Lightweight Directory Access Protocol

Eine Einführung mit openLDAP

Michael Dienert, Peter Maaß

Walther-Rathenau-Gewerbeschule
Freiburg

6. Januar 2014

Inhalt

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email $\Rightarrow$ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack $\Rightarrow$ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

Entwicklung von LDAP

- Das **Ligthweight Directory Access Protocol (LDAP)** benötigt lediglich TCP/IP $\Rightarrow$ weite Verbreitung
- LDAP implementiert nur einen Teil der X.500-Protokollspezifikation $\Rightarrow$ Fehlendes wird emuliert.
- LDAP wurde 1993 in der RFC 1487 beschrieben und geht auf Arbeiten an der *Universität von Michigan* zurück.

Entwicklung von LDAP

- Das **Ligthweight Directory Access Protocol (LDAP)** benötigt lediglich TCP/IP $\Rightarrow$ weite Verbreitung
- LDAP implementiert nur einen Teil der X.500-Protokollspezifikation $\Rightarrow$ Fehlendes wird emuliert.
- LDAP wurde 1993 in der RFC 1487 beschrieben und geht auf Arbeiten an der *Universität von Michigan* zurück.

Entwicklung von LDAP

- Das **Ligthweight Directory Access Protocol (LDAP)** benötigt lediglich TCP/IP $\Rightarrow$ weite Verbreitung
- LDAP implementiert nur einen Teil der X.500-Protokollspezifikation $\Rightarrow$ Fehlendes wird emuliert.
- LDAP wurde 1993 in der RFC 1487 beschrieben und geht auf Arbeiten an der *Universität von Michigan* zurück.

Entwicklung von LDAP

- Das **Ligthweight Directory Access Protocol (LDAP)** benötigt lediglich TCP/IP $\Rightarrow$ weite Verbreitung
- LDAP implementiert nur einen Teil der X.500-Protokollspezifikation $\Rightarrow$ Fehlendes wird emuliert.
- LDAP wurde 1993 in der RFC 1487 beschrieben und geht auf Arbeiten an der *Universität von Michigan* zurück.

Entwicklung von LDAP

- Das **Ligthweight Directory Access Protocol (LDAP)** benötigt lediglich TCP/IP $\Rightarrow$ weite Verbreitung
- LDAP implementiert nur einen Teil der X.500-Protokollspezifikation $\Rightarrow$ Fehlendes wird emuliert.
- LDAP wurde 1993 in der RFC 1487 beschrieben und geht auf Arbeiten an der *Universität von Michigan* zurück.

Entwicklung von LDAP

- Das **Ligthweight Directory Access Protocol (LDAP)** benötigt lediglich TCP/IP $\Rightarrow$ weite Verbreitung
- LDAP implementiert nur einen Teil der X.500-Protokollspezifikation $\Rightarrow$ Fehlendes wird emuliert.
- LDAP wurde 1993 in der RFC 1487 beschrieben und geht auf Arbeiten an der *Universität von Michigan* zurück.

Bestandteile von LDAP

Information Model Besser wäre hier der Begriff *Data Model* zu verwenden. Das Data (Information) Model beschreibt, wie die Informationen in einem LDAP-System repräsentiert werden (s.u.). Das Information Model beschreibt *nicht* wie die eigentliche Backend-Datenbank aussieht und somit nicht wie die Daten gespeichert werden, sondern nur wie darauf *zugegriffen* wird.

Naming Model Dieses Modell beschreibt die Namenskonventionenen im Datenmodell. Beispiel: *'dc=example,dc=com'*

Functional Model bei Lese-, Such- und Schreibzugriffen wird das **Functional Model** benutzt.

Security Model Mit Hilfe dieses Modells lässt sich sehr fein einstellen, wer was mit welchen Daten anstellen darf.

Bestandteile von LDAP

Information Model Besser wäre hier der Begriff *Data Model* zu verwenden. Das Data (Information) Model beschreibt, wie die Informationen in einem LDAP-System repräsentiert werden (s.u.). Das Information Model beschreibt *nicht* wie die eigentliche Backend-Datenbank aussieht und somit nicht wie die Daten gespeichert werden, sondern nur wie darauf *zugegriffen* wird.

Naming Model Dieses Modell beschreibt die Namenskonventionenen im Datenmodell. Beispiel: *'dc=example,dc=com'*

Functional Model bei Lese-, Such- und Schreibzugriffen wird das **Functional Model** benutzt.

Security Model Mit Hilfe dieses Modells lässt sich sehr fein einstellen, wer was mit welchen Daten anstellen darf.

Bestandteile von LDAP

Information Model Besser wäre hier der Begriff *Data Model* zu verwenden. Das Data (Information) Model beschreibt, wie die Informationen in einem LDAP-System repräsentiert werden (s.u.). Das Information Model beschreibt *nicht* wie die eigentliche Backend-Datenbank aussieht und somit nicht wie die Daten gespeichert werden, sondern nur wie darauf *zugegriffen* wird.

Naming Model Dieses Modell beschreibt die Namenskonventionenen im Datenmodell. Beispiel: *'dc=example,dc=com'*

Functional Model bei Lese-, Such- und Schreibzugriffen wird das **Functional Model** benutzt.

Security Model Mit Hilfe dieses Modells lässt sich sehr fein einstellen, wer was mit welchen Daten anstellen darf.

Bestandteile von LDAP

Information Model Besser wäre hier der Begriff *Data Model* zu verwenden. Das Data (Information) Model beschreibt, wie die Informationen in einem LDAP-System repräsentiert werden (s.u.). Das Information Model beschreibt *nicht* wie die eigentliche Backend-Datenbank aussieht und somit nicht wie die Daten gespeichert werden, sondern nur wie darauf *zugegriffen* wird.

Naming Model Dieses Modell beschreibt die Namenskonventionenen im Datenmodell. Beispiel: *'dc=example,dc=com'*

Functional Model bei Lese-, Such- und Schreibzugriffen wird das **Functional Model** benutzt.

Security Model Mit Hilfe dieses Modells lässt sich sehr fein einstellen, wer was mit welchen Daten anstellen darf.

Bestandteile von LDAP

Information Model Besser wäre hier der Begriff *Data Model* zu verwenden. Das Data (Information) Model beschreibt, wie die Informationen in einem LDAP-System repräsentiert werden (s.u.). Das Information Model beschreibt *nicht* wie die eigentliche Backend-Datenbank aussieht und somit nicht wie die Daten gespeichert werden, sondern nur wie darauf *zugegriffen* wird.

Naming Model Dieses Modell beschreibt die Namenskonventionenen im Datenmodell. Beispiel: *'dc=example,dc=com'*

Functional Model bei Lese-, Such- und Schreibzugriffen wird das **Functional Model** benutzt.

Security Model Mit Hilfe dieses Modells lässt sich sehr fein einstellen, wer was mit welchen Daten anstellen darf.

Bestandteile von LDAP

Information Model Besser wäre hier der Begriff *Data Model* zu verwenden. Das Data (Information) Model beschreibt, wie die Informationen in einem LDAP-System repräsentiert werden (s.u.). Das Information Model beschreibt *nicht* wie die eigentliche Backend-Datenbank aussieht und somit nicht wie die Daten gespeichert werden, sondern nur wie darauf *zugegriffen* wird.

Naming Model Dieses Modell beschreibt die Namenskonventionenen im Datenmodell. Beispiel: *'dc=example,dc=com'*

Functional Model bei Lese-, Such- und Schreibzugriffen wird das **Functional Model** benutzt.

Security Model Mit Hilfe dieses Modells lässt sich sehr fein einstellen, wer was mit welchen Daten anstellen darf.

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT).**
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Das LDAP/X.500 Datenmodell: Directory Information Trees

- Beide, LDAP und X.500 speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen*, *Vererbung* und *Polymorphie*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (Directory Entries). Sie entsprechen *Objekten* (Instanzen) oder einfach: *Datensätzen*.
- Analog zu einem Dateisystem, gibt es
  - *Containerobjekte*, die andere Objekte enthalten können. Diese entsprechen Verzeichnissen.
  - *Blattobjekte*, die am Ende des Baum sitzen und Dateien entsprechen.
- Das Wurzelelement heisst *root*, oder auch *base* oder *suffix*

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest

  dc Domainname Component

  o Organization

  ou Oganizatinol Unit

  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest

  dc Domainname Component

  o Organization

  ou Oganizatinol Unit

  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

eine umfassende Auflistung gibt es hier:
http://ldap.akbkhome.com

Die Datenstruktur eines Directory Information Trees

- Die Verzeichnis-Einträge enthalten *Attribut-Werte-Paare*.
- Ein Verzeichnis-Eintrag kann einen oder mehrere Attribut-Werte-Paare enthalten.
- Einem Attribut kann kein, ein einzelner Wert oder auch ein Satz von Werten zugeordnet werden. Ein Verzeichnis-Eintrag darf also *nicht-atomare* Werte enthalten (vgl. RDBMS, 1NF) Einige Attribute:

  dn Distinguished Name, legt *absoluten Pfad* im Baum fest
  dc Domainname Component
  o Organization
  ou Oganizatinol Unit
  cn Common Name

  eine umfassende Auflistung gibt es hier:
  `http://ldap.akbkhome.com`

Beispiel für einen Directory Information Tree

Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Beispiel für einen Directory Information Tree

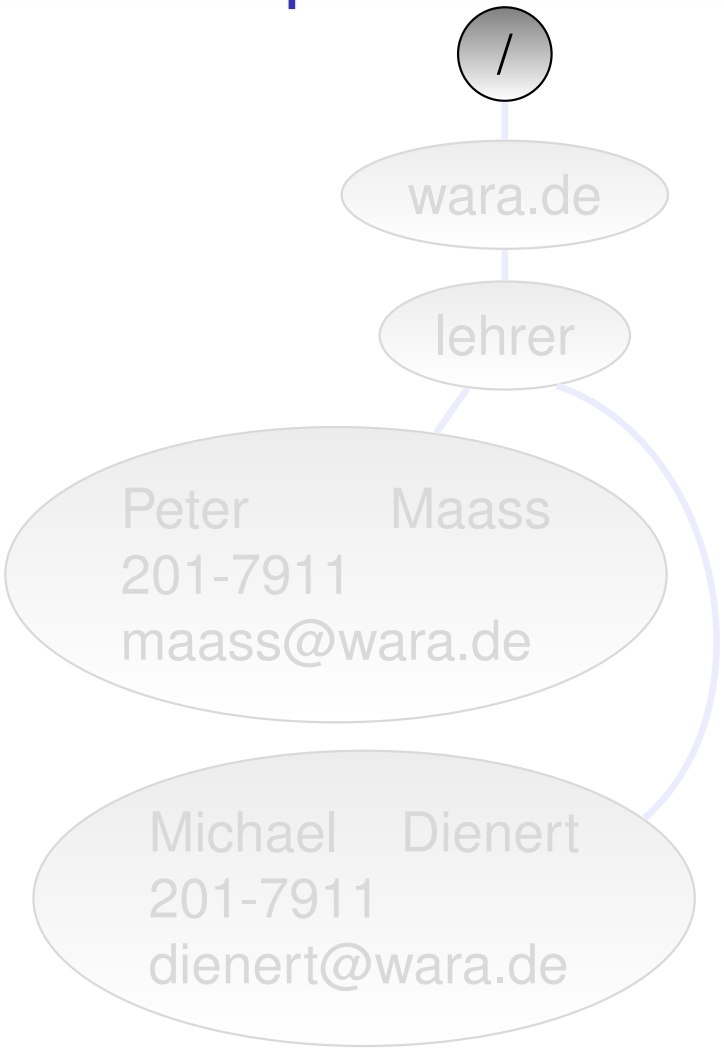


Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Beispiel für einen Directory Information Tree

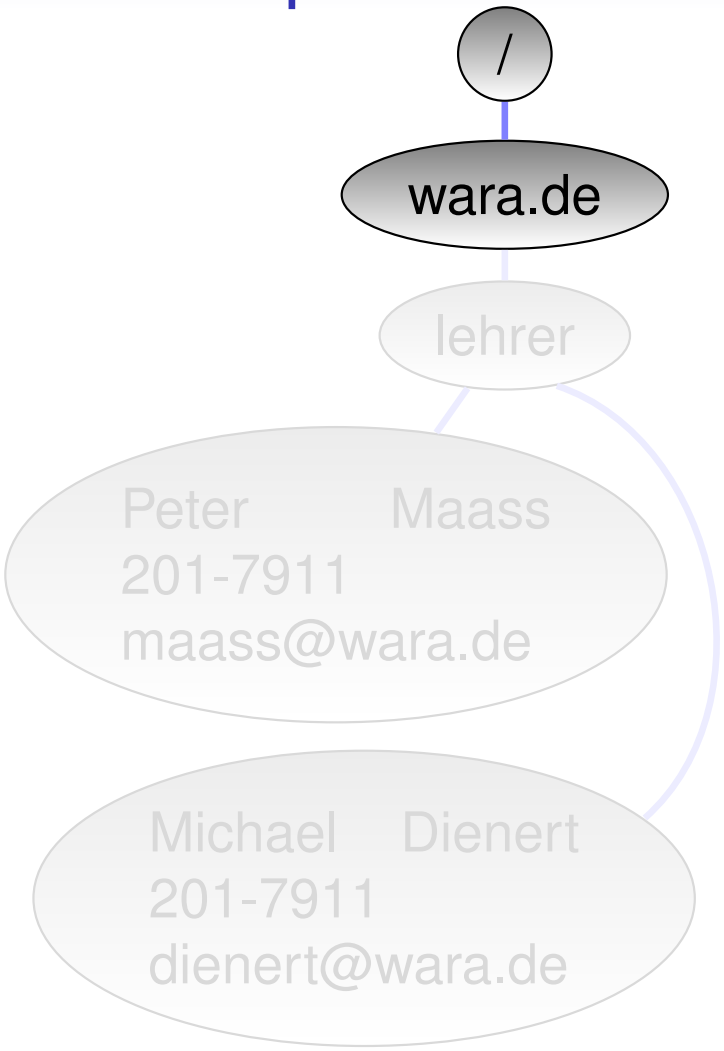


Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Beispiel für einen Directory Information Tree

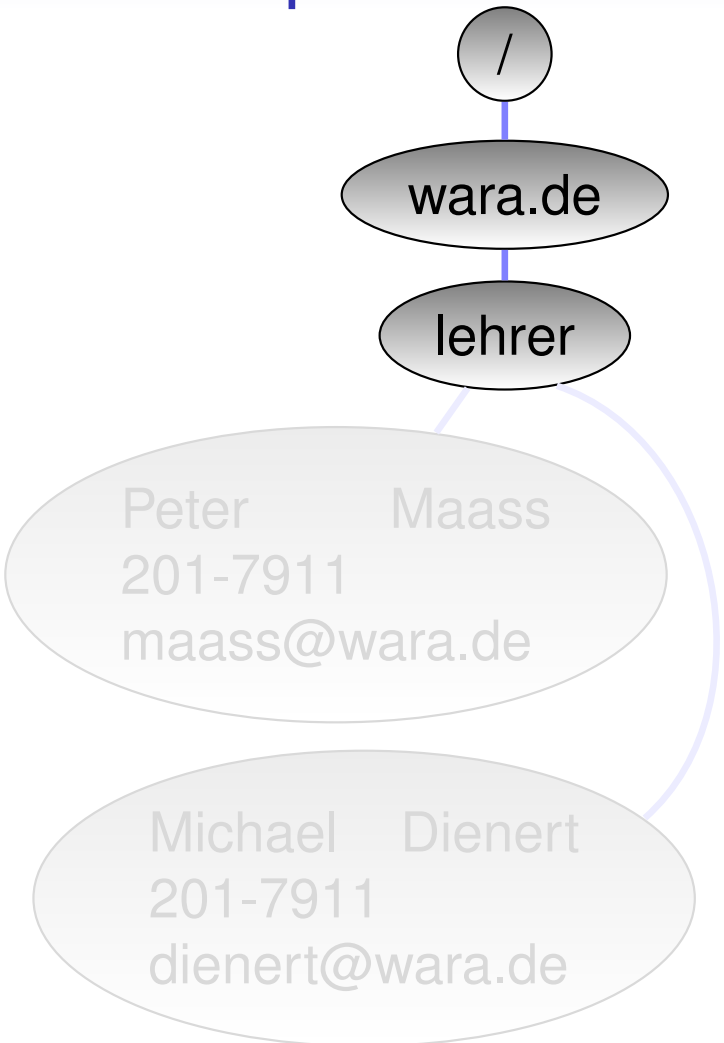


Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Beispiel für einen Directory Information Tree

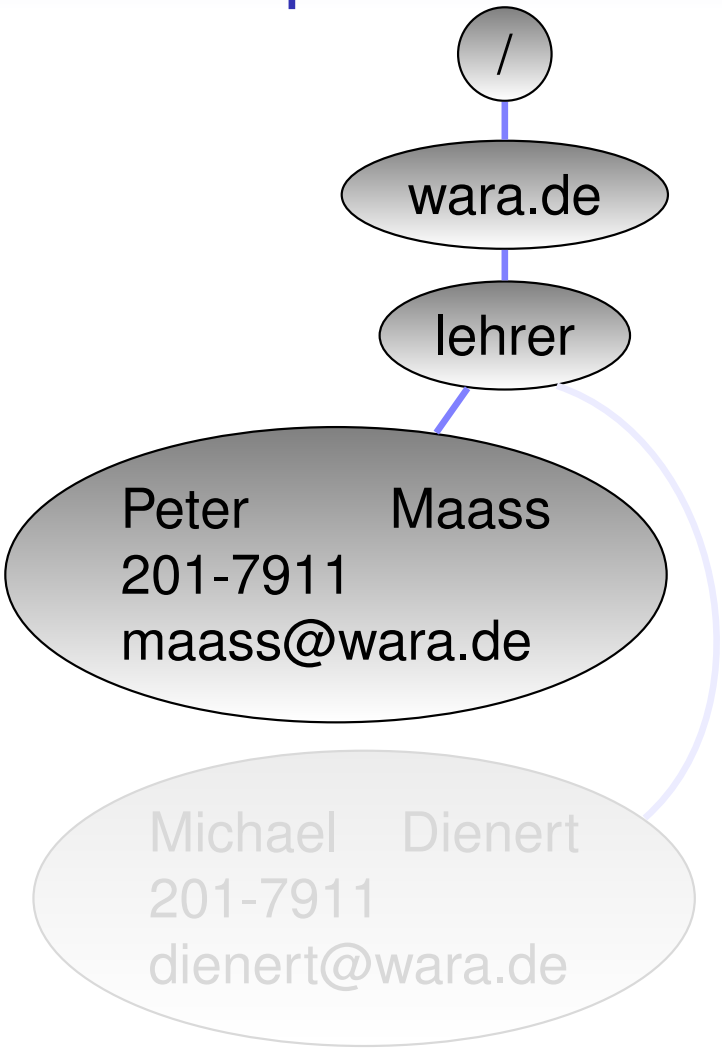


Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Beispiel für einen Directory Information Tree

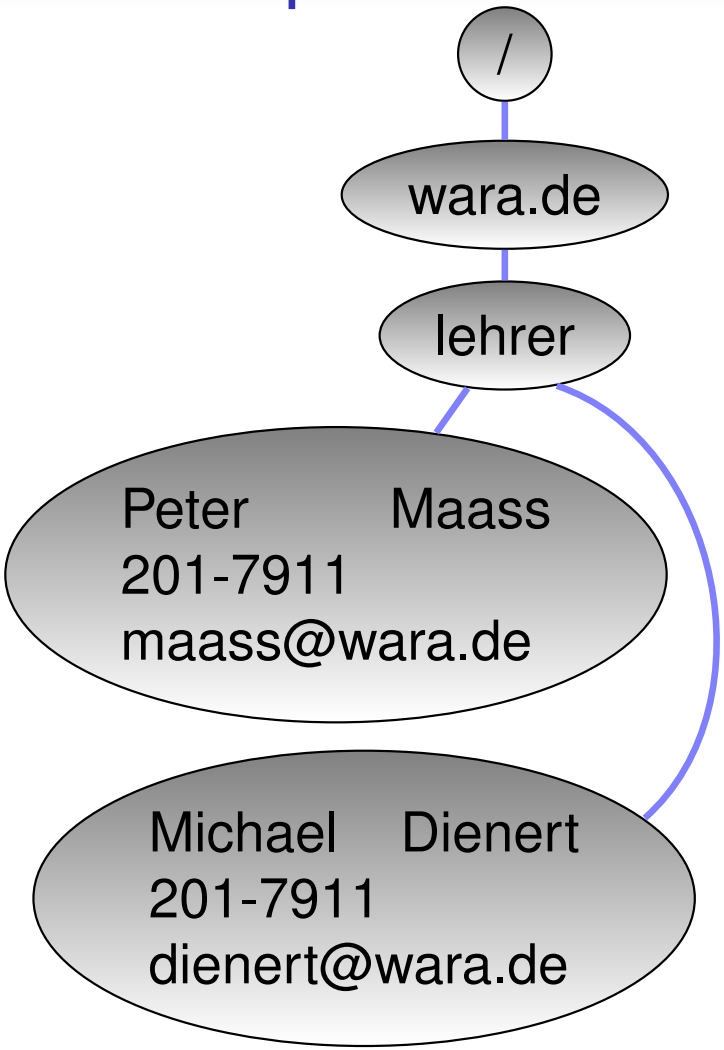


Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Beispiel für einen Directory Information Tree

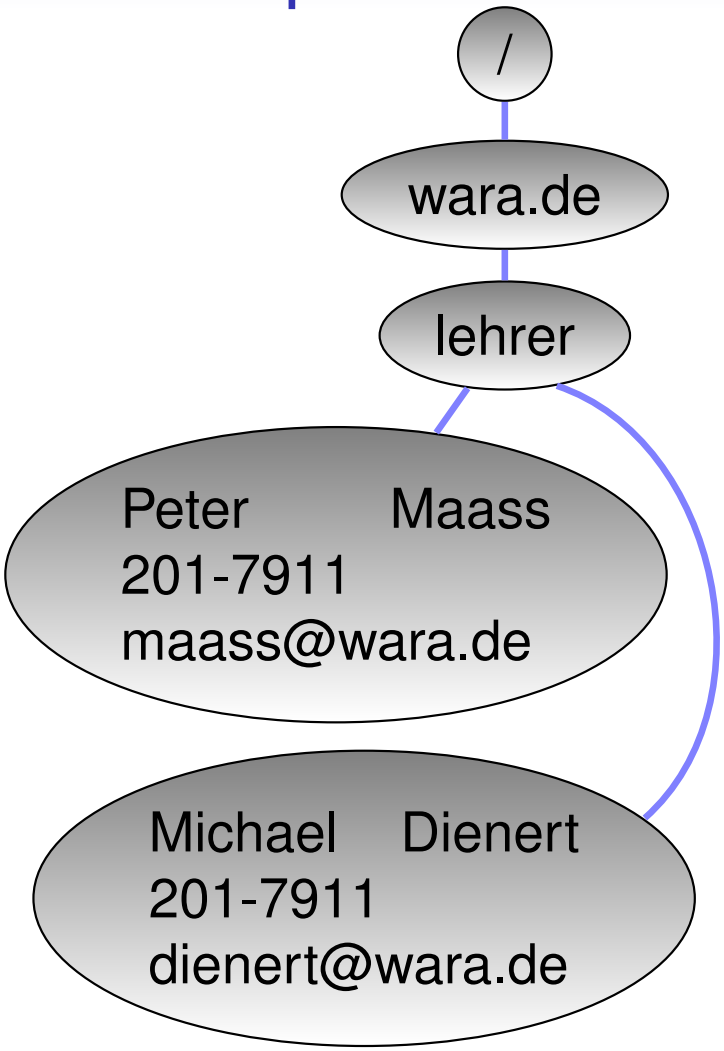


Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Beispiel für einen Directory Information Tree

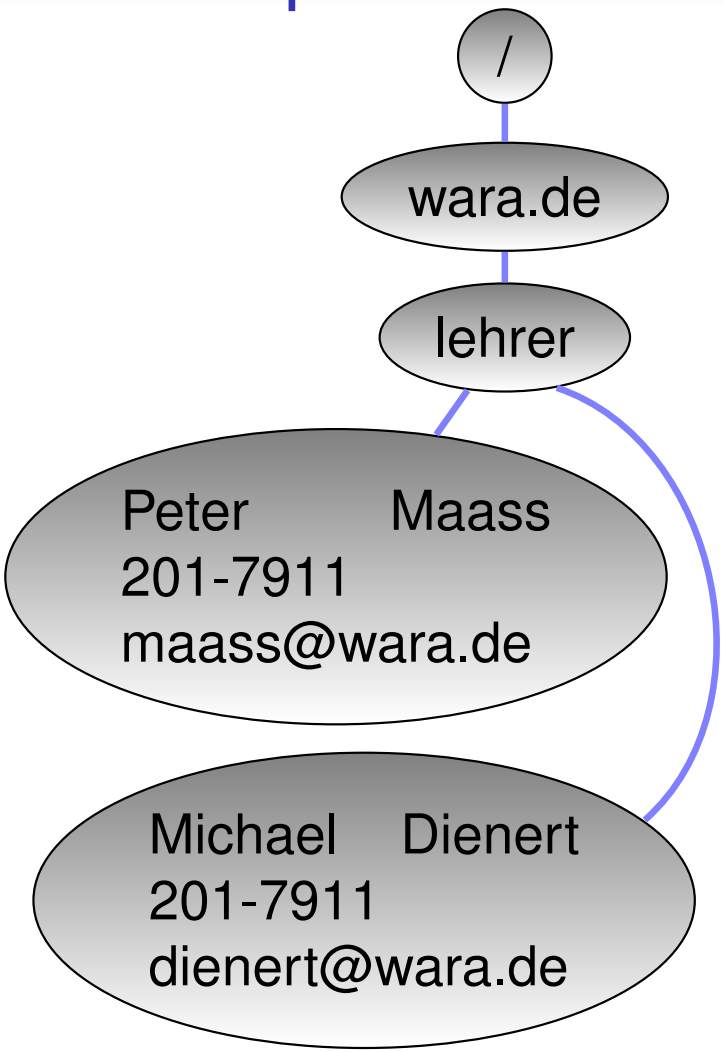


Abbildung 1 : Ein einfacher X.500-Verzeichnisbaum

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Ojektklassen

- Am Beispiel fällt die häufige Verwendung des Attributs **objectclass** auf.
- Das ObjectClass-Attribut bestimmt, welche Attribut-Werte-Paare in einem Verzeichnis-Eintrag (Knoten) existieren müssen (vorgeschriebene Attribute) oder dürfen (optionale Attribute).
- Pro Verzeichnis-Eintrag gibt es mindestens ein ObjectClass-Attribut. Es dürfen aber auch mehrere ObjectClass-Attribute vorhanden sein und damit der Verzeichnis-Eintrag zu mehreren Objektklassen gehören $\Rightarrow$ *Mehrfachvererbung*.

Ojektklassen

- Am Beispiel fällt die häufige Verwendung des Attributs **objectclass** auf.
- Das ObjectClass-Attribut bestimmt, welche Attribut-Werte-Paare in einem Verzeichnis-Eintrag (Knoten) existieren müssen (vorgeschriebene Attribute) oder dürfen (optionale Attribute).
- Pro Verzeichnis-Eintrag gibt es mindestens ein ObjectClass-Attribut. Es dürfen aber auch mehrere ObjectClass-Attribute vorhanden sein und damit der Verzeichnis-Eintrag zu mehreren Objektklassen gehören $\Rightarrow$ *Mehrfachvererbung*.

Ojektklassen

- Am Beispiel fällt die häufige Verwendung des Attributs **objectclass** auf.
- Das ObjectClass-Attribut bestimmt, welche Attribut-Werte-Paare in einem Verzeichnis-Eintrag (Knoten) existieren müssen (vorgeschriebene Attribute) oder dürfen (optionale Attribute).
- Pro Verzeichnis-Eintrag gibt es mindestens ein ObjectClass-Attribut. Es dürfen aber auch mehrere ObjectClass-Attribute vorhanden sein und damit der Verzeichnis-Eintrag zu mehreren Objektklassen gehören $\Rightarrow$ *Mehrfachvererbung*.

Ojektklassen

- Am Beispiel fällt die häufige Verwendung des Attributs **objectclass** auf.
- Das ObjectClass-Attribut bestimmt, welche Attribut-Werte-Paare in einem Verzeichnis-Eintrag (Knoten) existieren müssen (vorgeschriebene Attribute) oder dürfen (optionale Attribute).
- Pro Verzeichnis-Eintrag gibt es mindestens ein ObjectClass-Attribut. Es dürfen aber auch mehrere ObjectClass-Attribute vorhanden sein und damit der Verzeichnis-Eintrag zu mehreren Objektklassen gehören
  $\Rightarrow$ *Mehrfachvererbung*.

Ojektklassen

- Am Beispiel fällt die häufige Verwendung des Attributs **objectclass** auf.
- Das ObjectClass-Attribut bestimmt, welche Attribut-Werte-Paare in einem Verzeichnis-Eintrag (Knoten) existieren müssen (vorgeschriebene Attribute) oder dürfen (optionale Attribute).
- Pro Verzeichnis-Eintrag gibt es mindestens ein ObjectClass-Attribut. Es dürfen aber auch mehrere ObjectClass-Attribute vorhanden sein und damit der Verzeichnis-Eintrag zu mehreren Objektklassen gehören $\Rightarrow$ *Mehrfachvererbung*.

Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:

Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:

Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:

Objektklassen

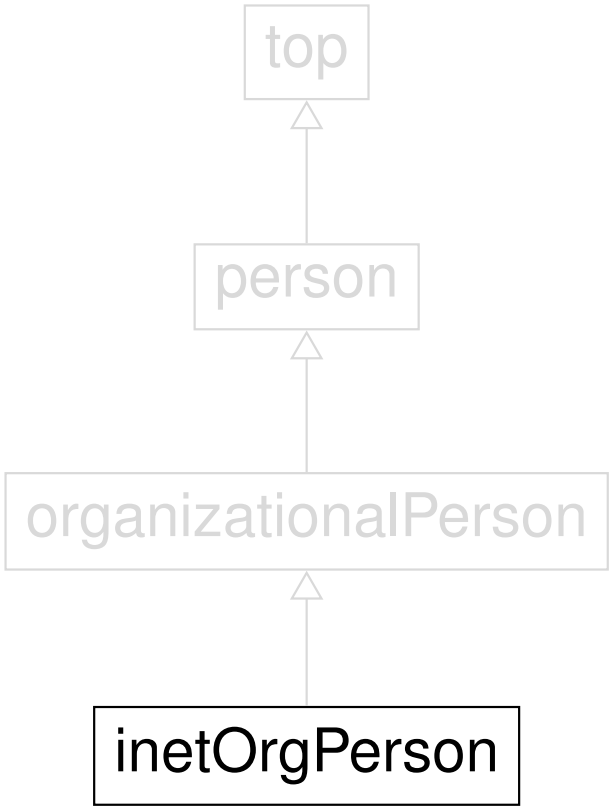
- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:

Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:

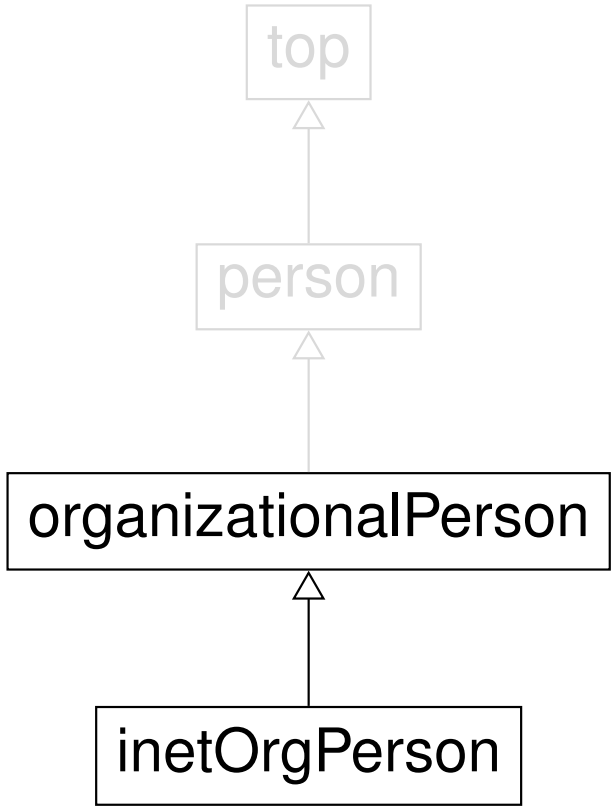
Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:



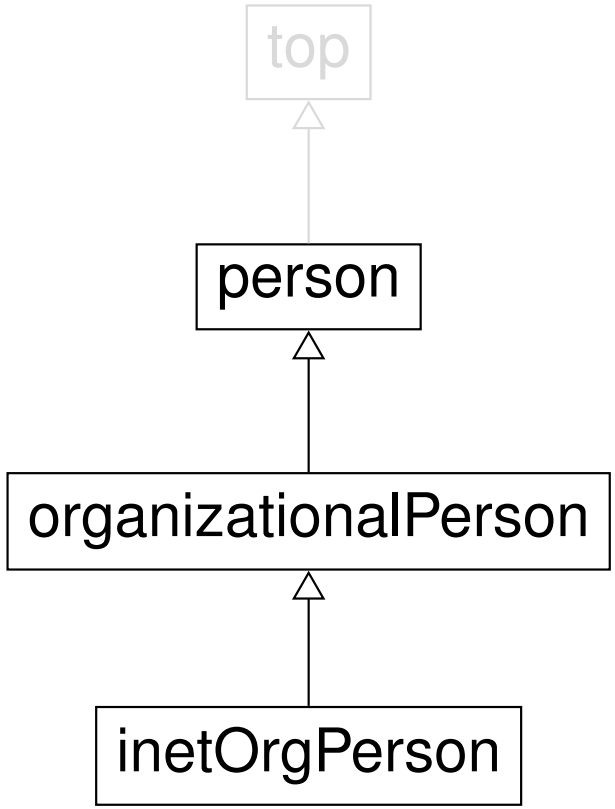
Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:



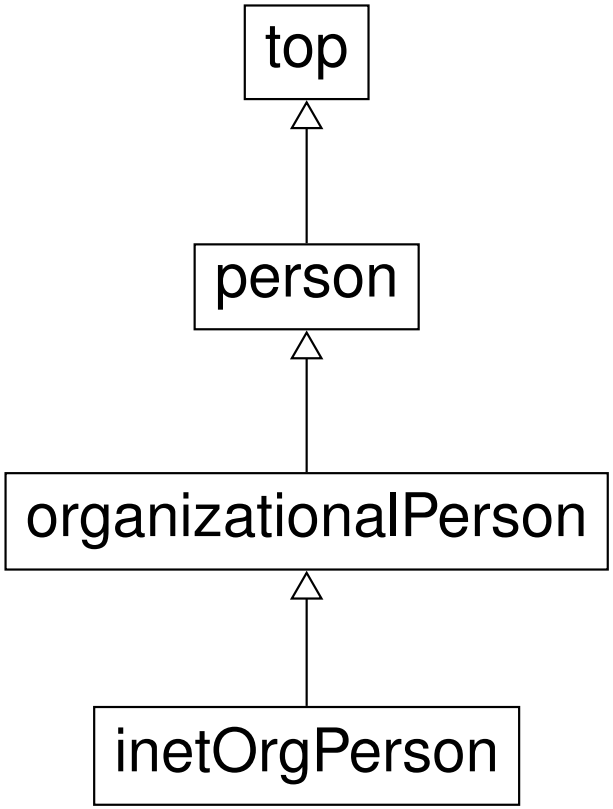
Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:



Objektklassen

- Eine Objektklasse ist Teil einer *Vererbungshierarchie*.
- Eine Objektklasse erbt alle Eigenschaften ihrer Eltern.
- Der Stammvater aller Objektklassen ist die abstrakte Objektklasse **top**. *top* entspricht somit der Klasse `Object` in Java. Hier ein Beispiel:



Beispiele für Objektklassen und Attribut-Definitionen

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
        DESC 'Abstraction of an account with POSIX attributes'
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description ) )
```

```
attributetype ( 2.5.4.5 NAME 'serialNumber'
        DESC 'RFC2256: serial number of the entity'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{64} )
```

LDAP DIT: Zusammenfassung

- Der *Directory Information Tree* besteht aus *Verzeichnis Einträgen* (Directory Entries).
- Jeder Eintrag hat genau ein *Elternelement* und kein, ein oder viele *Kindelemente*. Im LDAP-Jargon heissen die Eltern *Superior*, abgekürzt **SUP**.
- Jeder Eintrag besteht ausschliesslich aus *Attribut-Werte*-Paaren.
- Jedes Attribut hat einen Namen und ist Mitglied einer oder mehrerer *Objektklassen*.
- Jeder Eintrag wiederum ist *Instanz* einer oder mehrerer *Objektklassen*.
- Welche Objektklassen das sind, wird durch Attribute vom Typ **objectclass** festgelegt.

LDAP DIT: Zusammenfassung

- Der *Directory Information Tree* besteht aus *Verzeichnis Einträgen* (Directory Entries).
- Jeder Eintrag hat genau ein *Elternelement* und kein, ein oder viele *Kindelemente*. Im LDAP-Jargon heissen die Eltern *Superior*, abgekürzt **SUP**.
- Jeder Eintrag besteht ausschliesslich aus *Attribut-Werte*-Paaren.
- Jedes Attribut hat einen Namen und ist Mitglied einer oder mehrerer *Objektklassen*.
- Jeder Eintrag wiederum ist *Instanz* einer oder mehrerer *Objektklassen*.
- Welche Objektklassen das sind, wird durch Attribute vom Typ **objectclass** festgelegt.

LDAP DIT: Zusammenfassung

- Der *Directory Information Tree* besteht aus *Verzeichnis Einträgen* (Directory Entries).
- Jeder Eintrag hat genau ein *Elternelement* und kein, ein oder viele *Kindelemente*. Im LDAP-Jargon heissen die Eltern *Superior*, abgekürzt **SUP**.
- Jeder Eintrag besteht ausschliesslich aus *Attribut-Werte*-Paaren.
- Jedes Attribut hat einen Namen und ist Mitglied einer oder mehrerer *Objektklassen*.
- Jeder Eintrag wiederum ist *Instanz* einer oder mehrerer *Objektklassen*.
- Welche Objektklassen das sind, wird durch Attribute vom Typ **objectclass** festgelegt.

LDAP DIT: Zusammenfassung

- Der *Directory Information Tree* besteht aus *Verzeichnis Einträgen* (Directory Entries).
- Jeder Eintrag hat genau ein *Elternelement* und kein, ein oder viele *Kindelemente*. Im LDAP-Jargon heissen die Eltern *Superior*, abgekürzt **SUP**.
- Jeder Eintrag besteht ausschliesslich aus *Attribut-Werte*-Paaren.
- Jedes Attribut hat einen Namen und ist Mitglied einer oder mehrerer *Objektklassen*.
- Jeder Eintrag wiederum ist *Instanz* einer oder mehrerer *Objektklassen*.
- Welche Objektklassen das sind, wird durch Attribute vom Typ **objectclass** festgelegt.

LDAP DIT: Zusammenfassung

- Der *Directory Information Tree* besteht aus *Verzeichnis Einträgen* (Directory Entries).
- Jeder Eintrag hat genau ein *Elternelement* und kein, ein oder viele *Kindelemente*. Im LDAP-Jargon heissen die Eltern *Superior*, abgekürzt **SUP**.
- Jeder Eintrag besteht ausschliesslich aus *Attribut-Werte*-Paaren.
- Jedes Attribut hat einen Namen und ist Mitglied einer oder mehrerer *Objektklassen*.
- Jeder Eintrag wiederum ist *Instanz* einer oder mehrerer *Objektklassen*.
- Welche Objektklassen das sind, wird durch Attribute vom Typ **objectclass** festgelegt.

LDAP DIT: Zusammenfassung

- Der *Directory Information Tree* besteht aus *Verzeichnis Einträgen* (Directory Entries).
- Jeder Eintrag hat genau ein *Elternelement* und kein, ein oder viele *Kindelemente*. Im LDAP-Jargon heissen die Eltern *Superior*, abgekürzt **SUP**.
- Jeder Eintrag besteht ausschliesslich aus *Attribut-Werte*-Paaren.
- Jedes Attribut hat einen Namen und ist Mitglied einer oder mehrerer *Objektklassen*.
- Jeder Eintrag wiederum ist *Instanz* einer oder mehrerer *Objektklassen*.
- Welche Objektklassen das sind, wird durch Attribute vom Typ **objectclass** festgelegt.

LDAP DIT: Zusammenfassung

- Der *Directory Information Tree* besteht aus *Verzeichnis Einträgen* (Directory Entries).
- Jeder Eintrag hat genau ein *Elternelement* und kein, ein oder viele *Kindelemente*. Im LDAP-Jargon heissen die Eltern *Superior*, abgekürzt **SUP**.
- Jeder Eintrag besteht ausschliesslich aus *Attribut-Werte*-Paaren.
- Jedes Attribut hat einen Namen und ist Mitglied einer oder mehrerer *Objektklassen*.
- Jeder Eintrag wiederum ist *Instanz* einer oder mehrerer *Objektklassen*.
- Welche Objektklassen das sind, wird durch Attribute vom Typ **objectclass** festgelegt.

LDAP-Schemas

- Ein LDAP-Schema ist eine Art *Package*, das Definitionen von Objektklassen und Attributen enthält.
- dabei gilt folgende Regel: jedes Attribut und jede Objektklasse muss in einem Schema definiert sein
- möchte man Attribute und Objektklassen verwenden, muss das entsprechende Schema dem LDAP-Server bekannt gemacht werden.
-
  - altes Verfahren: Datei `slapd.conf`
  - aktuelles Verfahren: *on-line configuration* (OLC); Konfiguration wird selbst wieder über einen DIT vorgenommen

LDAP-Schemas

- Ein LDAP-Schema ist eine Art *Package*, das Definitionen von Objektklassen und Attributen enthält.
- dabei gilt folgende Regel: jedes Attribut und jede Objektklasse muss in einem Schema definiert sein
- möchte man Attribute und Objektklassen verwenden, muss das entsprechende Schema dem LDAP-Server bekannt gemacht werden.
-
  - altes Verfahren: Datei `slapd.conf`
  - aktuelles Verfahren: *on-line configuration* (OLC); Konfiguration wird selbst wieder über einen DIT vorgenommen

LDAP-Schemas

- Ein LDAP-Schema ist eine Art *Package*, das Definitionen von Objektklassen und Attributen enthält.
- dabei gilt folgende Regel: jedes Attribut und jede Objektklasse muss in einem Schema definiert sein
- möchte man Attribute und Objektklassen verwenden, muss das entsprechende Schema dem LDAP-Server bekannt gemacht werden.
-
  - altes Verfahren: Datei `slapd.conf`
  - aktuelles Verfahren: *on-line configuration* (OLC); Konfiguration wird selbst wieder über einen DIT vorgenommen

LDAP-Schemas

- Ein LDAP-Schema ist eine Art *Package*, das Definitionen von Objektklassen und Attributen enthält.
- dabei gilt folgende Regel: jedes Attribut und jede Objektklasse muss in einem Schema definiert sein
- möchte man Attribute und Objektklassen verwenden, muss das entsprechende Schema dem LDAP-Server bekannt gemacht werden.
-
  - altes Verfahren: Datei `slapd.conf`
  - aktuelles Verfahren: *on-line configuration* (OLC); Konfiguration wird selbst wieder über einen DIT vorgenommen

LDAP-Schemas

- Ein LDAP-Schema ist eine Art *Package*, das Definitionen von Objektklassen und Attributen enthält.
- dabei gilt folgende Regel: jedes Attribut und jede Objektklasse muss in einem Schema definiert sein
- möchte man Attribute und Objektklassen verwenden, muss das entsprechende Schema dem LDAP-Server bekannt gemacht werden.
-
  - altes Verfahren: Datei `slapd.conf`
  - aktuelles Verfahren: *on-line configuration* (OLC); Konfiguration wird selbst wieder über einen DIT vorgenommen

LDAP-Schemas

- Ein LDAP-Schema ist eine Art *Package*, das Definitionen von Objektklassen und Attributen enthält.
- dabei gilt folgende Regel: jedes Attribut und jede Objektklasse muss in einem Schema definiert sein
- möchte man Attribute und Objektklassen verwenden, muss das entsprechende Schema dem LDAP-Server bekannt gemacht werden.
-
  - altes Verfahren: Datei `slapd.conf`
  - aktuelles Verfahren: *on-line configuration* (OLC); Konfiguration wird selbst wieder über einen DIT vorgenommen

LDAP-Schemas

- Ein LDAP-Schema ist eine Art *Package*, das Definitionen von Objektklassen und Attributen enthält.
- dabei gilt folgende Regel: jedes Attribut und jede Objektklasse muss in einem Schema definiert sein
- möchte man Attribute und Objektklassen verwenden, muss das entsprechende Schema dem LDAP-Server bekannt gemacht werden.
-
  - altes Verfahren: Datei `slapd.conf`
  - aktuelles Verfahren: *on-line configuration* (OLC); Konfiguration wird selbst wieder über einen DIT vorgenommen

Erzeugen des Baums und Einfügen von Daten

Um den LDAP-Baum aufzubauen, gibt es mehrere Möglichkeiten:

1. Einfügen von Einträgen direkt in die Backend-Datenbank: `slapadd, slapcat, ...`
2. Einfügen von Einträgen mit dem LDAP-Protokoll selbst: `ldapsearch, ldapadd, ldappaswd, ldapdelete, ...`
3. Mit grafischen Werkzeugen, sog. LDAP-Browsern: `Apache Direct Studio, jexplore, luma, ...`
4. Durch Einspielen eines *LDAP Data Interchange Files*

Erzeugen des Baums und Einfügen von Daten

Um den LDAP-Baum aufzubauen, gibt es mehrere Möglichkeiten:

1. Einfügen von Einträgen direkt in die Backend-Datenbank: `slapadd, slapcat, ...`
2. Einfügen von Einträgen mit dem LDAP-Protokoll selbst: `ldapsearch, ldapadd, ldappaswd, ldapdelete, ...`
3. Mit grafischen Werkzeugen, sog. LDAP-Browsern: `Apache Direct Studio, jexplore, luma, ...`
4. Durch Einspielen eines *LDAP Data Interchange Files*

Erzeugen des Baums und Einfügen von Daten

Um den LDAP-Baum aufzubauen, gibt es mehrere Möglichkeiten:

1. Einfügen von Einträgen direkt in die Backend-Datenbank: `slapadd, slapcat, ...`
2. Einfügen von Einträgen mit dem LDAP-Protokoll selbst: `ldapsearch, ldapadd, ldappaswd, ldapdelete, ...`
3. Mit grafischen Werkzeugen, sog. LDAP-Browsern: `Apache Direct Studio, jexplore, luma, ...`
4. Durch Einspielen eines *LDAP Data Interchange Files*

Erzeugen des Baums und Einfügen von Daten

Um den LDAP-Baum aufzubauen, gibt es mehrere Möglichkeiten:

1. Einfügen von Einträgen direkt in die Backend-Datenbank: `slapadd, slapcat, ...`
2. Einfügen von Einträgen mit dem LDAP-Protokoll selbst: `ldapsearch, ldapadd, ldappaswd, ldapdelete, ...`
3. Mit grafischen Werkzeugen, sog. LDAP-Browsern: `Apache Direct Studio, jexplore, luma, ...`
4. Durch Einspielen eines *LDAP Data Interchange Files*

Erzeugen des Baums und Einfügen von Daten

Um den LDAP-Baum aufzubauen, gibt es mehrere Möglichkeiten:

1. Einfügen von Einträgen direkt in die Backend-Datenbank: `slapadd, slapcat, ...`
2. Einfügen von Einträgen mit dem LDAP-Protokoll selbst: `ldapsearch, ldapadd, ldappaswd, ldapdelete, ...`
3. Mit grafischen Werkzeugen, sog. LDAP-Browsern: `Apache Direct Studio, jexplore, luma, ...`
4. Durch Einspielen eines *LDAP Data Interchange Files*

Erzeugen des Baums und Einfügen von Daten

Um den LDAP-Baum aufzubauen, gibt es mehrere Möglichkeiten:

1. Einfügen von Einträgen direkt in die Backend-Datenbank: `slapadd, slapcat, ...`
2. Einfügen von Einträgen mit dem LDAP-Protokoll selbst: `ldapsearch, ldapadd, ldappaswd, ldapdelete, ...`
3. Mit grafischen Werkzeugen, sog. LDAP-Browsern: `Apache Direct Studio, jexplore, luma, ...`
4. Durch Einspielen eines *LDAP Data Interchange Files*

Ein LDIF-Beispiel

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

- Die Angabe der Version ist nicht zwingend, wird aber empfohlen.
- Root/Base/Suffix: `wara.de`. Die Verwendung von DNS-Namen fuer X.500-Verzeichnisse ist in der RFC2377 beschrieben.
- Attribut **dn**: Die Zeilen, die mit **dn:** beginnen bestimmen, an welche Stelle im Baum die Einträge hingehören. Natürlich muss der Baum von der Wurzel her aufgebaut werden, d.h. die Reihenfolge in der LDIF-Datei ist wichtig.

Ein LDIF-Beispiel

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

- Die Angabe der Version ist nicht zwingend, wird aber empfohlen.
- Root/Base/Suffix: `wara.de`. Die Verwendung von DNS-Namen fuer X.500-Verzeichnisse ist in der RFC2377 beschrieben.
- Attribut **dn**: Die Zeilen, die mit **dn:** beginnen bestimmen, an welche Stelle im Baum die Einträge hingehören. Natürlich muss der Baum von der Wurzel her aufgebaut werden, d.h. die Reihenfolge in der LDIF-Datei ist wichtig.

Ein LDIF-Beispiel

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

- Die Angabe der Version ist nicht zwingend, wird aber empfohlen.
- Root/Base/Suffix: `wara.de`. Die Verwendung von DNS-Namen fuer X.500-Verzeichnisse ist in der RFC2377 beschrieben.
- Attribut **dn**: Die Zeilen, die mit **dn:** beginnen bestimmen, an welche Stelle im Baum die Einträge hingehören. Natürlich muss der Baum von der Wurzel her aufgebaut werden, d.h. die Reihenfolge in der LDIF-Datei ist wichtig.

Ein LDIF-Beispiel

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

- Die Angabe der Version ist nicht zwingend, wird aber empfohlen.
- Root/Base/Suffix: `wara.de`. Die Verwendung von DNS-Namen fuer X.500-Verzeichnisse ist in der RFC2377 beschrieben.
- Attribut **dn**: Die Zeilen, die mit **dn:** beginnen bestimmen, an welche Stelle im Baum die Einträge hingehören. Natürlich muss der Baum von der Wurzel her aufgebaut werden, d.h. die Reihenfolge in der LDIF-Datei ist wichtig.

Ein LDIF-Beispiel

```
version: 1

dn: dc=wara,dc=de
dc: wara
description: die beste it-schule in freiburg
objectClass: dcObject
objectClass: organization
o: Walther-Rathenau-Gewerbeschule

dn: ou=lehrer, dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

- Die Angabe der Version ist nicht zwingend, wird aber empfohlen.
- Root/Base/Suffix: `wara.de`. Die Verwendung von DNS-Namen fuer X.500-Verzeichnisse ist in der RFC2377 beschrieben.
- Attribut **dn**: Die Zeilen, die mit **dn:** beginnen bestimmen, an welche Stelle im Baum die Einträge hingehören. Natürlich muss der Baum von der Wurzel her aufgebaut werden, d.h. die Reihenfolge in der LDIF-Datei ist wichtig.

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Distinguished Names: absolut und relativ

Sehr wichtig sind die Attribute **dn** und **rdn**

- Der **DN** entspricht einem *absoluten Pfad* in einem Dateisystem
- Der **RDN** entspricht einem *Dateinamen* in einem Dateisystem, kann sich allerdings aus mehreren Attributen zusammensetzen (s.u.)
- wie auch in einem Dateisystem, wird der absolute Pfad, also der DN *nicht* in der Backend-DB selbst gespeichert!
- Im obigen Lehrerbeispiel sind die rdns identisch mit den cn der Lehrer.
- Was passiert bei Namensgleicheit? $\Rightarrow$ *Multivalued RDN*

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Multivalued RDN

Im Kollegium gibt es zwei Personen *Frank Müller* in der Abteilung *Allgemeinbildung*.

```
dn: cn=Frank Müller+lb=englisch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: englisch
```

```
dn: cn=Frank Müller+lb=deutsch,
 ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Frank Müller
ou: Allgemeinbildung
lb: deutsch
```

- Damit wir eindeutige RDNs erhalten, wird noch das Attribut **lb** (Lehrbefähigung) hinzugenommen
- Die RDNs setzen sich nun aus cn und lb zusammen:
  - cn=Frank Müller+lb=englisch
  - cn=Frank Müller+lb=deutsch
- natürlich müsste man dazu das Attribut **lb** definieren und die Objektklassen erweitern.
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Ein bisschen Praxis zur Entspannung

1. erste Schritte mit: **OpenLDAP**
2. warum OpenLDAP?
   - frei, quelloffen
   - Standard unter Linux, BSD und OSX-Server
   - Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
   - unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
   - Anbindung an viele Dienste möglich (out-of-the-box)
3. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

Grundkonfiguration: nächste Folie!

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

Grundkonfiguration des slapd

Die folgenden Schritte können nach Eingabe von

```
dpkg-reconfigure slapd
```

beliebig oft wiederholt werden.

1. Konfiguration *nicht* überspringen
2. DNS-Domainnamen wählen. Beispiel: wara.de
3. Firmen/Schulnamen wählen. Beispiel: wara
4. Administrator-Passwort wählen. Beispiel:

   ```
   ljwml! (letztes jahr war mehr lametta ! (bei Fam. Hoppensted) )
   ```

5. **HDB** als Datenbankbackend wählen.
6. LDAPv2 **nicht** erlauben

LDIF-Datei erstellen und einspielen

Mit folgender Kommandozeile kann man sich die komplette *slapd* Grundkonfiguration einschliesslich aller geladener Schemata ansehen:

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config" |less
```

- Das Kommando setzt root-Rechte voraus (uid=0), sonst bekommt man nicht viel zu sehen. Der Benutzer mit der uid=0 (root) hat alle Rechte.
- verantwortlich dafür ist das Attribut **olcAccess**

LDIF-Datei erstellen und einspielen

Mit folgender Kommandozeile kann man sich die komplette *slapd* Grundkonfiguration einschliesslich aller geladener Schemata ansehen:

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config" |less
```

- Das Kommando setzt root-Rechte voraus (uid=0), sonst bekommt man nicht viel zu sehen. Der Benutzer mit der uid=0 (root) hat alle Rechte.
- verantwortlich dafür ist das Attribut **olcAccess**

LDIF-Datei erstellen und einspielen

Mit folgender Kommandozeile kann man sich die komplette *slapd* Grundkonfiguration einschliesslich aller geladener Schemata ansehen:

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config" |less
```

- Das Kommando setzt root-Rechte voraus (uid=0), sonst bekommt man nicht viel zu sehen. Der Benutzer mit der uid=0 (root) hat alle Rechte.
- verantwortlich dafür ist das Attribut **olcAccess**

LDIF-Datei erstellen und einspielen

Mit folgender Kommandozeile kann man sich die komplette *slapd* Grundkonfiguration einschliesslich aller geladener Schemata ansehen:

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config" |less
```

- Das Kommando setzt root-Rechte voraus (uid=0), sonst bekommt man nicht viel zu sehen. Der Benutzer mit der uid=0 (root) hat alle Rechte.
- verantwortlich dafür ist das Attribut **olcAccess**

LDIF-Datei erstellen und einspielen

Mit folgender Kommandozeile kann man sich die komplette *slapd* Grundkonfiguration einschliesslich aller geladener Schemata ansehen:

```
ldapsearch -Y EXTERNAL -H ldapi:/// -b "cn=config" |less
```

- Das Kommando setzt root-Rechte voraus (uid=0), sonst bekommt man nicht viel zu sehen. Der Benutzer mit der uid=0 (root) hat alle Rechte.
- verantwortlich dafür ist das Attribut **olcAccess**

LDIF-Datei erstellen und einspielen

- Entwerfen Sie einen X.500-Baum, z.B. wie bei Abb. 1 gezeigt.
- Schreiben Sie die zugehörige LDIF-Datei mit einem Editor (z.B. gedit, emacs, vi, nano).Achtung: ein Teil des Baumes wurde schon bei der Grundkonfiguration erzeugt!
- Einspielen der LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f test.ldif -W -x
```

  Passwort ist das Passwort des Administrators des wara.de-Baums, das wir bei der Grundkonfiguration eingegeben haben.

- Abfragen des Verzeichnisses:

```
ldapsearch  -b "dc=wara,dc=de" -x
```

LDIF-Datei erstellen und einspielen

- Entwerfen Sie einen X.500-Baum, z.B. wie bei Abb. 1 gezeigt.
- Schreiben Sie die zugehörige LDIF-Datei mit einem Editor (z.B. gedit, emacs, vi, nano).Achtung: ein Teil des Baumes wurde schon bei der Grundkonfiguration erzeugt!
- Einspielen der LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f test.ldif -W -x
```

  Passwort ist das Passwort des Administrators des wara.de-Baums, das wir bei der Grundkonfiguration eingegeben haben.

- Abfragen des Verzeichnisses:

```
ldapsearch  -b "dc=wara,dc=de" -x
```

LDIF-Datei erstellen und einspielen

- Entwerfen Sie einen X.500-Baum, z.B. wie bei Abb. 1 gezeigt.
- Schreiben Sie die zugehörige LDIF-Datei mit einem Editor (z.B. gedit, emacs, vi, nano).Achtung: ein Teil des Baumes wurde schon bei der Grundkonfiguration erzeugt!
- Einspielen der LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f test.ldif -W -x
```

  Passwort ist das Passwort des Administrators des wara.de-Baums, das wir bei der Grundkonfiguration eingegeben haben.

- Abfragen des Verzeichnisses:

```
ldapsearch  -b "dc=wara,dc=de" -x
```

LDIF-Datei erstellen und einspielen

- Entwerfen Sie einen X.500-Baum, z.B. wie bei Abb. 1 gezeigt.
- Schreiben Sie die zugehörige LDIF-Datei mit einem Editor (z.B. gedit, emacs, vi, nano).Achtung: ein Teil des Baumes wurde schon bei der Grundkonfiguration erzeugt!
- Einspielen der LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f test.ldif -W -x
```

  Passwort ist das Passwort des Administrators des wara.de-Baums, das wir bei der Grundkonfiguration eingegeben haben.

- Abfragen des Verzeichnisses:

```
ldapsearch  -b "dc=wara,dc=de" -x
```

LDIF-Datei erstellen und einspielen

- Entwerfen Sie einen X.500-Baum, z.B. wie bei Abb. 1 gezeigt.
- Schreiben Sie die zugehörige LDIF-Datei mit einem Editor (z.B. gedit, emacs, vi, nano).Achtung: ein Teil des Baumes wurde schon bei der Grundkonfiguration erzeugt!
- Einspielen der LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f test.ldif -W -x
```

Passwort ist das Passwort des Administrators des wara.de-Baums, das wir bei der Grundkonfiguration eingegeben haben.

- Abfragen des Verzeichnisses:

```
ldapsearch  -b "dc=wara,dc=de" -x
```

LDIF-Datei erstellen und einspielen

- Entwerfen Sie einen X.500-Baum, z.B. wie bei Abb. 1 gezeigt.
- Schreiben Sie die zugehörige LDIF-Datei mit einem Editor (z.B. gedit, emacs, vi, nano).Achtung: ein Teil des Baumes wurde schon bei der Grundkonfiguration erzeugt!
- Einspielen der LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f test.ldif -W -x
```

  Passwort ist das Passwort des Administrators des wara.de-Baums, das wir bei der Grundkonfiguration eingegeben haben.

- Abfragen des Verzeichnisses:

```
ldapsearch  -b "dc=wara,dc=de" -x
```

LDIF-Datei erstellen und einspielen

- Entwerfen Sie einen X.500-Baum, z.B. wie bei Abb. 1 gezeigt.
- Schreiben Sie die zugehörige LDIF-Datei mit einem Editor (z.B. gedit, emacs, vi, nano).Achtung: ein Teil des Baumes wurde schon bei der Grundkonfiguration erzeugt!
- Einspielen der LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f test.ldif -W -x
```

  Passwort ist das Passwort des Administrators des wara.de-Baums, das wir bei der Grundkonfiguration eingegeben haben.

- Abfragen des Verzeichnisses:

```
ldapsearch  -b "dc=wara,dc=de" -x
```

Ein paar Aufgaben

- Frage: was bewirken die Optionen -D, -f, -W, -x ? (man-Pages konsultieren)
- Fügen Sie noch einen weiteren Lehrpersoneneintrag hinzu. Hierzu müssen Sie eine weitere LDIF-Datei erstellen und mit *ldapadd* einspielen.

Ein paar Aufgaben

- Frage: was bewirken die Optionen -D, -f, -W, -x ? (man-Pages konsultieren)
- Fügen Sie noch einen weiteren Lehrpersoneneintrag hinzu. Hierzu müssen Sie eine weitere LDIF-Datei erstellen und mit *ldapadd* einspielen.

Ein paar Aufgaben

- Frage: was bewirken die Optionen -D, -f, -W, -x ? (man-Pages konsultieren)
- Fügen Sie noch einen weiteren Lehrpersoneneintrag hinzu. Hierzu müssen Sie eine weitere LDIF-Datei erstellen und mit *ldapadd* einspielen.

Lösungen

LDIF-Datei

```
version: 1

dn: ou=lehrer,dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Lösungen

LDIF-Datei

```
version: 1

dn: ou=lehrer,dc=wara,dc=de
ou: lehrer
description: der lehrkoerper der wara
objectClass: organizationalUnit

dn: cn=Peter Maass,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Peter Maass
sn: Maass
uid: mas
mail: maass@wara.de
telephonenumber: 201-7911
ou: it

dn: cn=Michael Dienert,ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: Michael Dienert
sn: Dienert
uid: dt
mail: dienert@wara.de
telephonenumber: 201-7911
ou: it
```

Lösungen

-D Distinguished Name (=abs. Pfad) desjenigen, der sich mit dem Verzeichnis verbindet (*bind*). Muss ggfs. authorisiert sein.

-f **F**ile.

-W Passwort über Eingabeaufforderung einlesen. Alternative: mit Option -w direkt mitgeben.

-x einfache Authorisierung anstelle von SASL verwenden

```
version: 1

dn: cn="Alfred E. Neumann",ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: "Alfred E. Neumann"
sn: Neumann
uid: aen
mail: neumann@wara.de
telephonenumber: 201-8888
```

Lösungen

- **-D** Distinguished Name (=abs. Pfad) desjenigen, der sich mit dem Verzeichnis verbindet (*bind*). Muss ggfs. authorisiert sein.
- **-f** **F**ile.
- **-W** Passwort über Eingabeaufforderung einlesen. Alternative: mit Option -w direkt mitgeben.
- **-x** einfache Authorisierung anstelle von SASL verwenden

```
version: 1

dn: cn="Alfred E. Neumann",ou=lehrer,dc=wara,dc=de
objectclass: inetOrgPerson
cn: "Alfred E. Neumann"
sn: Neumann
uid: aen
mail: neumann@wara.de
telephonenumber: 201-8888
```

Änderungen an einem Verzeichnis

Mit folgender LDIF-Datei können einige Einträge von Alfred geändert werden:

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

Einspielen lässt sich diese LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f aenderung.ldif -W -x
```

Änderungen an einem Verzeichnis

Mit folgender LDIF-Datei können einige Einträge von Alfred geändert werden:

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

Einspielen lässt sich diese LDIF-Datei mit:

```
ldapadd -D "cn=admin,dc=wara,dc=de" -f aenderung.ldif -W -x
```

Änderungen an einem Verzeichnis

Mit folgender LDIF-Datei können einige Einträge von Alfred geändert werden:

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

Einspielen lässt sich diese LDIF-Datei mit:

```
ldapadd -D "cn=admin,dc=wara,dc=de" -f aenderung.ldif -W -x
```

Änderungen an einem Verzeichnis

Mit folgender LDIF-Datei können einige Einträge von Alfred geändert werden:

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

Einspielen lässt sich diese LDIF-Datei mit:

```
ldapadd  -D "cn=admin,dc=wara,dc=de" -f aenderung.ldif -W -x
```

Änderungen an einem Verzeichnis

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,
 ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.

Änderungen an einem Verzeichnis

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,
 ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.

Änderungen an einem Verzeichnis

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,
 ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.

Änderungen an einem Verzeichnis

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,
 ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.

Änderungen an einem Verzeichnis

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,
 ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.

Änderungen an einem Verzeichnis

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,
 ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.

Änderungen an einem Verzeichnis

```
version: 1

##aenderungen bei alfred

dn: cn=Alfred E. Neumann,
 ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.

Schluss

Vielen Dank für’s Zuhören und Mitmachen!