OpenLDAP

A developer's perspective

Gerrit Beine

mail@gerritbeine.com

Motivation:
Warum dieser Vortrag?

Ziele des Vortrags

- OpenLDAP als objektbasierte Datenbank verstehen lernen
- CRUD Operationen aus Java und Perl ausführen
- Grundlegendes Verständnis von Schema Dateien
- Suchen im Verzeichnis, Aliase und extensibleObjects

Grundlegendes zu (Open)LDAP

LDAP

- LDAP: Lightweight Directory Access Protocol
- Version 2: RFC 1777-1778, 1823, 1959-1960
- Version 3: RFC 2251-2253, 4510-4519
- Frontend zu hierarchischen Datenbanken
- Optimiert für Suchen und Finden von hierarchisch abgelegten Informationen
- Basiert auf X.500 (DAP), vereinfacht dieses jedoch für TCP/IP-Netze

Begriffswelt

- bind – Information, wie man sich beim LDAP anmelden will: anonym oder via authorisiertem DN
- dn – distinguished name;
  eindeutiger Identifikator für Objekte im Verzeichnis
- objectClass – Strukturbeschreibung von Objekten im Verzeichnis
- structural – Spezielle Objektklassen, nur eine pro Objekt
- auxiliary – Optionale, ergänzende Objektklassen
- schema – Beschreibung von Objektklassen und Attributen
- scope – Suchbereich im Verzeichnis (base, one, subtree)
- baseDn – Verzeichnisknoten, der Basis für Operationen ist
- LDIF – LDAP Data Interchange Format,
  Dateiformat für den Austausch von Verzeichnisdaten in Textform

LDAP-Zugriff von Java und Perl

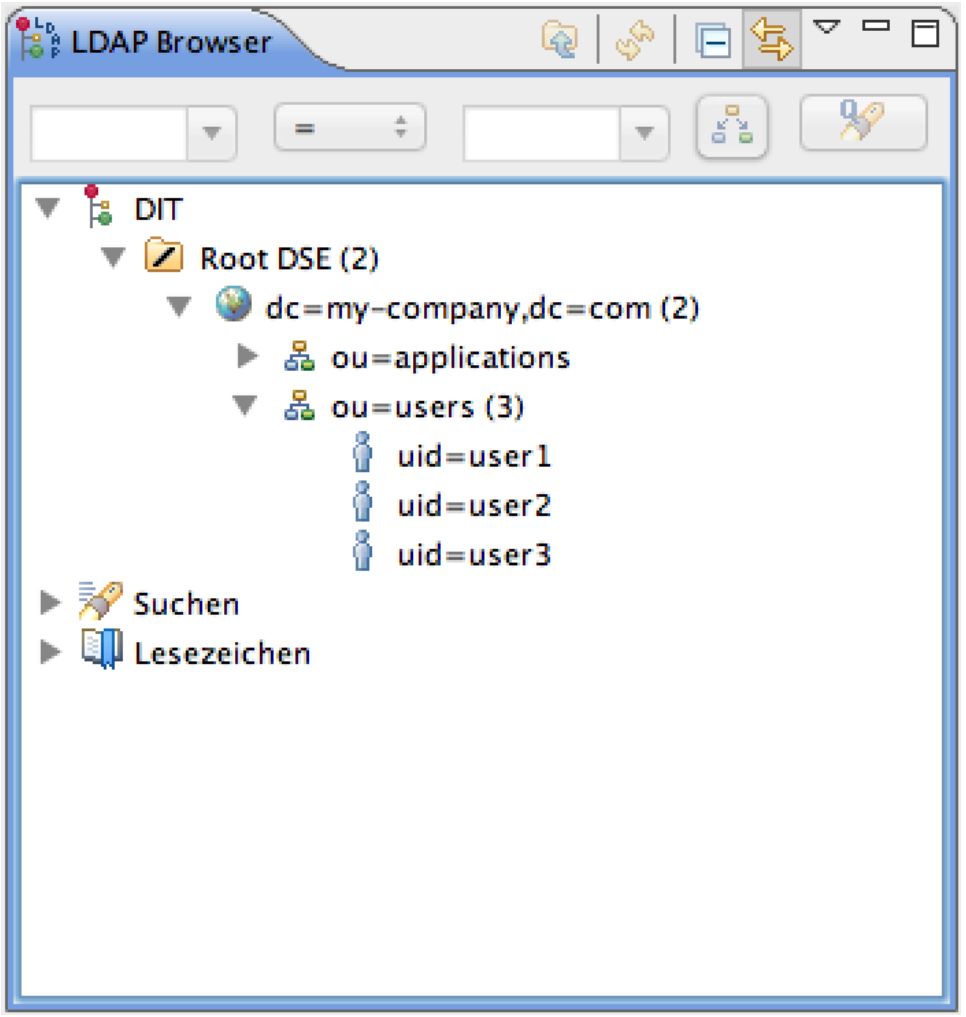
Perl & Net::LDAP

- Net::LDAP bzw. Perl LDAP: http://ldap.perl.org/
- Objektorienterte API für den Zugriff auf LDAP-Dienste
- Verbindungsaufbau und Authentifizierung
- Schnittstelle für CRUD-Operationen im Verzeichnis
- LDIF-Schnittstelle
- Schema-Bearbeitung

Suchen mit Perl

```perl
#!/usr/bin/perl

use strict;
use warnings;

use Net::LDAP;
use Data::Dumper;

my $host = '172.16.166.129';
my $base = 'ou=users,dc=my-company,dc=com';
my $filter = '(&(objectClass=person)(uid=user*))     ';
my $message;

my $ldap = new Net::LDAP($host);

# Anonym arbeiten, keine Authentifizierung
$message = $ldap->bind();

# Suchen unter der angegebenen base mit dem Filter
$message = $ldap->search(base => $base, filter => $filter);

foreach my $entry ($message->entries) {
  # Jeder gefundene Eintrag hat im Hash 'asn' ein Attribut 'objectName',
  # in dem der DN enthalten ist
  print $entry->{'asn'}->{'objectName'}, "\n";
}
```

Ergebnis der Suche



- Ergebnis der Scriptausführung:

```
gbeine$ perl -w search.pl
uid=user1,ou=users,dc=my-company,dc=com
uid=user2,ou=users,dc=my-company,dc=com
uid=user3,ou=users,dc=my-company,dc=com
```

Anlegen mit Perl - I

```
 1 #!/usr/bin/perl
 2
 3 use strict;
 4 use warnings;
 5
 6 use Net::LDAP;
 7 use Data::Dumper;
 8
 9 my $host = '172.16.166.129';
10 my $base = 'ou=users,dc=my-company,dc=com';
11 my $message;
12
13 my $ldap = new Net::LDAP($host);
14
15 # Schreiben erfordert Authentifizierung
16 $message = $ldap->bind(
17         'cn=Administrator,dc=my-company,dc=com',
18         password => 'linux'
19 );
20
21 $message->code && warn "Fehler bei Auth: ", $message->error;
22
```

Anlegen mit Perl - II

```
23 # Anlegen eines vierten Nutzers
24 $message = $ldap->add( 'uid=user4,'.$base,
25         attr => [
26             'uid'  => 'user4',
27             'uidNumber' => '4',
28             'gidNumber'  => '1',
29             'cn'   => 'User',
30             'sn'   => 'Four',
31             'homeDirectory' => '/home/user4',
32             'objectclass' => [ 'top', 'person', 'posixAccount' ],
33     ]
34 );
35
36 $message->code && warn "Fehler beim Anlegen: ", $message->error;
```

Ändern mit Perl

```perl
#!/usr/bin/perl

use strict;
use warnings;

use Net::LDAP;

my $host = '172.16.166.129';
my $base = 'ou=users,dc=my-company,dc=com';
my $message;

my $ldap = new Net::LDAP($host);

# Schreiben erfordert Authentifizierung
$message = $ldap->bind(
        'cn=Administrator,dc=my-company,dc=com',
        password => 'linux'
);

$message->code && warn "Fehler bei Auth: ", $message->error;

# Attribute ändern
$message = $ldap->modify(
        'uid=user4,'.$base, replace => { 'sn' => 'Five' } );

$message->code && warn "Fehler beim Ändern: ", $message->error;
```

Umbenennen mit Perl

```perl
#!/usr/bin/perl

use strict;
use warnings;

use Net::LDAP;

my $host = '172.16.166.129';
my $base = 'ou=users,dc=my-company,dc=com';
my $message;

my $ldap = new Net::LDAP($host);

# Schreiben erfordert Authentifizierung
$message = $ldap->bind(
        'cn=Administrator,dc=my-company,dc=com',
        password => 'linux'
);

$message->code && warn "Fehler bei Auth: ", $message->error;

# DN ändern
$message = $ldap->moddn( 'uid=user4,'.$base,
        newrdn => 'uid=user5', deleteoldrdn => 1);

$message->code && warn "Fehler beim Ändern: ", $message->error;
```

Löschen mit Perl

```perl
#!/usr/bin/perl

use strict;
use warnings;

use Net::LDAP;

my $host = '172.16.166.129';
my $base = 'ou=users,dc=my-company,dc=com';
my $message;

my $ldap = new Net::LDAP($host);

# Schreiben erfordert Authentifizierung
$message = $ldap->bind(
        'cn=Administrator,dc=my-company,dc=com',
        password => 'linux'
);

$message->code && warn "Fehler bei Auth: ", $message->error;

# Löschen
$message = $ldap->delete( 'uid=user5,'.$base );

$message->code && warn "Fehler beim Löschen: ", $message->error;
```

Java & LDAP

- Viele Wege führen nach Rom:
- javax.naming.ldap – Zugriff auf LDAP via Java-Namendienste-API
- Java LDAP – Ehemals Novell, jetzt an OpenLDAP übergeben
  http://www.openldap.org/jldap/
- Mozilla LDAP SDK – Ehemals von Netscape entwickelt, seit 2008 tot
  http://www.mozilla.org/directory/javasdk.html
- SPRING LDAP – LDAP-Zugriff via JDBC-Templates
  http://www.springsource.org/ldap

Suchen mit Java - I

```
 3 import java.util.Hashtable;
 4
 5 import javax.naming.Context;
 6 import javax.naming.NamingEnumeration;
 7 import javax.naming.directory.DirContext;
 8 import javax.naming.directory.InitialDirContext;
 9 import javax.naming.directory.SearchControls;
10 import javax.naming.directory.SearchResult;
11
12
13 public class Search {
14
15         private final static String ldapAdServer =
16                         "ldap://172.16.166.129:389";
17         private final static String ldapSearchBase =
18                         "ou=users,dc=my-company,dc=com";
19         private final static String ldapFilter =
20                         "(&(objectClass=person)(user=user*))";
21
22         public static void main(String[] args) {
23                 Hashtable<String, Object> env =
24                                 new Hashtable<String, Object>();
25                 // Einfaches bind, ohne Authentifizierung
26                 env.put(Context.SECURITY_AUTHENTICATION, "simple");
27                 env.put(Context.INITIAL_CONTEXT_FACTORY,
28                         "com.sun.jndi.ldap.LdapCtxFactory");
29                 env.put(Context.PROVIDER_URL, ldapAdServer);
```

Suchen mit Java - II

```
30
31          try {
32                  DirContext ctx = new InitialDirContext(env);
33
34                  SearchControls searchControls =
35                                      new SearchControls();
36                  searchControls.setSearchScope(SearchControls.SUBTREE_SCOPE );
37
38                  NamingEnumeration<SearchResult> results =
39                        ctx.search(ldapSearchBase, ldapFilter, searchControls);
40
41                  SearchResult searchResult = null;
42                  while(results.hasMoreElements()) {
43                        searchResult = (SearchResult) results.nextElement();
44                        System.out.println(searchResult.getNameInNamespace());
45                  }
46          } catch (Exception e) {
47                   System.err.println(e);
48          }
49
50      }
51  }
```

Arbeiten mit LDAP-Schema

Aufbau von Schema Definitionen

```
# Attributtypen

AttributeType ( <OID> NAME '<name>'
  DESC '<description>'
  EQUALITY <matchingRule>
  SUBSTR <matchingRule>
  SYNTAX <syntaxOID>
)

AttributeType ( <OID> NAME '<name>'
  DESC '<description>'
  SUP <superType>
)


# Klassen

objectclass ( <OID> NAME '<name>'
  DESC '<description>'
  SUP <superClass> # oder:
  < STRUCTURAL | ABSTRACT | AUXILIARY >
  MUST ( <attribute1> $ <attribute2> )
  MAY <attribute3> )
```

Syntax-Typen

Name	Object Identifier	Beschreibung
boolean	1.3.6.1.4.1.1466.115.121.1.7	Wahrheitswerte
directoryString	1.3.6.1.4.1.1466.115.121.1.15	Unicode (UTF-8) Zeichenkette
distinguishedName	1.3.6.1.4.1.1466.115.121.1.12	LDAP DN's
integer	1.3.6.1.4.1.1466.115.121.1.27	Zahlen
numericString	1.3.6.1.4.1.1466.115.121.1.36	Numerische Zeichenketten
OID	1.3.6.1.4.1.1466.115.121.1.38	Object Identifier
octetString	1.3.6.1.4.1.1466.115.121.1.40	BLOB's

- An die Syntax kann in geschweiften Klammern noch eine Länge angehängt werden:
- 1.3.6.1.4.1.1466.115.121.1.15{256}
  steht für eine 256 Zeichen lange Zeichenkette

Matching Rules

Name	Anwendbarkeit	Beschreibung
booleanMatch	Gleichheit	
caseIgnoreMatch	Gleichheit	Ignoriert Groß-/Kleinschreibung und Leerzeichen
caseIgnoreOrderingMatch	Sortierung	Ignoriert Groß-/Kleinschreibung und Leerzeichen
caseIgnoreSubstringsMatch	Teilzeichenketten	Ignoriert Groß-/Kleinschreibung und Leerzeichen
caseExactMatch	Gleichheit	Ignoriert Leerzeichen
caseExactOrderingMatch	Sortierung	Ignoriert Leerzeichen
caseExactSubstringsMatch	Teilzeichenketten	Ignoriert Leerzeichen
distinguishedNameMatch	Gleichheit	
integerMatch	Gleichheit	
integerOrderingMatch	Sortierung	
numericStringMatch	Gleichheit	
numericStringOrderingMatch	Sortierung	
numericStringSubstringsMatch	Teilzeichenketten	
octetStringMatch	Gleichheit	
octetStringOrderingStringMatch	Sortierung	
octetStringSubstringsStringMatch	Teilzeichenketten	
objectIdentiferMatch	Gleichheit	

Beispiel-Schema

```
# Basis OID für alle LDAP-Typen meines Unternehmens
objectIdentifier GBGmbH 1.3.6.1.4.1.40207;

# Empfohlene Aufteilung für SNMP und LDAP
objectIdentified MySNMP GBGmbH:1
objectIdentified MyLDAP GBGmbH:2
objectIdentifier MyAttributeType MyLDAP:1
objectIdentifier MyObjectClass MyLDAP:2

# Ein Attribut, von name geerbt
attributetype ( MyAttributeType:1 # ergibt 1.3.6.1.4.1.40207.2.1.1
    NAME 'applicationName'
    DESC 'Name of an Application'
    SUP name SINGLE-VALUE
)

# Eine Struktur-Klasse
objectclass ( MyObjectClass:1 # ergibt 1.3.6.1.4.1.40207.2.2.1
    NAME 'application'
    DESC 'An application object'
    STRUCTURAL
    MUST ( applicationName )
)
```

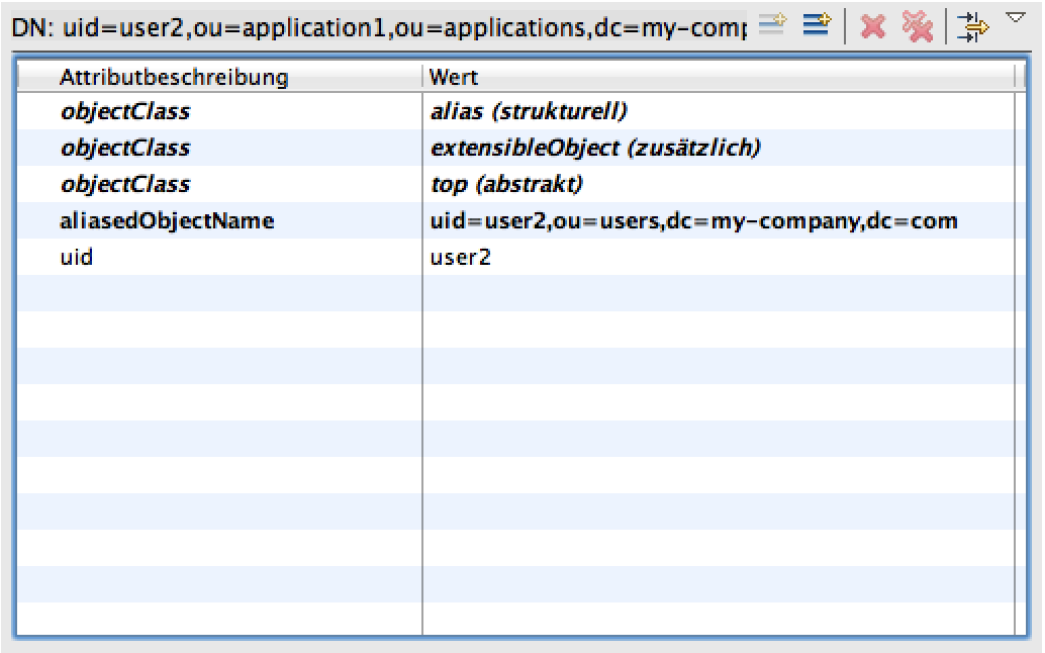
Aufbau eines Verzeichnisses

Szenario

- Benutzerverwaltung in Kombination
- Authentifizierung: LDAP
- Generisches Rechtemanagement: SQL-Datenbank
- Teilweise datenbezogene Berechtigungen bis auf Objektebene für viele verschiedene Anwendungen
- Skaliert mit Tabellen nur schwer

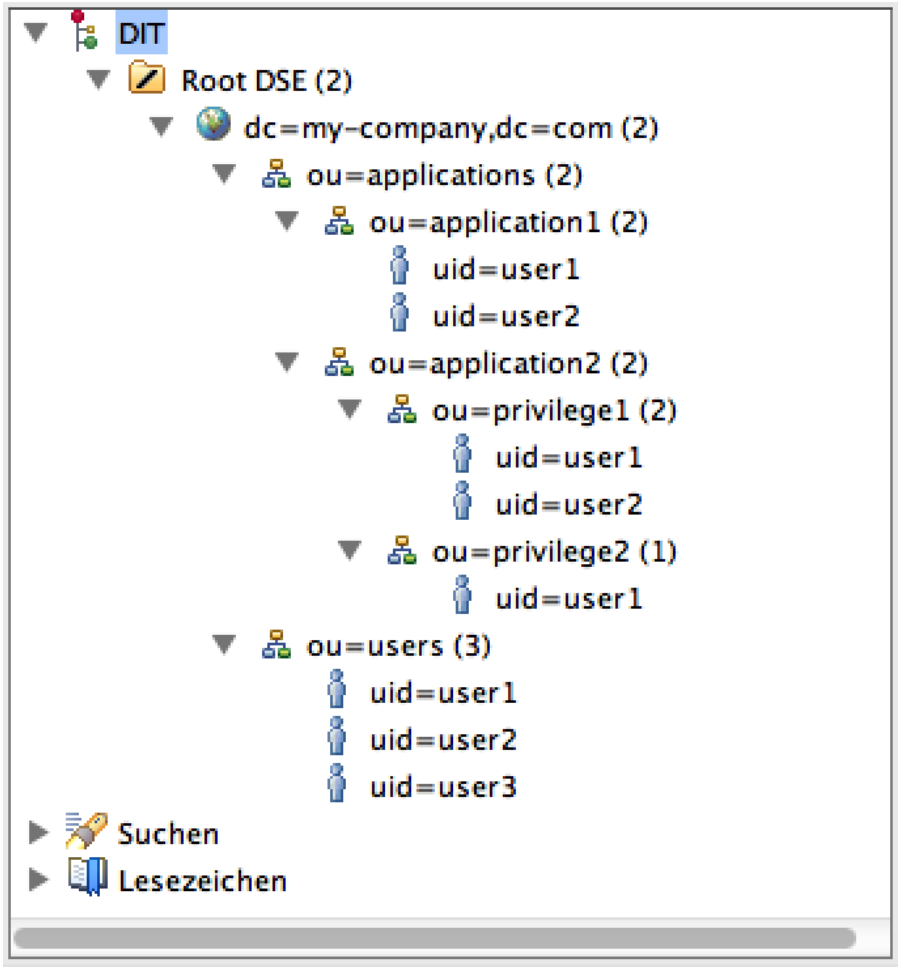
- Idee: Erweiterte Nutzung des LDAP für Berechtigungen
- Aliase auf Nutzer ersetzen JOIN ;-)
- Aliase in LDAP entsprechen Symlinks in Unix-Dateisystemen
- extensibleObjects lassen Aliase transparent erscheinen

Anlegen eines Alias



- objectClass alias auswählen
- Wenn das neue Objekt transparent sein soll: ObjectClass extensibleObject auswählen
- Wenn extensibleObject möglich, gleiches DN-Attribut nutzen, wie im Original
- Aliases sind geringfügig langsamer als echte Objekte

Die Baumstruktur mit Aliases



- user1 und user2 unterhalb von ou=applications sind Links auf die echten Einträge unter ou=users
- Änderungen an Einträgen unter ou=users wirken sich direkt aus
- Suchen nach objectClass=person unter ou=applications liefert user1 und user2 von ou=users
- Für Suchen muss deref=always gesetzt sein

Wie funktioniert der „JOIN“?

- Über den Distinguished Name!
- Original:
  dn=uid=user1,ou=users,dc=my-company,dc=com
- Alias:
  dn=uid=user1,ou=application1,ou=applications,dc=my-company,dc=com
- Suche nach (&(objectClass=person)(uid=user1)) unterhalb von ou=application1,ou=applications,dc=my-company,dc=com liefert die Information, ob user1 das Recht hat, application1 zu nutzen
- Suche nach (&(objectClass=person)(uid=user2)) unterhalb von ou=privilege1,ou=application2,ou=applications,dc=my-company,dc=com liefert die Information, ob user2 das Recht für privilege1 in application2 zu nutzen
- Die Existenz eines Suchergebnisses bedeutet, das Recht ist vorhanden

Weitere Anwendungsmöglichkeiten, Beachtenswertes

- Verfeinerung der Berechtigungen über Erweiterung der Hierarchie möglich – Funktionsprinzip bleibt identisch
- Kein Tabellen-Jonglieren bei Authorisierung von Nutzern – Jede Anwendung kennt ihren eigenen Subtree im Verzeichnis und nur diesen!
- Teilbäume können repliziert werden

- Indizierung des Verzeichnisbaumes beachten – Aliase sind langsamer als echte Objekte
- extensibleObject mit Vorsicht genießen – semistrukturierte Daten erfordert Anwendungslogik
- Skalierungsmöglichkeiten berücksichtigen – OpenLDAP liest schnell, schreibt aber verhältnismäßig langsam

Literaturempfehlungen

- OpenLDAP: http://www.openldap.org/doc/admin24/index.html
- LDAP für Java Entwickler: http://www.amazon.de/LDAP-für-Java-Entwickler-3-Auflage/dp/3939084077/
- LDAP verstehen, OpenLDAP einsetzen: http://www.amazon.de/LDAP-verstehen-OpenLDAP-einsetzen-Praxiseinsatz/dp/3898642631/
- LDAPman home page: http://ldapman.org/
- LDAP Linux HOWTO: http://tldp.org/HOWTO/LDAP-HOWTO/

Viel Spaß noch auf der FrOSCon 2012!