
    $ldap_port = "389";                                           //LDAP 服务器端口号
    $ldap_user = "";                                              //设定服务器用户名
    $ldap_pwd  = "";                                              //设定服务器密码
    $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                or die("Can't connect to LDAP server");           //建立与 LDAP 服务器的连接
    ldap_bind($ldap_conn, $ldap_user, $ldap_pwd)
                or die("Can't bind to LDAP server.");             //与服务器绑定
    $base_dn   = "ou=company,o=depart";                           //定义要进行查询的目录
    ldap_delete($ldap_conn, $base_dn) or die("Can't delete entry."); //修改记录
    ldap_unbind($ldap_conn)
                or die("Can't unbind from LDAP server.");         //与服务器断开连接
?>
```

11.4.10　错误处理

PHP 还提供了对 LDAP 操作的错误处理的方法。主要包括 3 个函数——ldap_errno、ldap_error 和 ldap_err2str。

ldap_errno 的语法格式如下所示。

```
int ldap_errno(resource)
```

其中 resource 是在 LDAP 操作中产生的对象，该函数将返回一个错误代码。

ldap_error 的语法格式如下所示。

```
string ldap_error(resource)
```

其中 resource 是在 LDAP 操作中产生的对象，该函数将返回一个错误信息。

ldap_err2str 的语法格式如下所示。

```
string ldap_error(int errno)
```

其中 errno 是前面返回的错误代码，该函数将返回一个错误信息。该函数主要用于将错误代码转换成错误信息输出。以下代码试图连接一个不存在的服务器，输出错误的代码和错误信息。

```
<?php
    $ldap_host = "ldap://192.168.3.44";                           //LDAP 服务器地址
    $ldap_port = "389";                                           //LDAP 服务器端口号
    $ldap_user = "";                                              //设定服务器用户名
    $ldap_pwd  = "";                                              //设定服务器密码
```

```
    $ldap_conn = ldap_connect($ldap_host, $ldap_port);                    //建立与 LDAP 服务器的连接
    @ldap_bind($ldap_conn, $ldap_user, $ldap_pwd);                        //与服务器绑定
    echo "Error number: ".ldap_errno($ldap_conn)."<br>";                  //输出错误代码
    echo "Error message: ".ldap_error($ldap_conn)."<br>";                 //输出错误信息
    echo ldap_err2str(ldap_errno($ldap_conn));                            //输出错误信息
?>
```

运行代码如下所示。

```
Error number: 81
Error message: Can't contact LDAP server
Can't contact LDAP server
```

上面的代码首先输出错误代码，然后输出错误信息，最后使用 ldap_err2str 函数将错误代码转换成错误信息。可以看到，两种方法输出的错误信息完全相同。

11.5　使用 LDAP 验证用户身份

本节将以一个实例来介绍如何使用 LDAP 验证用户身份。在实际应用中，可能会需要多个应用使用一个共同的用户名和密码来登陆。例如，一个企业使用多个系统来处理员工的日常操作，所有的系统均使用来自同一个 LDAP 目录的用户信息进行身份验证。这样，就不需要在每个系统中保存不同的密码，只需要在 LDAP 目录中保存一个密码即可。

使用 LDAP 验证用户身份的原理与上一节中介绍的绑定 LDAP 服务器的方法相同，不同的是验证时的用户名和密码来自用户的输入。完整的代码如下所示。

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER']))
{
      Header("WWW-Authenticate: Basic realm=\"login\"");
      Header("HTTP/1.0 401 Unauthorized");
}
else
{
   $ldap_host = "ldap://192.168.3.1";                                     //LDAP 服务器地址
   $ldap_port = "389";                                                    //LDAP 服务器端口号
   $ldap_user = $_SERVER['PHP_AUTH_USER'];                                //设定服务器用户名
   $ldap_pwd    = $_SERVER['PHP_AUTH_PW'];                                //设定服务器密码
   $ldap_conn = ldap_connect($ldap_host, $ldap_port)
                    or die("Can't connect to LDAP server");               //建立与 LDAP 服务器的连接
   @ldap_bind($ldap_conn, $ldap_user, $ldap_pwd);                         //与服务器绑定
   if(ldap_errno($ldap_conn)!=0)                                          //如果用户名密码错误
   {
      echo "Can't log in! ".ldap_error($ldap_conn)."<br>";
   }
   else                                                                   //如果用户名密码正确
   {
      echo "Welcome $ldap_user";
   }
}
?>
```

需要注意的是这里使用了$_SERVER['PHP_AUTH_USER']和$_SERVER['PHP_AUTH_PW']来获取用户名和密码，运行结果如图 11-8 所示。

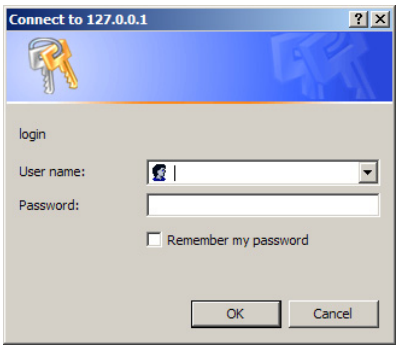

图 11-8　使用 LDAP 验证用户身份

11.6　小结

本章介绍了 PHP 与 LDAP 的应用。在企业信息化迅速发展的今天，LDAP 也越来越得到了企业用户的青睐。由于 LDAP 支持企业范围内的所有应用程序通过网络获取由 LDAP 提供的信息，有利于企业内部的信息整合。