FREE eBook

LEARNING

ldap

Free unaffiliated eBook created from
Stack Overflow contributors.

#ldap

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ldap

It is an unofficial and free ldap ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ldap.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with ldap

Remarks

This section provides an overview of what ldap is, and why a developer might want to use it.

It should also mention any large subjects within ldap, and link out to the related topics. Since the Documentation for ldap is new, you may need to create initial versions of those related topics.

Examples

Setting up PHP to work with LDAP

After you configured your LDAP server correctly, now we want to connect. For example by using PHP.

- DN = distinguished name. This means, in which part of the database are you working. Can be a user or a group (or even config settings).
- Entry: an entity, for example a user.
- Attribute: something inside an entry, for example name, phone number and email address.

Connecting

First, we define the following:

```
$server = "server.example.com";  //this is the LDAP server you're connecting with
$ds = ldap_connect("ldaps://$server", 636); //always connect securely via LDAPS when possible
```

Now, we are connected. Next thing we want, is to let the server know that we are a trustworthy person. The easiest solution is to use the root DN OR an existing user with proper permissions to view the database (probably every user in the database can do this by default). We authenticate using the bind function.

Set options

First, we declare these options. Depending on your server configuration, you can leave this out.

```
ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);
ldap_set_option($ds, LDAP_OPT_REFERRALS, 0);
```

Binding

Assume you are using admin and password is pass123notsafe

```
$dn = "uid=admin,cn=users,dc=server,dc=example,dc=com";
$pass = "pass123notsafe";
$ldapbind = ldap_bind($ds, $dn, $pass); //this is the point we are authenticating
```

That is very nice. We are in. Now we can perform a lot of different operations. For example, we can search, read, write and modify users and even groups.

Searching

Imagine we want to display all members of the group "users".

```
$dn = "cn=users,dc=server,dc=example,dc=com"; //very important: in which part of your database
are you looking
$filter = "uid=*"; //don't filter anyone out (every user has a uid)
$sr = ldap_search($ds, $dn, $filter) or die ("bummer"); //define your search scope

$results = ldap_get_entries($ds, $sr); //here we are pulling the actual entries from the
search we just defined
var_dump($results); //will give you all results is array form.
```

You can use foreach loops to display the data in a nice way.

Okay, got this? Now proceed with some filtering. We want to display only users in the group "bikeowners" who have registered an email address. It's important to know that *all* users are in cn=users. Next to this, they can also be member of other groups.

```
//did the connecting and binding

$dn = "cn=bikeowners,cn=groups,dc=server,dc=example,dc=com"; //note the extra "cn=groups" for
looking in a group that is not "users"
$filter = "email=*"; //email address must be set but can be anything
$sr = ldap_search($ds, $dn, $filter) or die ("bummer"); //define your search scope
```

And now proceed with `ldap_get_entries` again.

Efficiency in getting entries

An example in which efficiency is addressed, increasingly important with big databases with a lot of attributes per entry. Especially when you are storing pictures in the attribute jpegphoto, it may significantly reduce your loading time when you selectively pull your entries.

Imagine we want to look for users that are in the group "bikeowners". There is a lot of information stored inside these entries, let's say they have the following attributes: cn, uid, name, displayname, mail, initials, mobile, telephonenumber, street, postaladdress, postalcode and jpegphoto. Now we only need their uid, name and initials.

For this, we use an optional 4th parameter of ldap_search:

```
$dn = "cn=bikeowners,cn=groups,dc=server,dc=example,dc=com";
$filter = "uid=*"; //
$justhese = array("uid", "name", "initials");
$sr = ldap_search($ds, $dn, $filter, $justthese) or die ("bummer"); //define your search scope
```

And voila, running `ldap_get_entries` will give you only these data.

Advanced filtering

Of course, you can pull a whole database with LDAP, and process this further in PHP. However, just as with MySQL it's much more efficient to do the processing server-side. To demonstrate this, we can use an advanced filter.

Take one of above examples as starting point, but change $filter according to the line below. In our example, we want to display the data of users who are active. Normally, the attribute `shadowexpire` is used to store this information. However, this may differ among various LDAP systems. Not only we want to display the users that are active, but *also* their name must start with an "a" and they must live in Amsterdam.

Basically we want to do three things:

1. Easiest: must live in Amsterdam. In this example the place of residence is stored in attribute 'postaladdress'

```
$filter= "postaladdress=Amsterdam";
```

2. Name must start with an "a". In this example, UID is composed from the name, so:

```
$filter= "uid=a*";
```

3. User must be active. This value is stored in the default attribute `shadowexpire`, with value `-1`. Depending on your server configuration, `shadowexpire` can hold a myriad of values, even dates are possible. If a user is inactive, `shadowexpire` will be `1`. To be sure we get all users except for those who are really inactive, we don't choose to filter on `shadowexpire = -1`. Instead we say we *don't* want them to be inactive.

```
$filter= "(!(shadowexpire=1))"; //NOT is represented with "!"
```

Now the most interesting part: combine all three examples. We can do this with brackets, AND, OR and NOT expressions

```
$filter= "(&(postaladdress=Amsterdam)(uid=a*)(!(shadowexpire=1)))";
```

Finally, we could build in a OR statement using "|", for example if we want all users starting with "a" or "b"

```
$filter= "(|(uid=a*)(uid=n*))";
```

You can combine endlessly and build quite impressive filters, enjoy trying!

Read Getting started with ldap online: https://riptutorial.com/ldap/topic/5576/getting-started-with-ldap

Credits

S. No	Chapters	Contributors
1	Getting started with ldap	Community, SJDS