

Create an LDAP Address Book with PHP

Preface

What is LDAP and how does it relate to PHP? This paper explains what LDAP is and shows how to create an LDAP address book with PHP...

Probably you heard much about LDAP, but have no idea what it is or how it works.

LDAP is a protocol for distributing directory information to many different resources. Most commonly it is used as a centralized address book, but it can be much more powerful depending on an organization's needs.

LDAP in its most basic form is a standard way to connect to a database. The database is optimized for read queries. Thus, it retrieves information very quickly, in contrast to additions or updates which are slower.

It is important to note that LDAP most often uses a hierarchal database, rather than a relational database to store data. Therefore, the structure is better represented with a tree than a table.

As a result, SQL syntax will be rendered unusable.

In short, LDAP is a fast way to retrieve centralized, static data containing information about people and/or resources.

Requirements

PHP v.4 (previous versions may work but are untested) compiled with support for LDAP, I.E. --with-ldap. Publicly accessible LDAP directory. Two are provided in the example.

Overview of Example

- Setup Public LDAP Server Information
- Create LDAP Query
- Connect to LDAP Server
- Process Query if Connection Was Successful
- Format Output
- Close Connection
- Make HTML Form for Search Interface
- Echo Results
- Source Code

Setup Public LDAP Server Information

The first thing we need to do is define all of the LDAP servers we might want to search.

"LDAP_NAME" = The name of the new LDAP entry.
"LDAP_SERVER" = The IP address or hostname of the new LDAP entry.
"LDAP_ROOT_DN" = The root distinguished name of the new LDAP entry.

```
<?php
    $LDAP_NAME[0]      = "Netscape Net Center";
    $LDAP_SERVER[0]    = "memberdir.netscape.com";
    $LDAP_ROOT_DN[0]  = "ou=member_directory,o=netcenter.com";

    $LDAP_NAME[1]     = "Bigfoot";
    $LDAP_SERVER[1]    = "ldap.bigfoot.com";
    $LDAP_ROOT_DN[1]  = "";

    //If no server chosen set it to 0
    {
        if (!$SERVER_ID)
        {
            $SERVER_ID = 0;
        }
    }
?>
```

Create LDAP Query

As mentioned previously, LDAP queries are not much like SQL queries. Therefore, the syntax may seem a bit limiting, but here is a basic example and one that works in this scenario.

```
//Create Query
$ldap_query = "cn=$common";
```

In our example "cn" is the attribute on which we are performing the search, and \$common is the search string variable from the search form.

LDAP query syntax allows for wildcard matching using '*'. For example, '*stanley' will find 'dan stanley'.

Connect to LDAP Server

The given function connects to an LDAP resource and assigns the connection link identifier to a variable, much like connecting to a regular database, like MySQL.

```
<?php
    //Connect to LDAP
    $connect_id = ldap_connect ($LDAP_SERVER[$SERVER_ID]);
?>
```

In our example, "\$connect_id" is the link identifier, \$LDAP_SERVER is the array of possible ldap servers, and \$SERVER_ID is the LDAP server variable from the search form.

Process Query if Connection Was Successful

If our connection was successful, we will have a valid LDAP link identifier and we can process the query.

```
<?php
    if($connect_id)
    {
        //Authenticate
        $bind_id = ldap_bind ($connect_id);

        //Perform Search
        $search_id = ldap_search ($connect_id,
                                $LDAP_ROOT_DN[$SERVER_ID],
                                $ldap_query);

        //Assign Result Set to an Array
        $result_array = ldap_get_entries ($connect_id,
                                          $search_id);
    }
    else
    {
        //Echo Connection Error
        echo "Could not connect to LDAP server: $LDAP_SERVER[$SERVER_ID]";
    }
?>
```

Once we have established a connection to the LDAP services, we must identify ourselves. Most database connections with PHP send the username and password with the connection.

However, with LDAP, credentials are unknown until a bind is performed. In our example, "\$bind_id" is the bind link identifier.

We are performing an anonymous bind to the public LDAP servers. Therefore, no argument is sent to ldap_bind() except the connection link identifier.

After we have been authorized, via bind as anonymous, we perform the query using the ldap_search() function. \$search_id is created and is our search link identifier.

Format Output

Then, we assign our result set to the variable `$result_array` using the function `ldap_get_entries()`. This will allow us to sort the information in a logical manner for display. When an LDAP search is performed, the data is returned in whatever sequence it is found. In other words, there is not an easy way to sort the data like with a common SQL ORDER BY statement. As well, many public LDAP directories do not have standard capitalization.

Since the sort is based on the ASCII value of the strings, we must format the strings with all lowercase letters to appropriately alphabetize our output.

It is important to note, that an LDAP result set is returned as a multi-dimensional array.

Thus, at this point in our script `$result_array` contains something like this:

```
$result_array[0]["cn"][0]      = "Dannie Stanley"
                        ["dn"][0]      = "uid=dannie,dc=spinweb.net"
                        ["givenname"][0] = "Dannie"
                        ["sn"][0]      = "Stanley"
                        ["mail"][0]    = "danSPAM@spinweb.net"

$result_array[1]["cn"][0]      = "Michael Reynolds"
                        ["dn"][0]      = "uid=michael,dc=spinweb.net"
                        ["givenname"][0] = "Michael"
                        ["sn"][0]      = "Reynolds"
                        ["mail"][0]    = "michaelSPAM@spinweb.net"
```

The data is stored in this format because each attribute may have more than one value (i.e. a tree structure).

For example, if my name is 'Dannie,' yet everyone knows me as 'Dan,' I could add an attribute to LDAP to store both representations of my given name like this:

```
$result_array[0]["cn"][0]      = "Dannie Stanley"
                        ["dn"][0]      = "uid=dannie,dc=spinweb.net"
                        ["givenname"][0] = "Dannie"
                        ["givenname"][0] = "Dan"
                        ["sn"][0]      = "Stanley"
                        ["mail"][0]    = "danSPAM@spinweb.net"
```

For this search, we are only worried about the first value of every attribute so we will be using 0 as the index for each attribute, except for `dn` (Distinguished Name), which contains only one value.

Here is a brief list of attributes and their meaning:

```
"cn"      = Common Name
"dn"      = Distinguished Name
"givenname" = First Name
"sn"      = Last Name
"mail"    = Email Address
```

```
<?php
//Sort results if search was successful
if ($result_array)
{
    for($i=0; $i<count($result_array); $i++)
    {
        $format_array[$i][0] = strtolower($result_array[$i]["cn"][0]);
        $format_array[$i][1] = $result_array[$i]["dn"];
        $format_array[$i][2] = strtolower($result_array[$i]["givenname"][0]);
    }
}
```

Create an LDAP Address Book with PHP

```
$format_array[$i][3] = strtolower($result_array[$i]["sn"][0]);
$format_array[$i][4] = strtolower($result_array[$i]["mail"][0]);
}

//Sort array
sort ($format_array,
      "SORT_STRING");

for ($i = 0; $i < count($format_array); $i++)
{
    $cn = $format_array[$i][0];
    $dn = $format_array[$i][1];

    $fname = ucwords($format_array[$i][2]);
    $lname = ucwords($format_array[$i][3]);

    $email = $format_array[$i][4];

    if($dn && $fname && $lname && $email)
    {
        $result_list .=
            "<A HREF=\"ldap://$LDAP_SERVER[$SERVER_ID]/$dn\">$fname $lname</A>";
        $result_list .=
            " &lt;<A HREF=\"mailto:$email\">$email</A>&gt;<BR>\n";
    }
    elseif ($dn && $cn && $email)
    {
        $result_list .= "<A HREF=\"ldap://$LDAP_SERVER[$SERVER_ID]/$dn\">$cn</A>";
        $result_list .= " &lt;<A HREF=\"mailto:$email\">$email</A>&gt;<BR>\n";
    }
}
}
else
{
    echo "Result set empty for query: $ldap_query";
}
?>
```

In our example, `$format_array` is our new array which contains the query results in a format optimized for output.

First, we loop through every element of the `$result_array` and assign it to a two-dimensional array for sorting purposes. At the same time we are using the `strtolower()` function to make all values lower-case.

Second, we sort the array using a handy little search algorithm provided by PHP called `sort()`. The first argument is the array. The second is what type of sorting to perform, as defined by the PHP documentation. Since we are sorting by string, we use "SORT_STRING".

Third, we loop through the newly formatted array and assign it to an output string named `$result_list` that contains the HTML representation of the data. It is important to note that I have used the ldap URL format for the hyper-links. An example of this looks something like this:

```
HREF = "ldap://ldap.domain.net/uid=dannie,dc=domain.net".
```

Close Connection

Now that we have all of our data contained in `$result_list`, we can safely disconnect from the LDAP connection.

```
<?php
    //Close Connection
    ldap_close ($connect_id);
?>
```

Make HTML Form for Search Interface

Finally, we get to the HTML output of the script.

This set of code prints out the form that is used for performing the searches.

```
<?php
    //Make Form
    echo "<CENTER><FORM ACTION=\"\$PHP_SELF\" METHOD=\"GET\">";
    echo "Search in:<SELECT NAME=\"SERVER_ID\">";

    //Loop Through and Create SELECT OPTIONS
    for ($i = 0; $i < count($LDAP_NAME); $i++)
    {
        echo "<OPTION VALUE=\"$i\">".$LDAP_NAME[$i]."</OPTION>";
    }

    echo "</SELECT><BR>";
    echo "Search for:<INPUT TYPE=\"text\" NAME=\"common\">";
    echo "<INPUT TYPE=\"submit\" NAME=\"lookup\" VALUE=\"go\"><BR>";
    echo "(You can use * for wildcard searches, ";
    echo "ex. * Stanley will find all Stanleys)<BR>";
    echo "</FORM></CENTER>";
?>
```

The only portions of this code that is interpreted is `$PHP_SELF` which is a global constant for the name of the script itself and the loop that creates the SELECT box from our `$LDAP_NAME` variable.

Echo Results

Now that all of the work has been done, we print out the result set. If no results were returned, a message is given stating the same.

```
<?php
    //Echo Results
    if($result_list)
    {
        echo "<CENTER><TABLE BORDER=\"1\" CELLSPACING=\"0\" CELLPADDING=\"10\"
            BGCOLOR=\"#FFFFFFA\" WIDTH=\"450\"><TR><TD>$result_list</TD></TR>
            </TABLE></CENTER>";
    }
    else
    {
        echo "No Results";
    }
?>
```

Source Code

Here is the complete source code.

```
<?php
$LDAP_NAME[0]      = "Netscape Net Center";
$LDAP_SERVER[0]    = "memberdir.netscape.com";
$LDAP_ROOT_DN[0]   = "ou=member_directory,o=netcenter.com";

$LDAP_NAME[1]      = "Bigfoot";
$LDAP_SERVER[1]    = "ldap.bigfoot.com";
$LDAP_ROOT_DN[1]   = "";

//If no server chosen set it to 0
{
    if (!$SERVER_ID)
    {
        $SERVER_ID = 0;
    }

    //Create Query
    $ldap_query = "cn=$common";

    //Connect to LDAP
    $connect_id = ldap_connect ($LDAP_SERVER[$SERVER_ID]);

    if($connect_id)
    {
        //Authenticate
        $bind_id = ldap_bind ($connect_id);

        //Perform Search
        $search_id = ldap_search ($connect_id,
                                $LDAP_ROOT_DN[$SERVER_ID],
                                $ldap_query);

        //Assign Result Set to an Array
        $result_array = ldap_get_entries ($connect_id,
                                          $search_id);
    }
    else
    {
        //Echo Connection Error
        echo "Could not connect to LDAP server: $LDAP_SERVER[$SERVER_ID]";
    }

    //Sort results if search was successful
    if ($result_array)
    {
        for($i=0; $i<count($result_array); $i++)
        {
            $format_array[$i][0] = strtolower($result_array[$i]["cn"][0]);
            $format_array[$i][1] = $result_array[$i]["dn"];
            $format_array[$i][2] = strtolower($result_array[$i]["givenname"][0]);
            $format_array[$i][3] = strtolower($result_array[$i]["sn"][0]);
            $format_array[$i][4] = strtolower($result_array[$i]["mail"][0]);
        }

        //Sort array
        sort ($format_array,
              "SORT_STRING");
    }
}
```

Create an LDAP Address Book with PHP

```
for ($i = 0; $i < count($format_array); $i++)
{
    $cn = $format_array[$i][0];
    $dn = $format_array[$i][1];

    $fname = ucwords($format_array[$i][2]);
    $lname = ucwords($format_array[$i][3]);

    $email = $format_array[$i][4];

    if($dn && $fname && $lname && $email)
    {
        $result_list .= "<A HREF=\"ldap://$LDAP_SERVER[$SERVER_ID]/$dn\">
                        $fname $lname</A>";
        $result_list .= " &lt;<A HREF=\"mailto:$email\">$email</A>&gt;<BR>\n";
    }
    elseif ($dn && $cn && $email)
    {
        $result_list .= "<A HREF=\"ldap://$LDAP_SERVER[$SERVER_ID]/$dn\">$cn</A>";
        $result_list .= " &lt;<A HREF=\"mailto:$email\">$email</A>&gt;<BR>\n";
    }
}
}
else
{
    echo "Result set empty for query: $ldap_query";
}

//Close Connection
ldap_close ($connect_id);

//Make Form
echo "<CENTER><FORM ACTION=\"$PHP_SELF\" METHOD=\"GET\">";
echo "Search in:<SELECT NAME=\"$SERVER_ID\">";

//Loop Through and Create SELECT OPTIONS
for ($i = 0; $i < count($LDAP_NAME); $i++)
{
    echo "<OPTION VALUE=\"$i\">".$LDAP_NAME[$i]."</OPTION>";
}

echo "</SELECT><BR>";
echo "Search for:<INPUT TYPE=\"text\" NAME=\"common\">";
echo "<INPUT TYPE=\"submit\" NAME=\"lookup\" VALUE=\"go\"><BR>";
echo "(You can use * for wildcard searches, ";
echo "ex. * Stanley will find all Stanleys)<BR>";
echo "</FORM></CENTER>";

//Echo Results
if($result_list)
{
    echo "<CENTER><TABLE BORDER=\"1\" CELLSPACING=\"0\" CELLPADDING=\"10\"
        BGCOLOR=\"#FFFFFFA\" WIDTH=\"450\"><TR><TD>$result_list</TD></TR>
        </TABLE></CENTER>";
}
else
{
    echo "No Results";
}
}
?>
```