Installing Docker

- OS specific instructions at `Install Docker Engine`.
- Windows: Make sure to install Windows Subsystem for Linux version 2 (WSL 2). Install Ubuntu from Microsoft's App store and follow Docker Desktop WSL 2 backend.

Why LDAP?

- **Standardized** protocol like HTTP, SMB and others.
- “Glue” connecting an organization's applications and services.
  - Backend data for authentication / SSO, PKI, network FS etc.
  - Meta data repository for users, organisational units, network nodes.

LDAP introduction

- Read the introductory slides at Chapter 28, *LDAP* as well.

Running a Docker container

```
docker run --detach ❶ \
  --name openldap ❷\
  -p 389:389 \ ❸
  --env LDAP_ORGANISATION="Betrayers heaven" \ ❹
  --env LDAP_TLS=false ❺\
  --env LDAP_DOMAIN="betrayer.com" ❻\
  --env LDAP_ADMIN_PASSWORD="secret" ❼\
  --env LDAP_CONFIG_PASSWORD="secret" ❽\
  --volume ~/OpenLdap/Data:/var/lib/ldap ❾\
  --volume ~/OpenLdap/Config:/etc/ldap/slapd.d ❾\
osixia/openldap:1.4.0 ❿
```

Using docker-compose

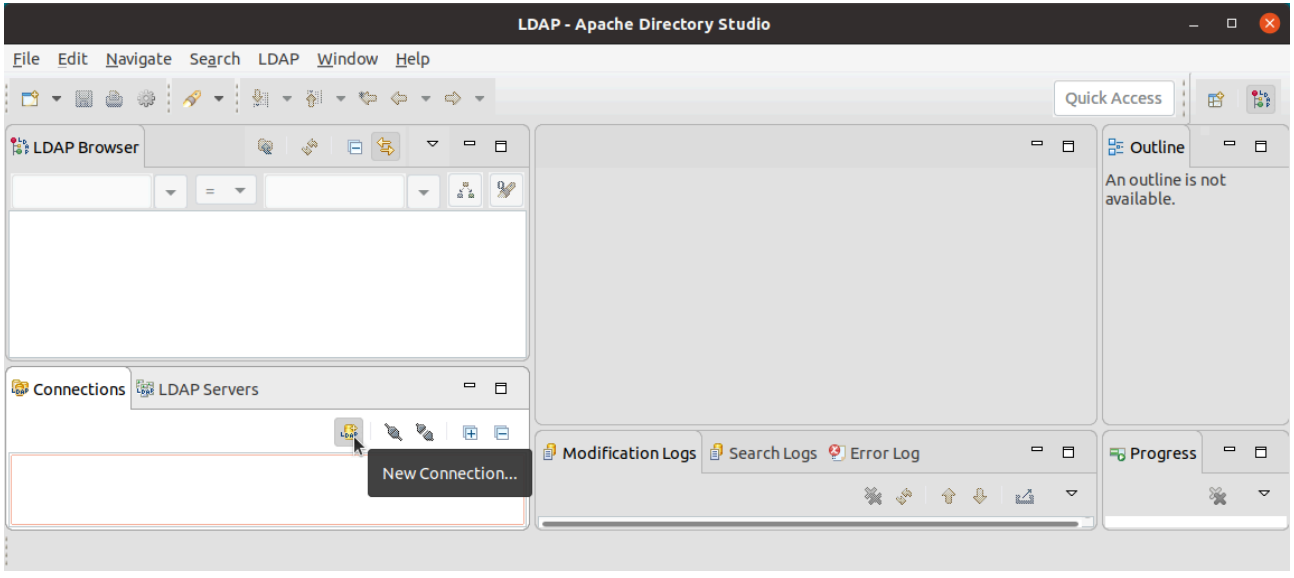
```
version: '3.7'

services:
    openldap:
        image: osixia/openldap:1.4.0
        container_name: openldap
        restart: always
        environment:
            LDAP_ORGANISATION: "Betrayers heaven"
            LDAP_TLS: "false"
            LDAP_DOMAIN: "betrayer.com"
            LDAP_ADMIN_PASSWORD: "secret"
            LDAP_CONFIG_PASSWORD: "secret"
        ports:
            - 389:389
        volumes:
            - ~/OpenLdap/Data:/var/lib/ldap
            - ~/OpenLdap/Config:/etc/ldap/slapd.d
```

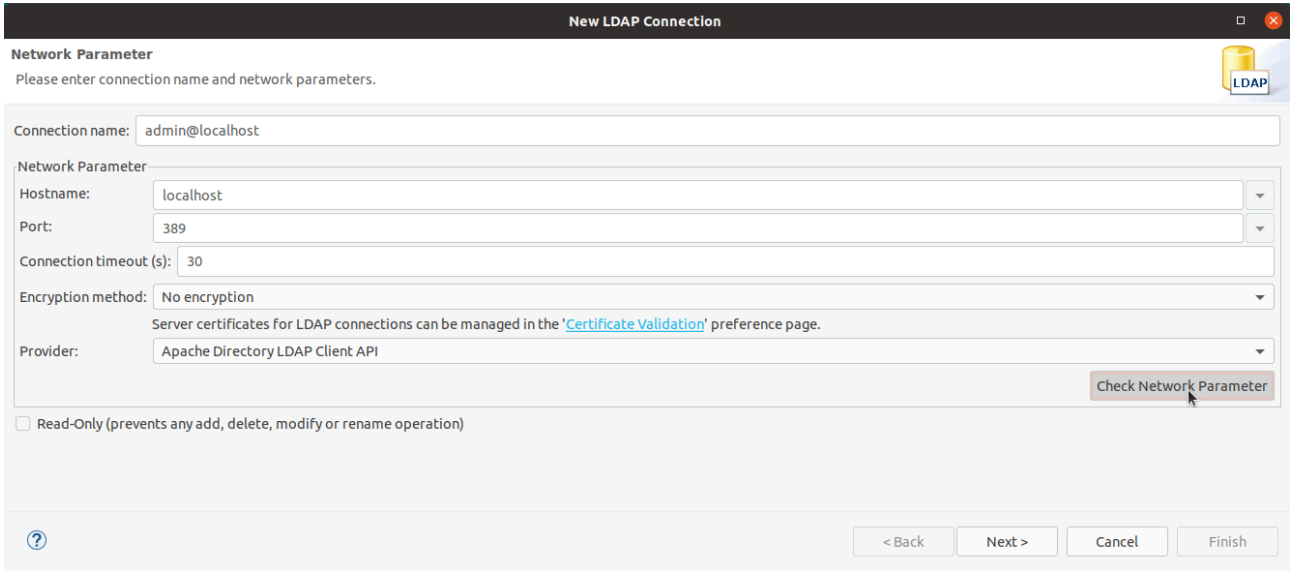
Installing Apache Directory Studio

- Download and install Apache Directory Studio.
- Configure access to your local docker container.

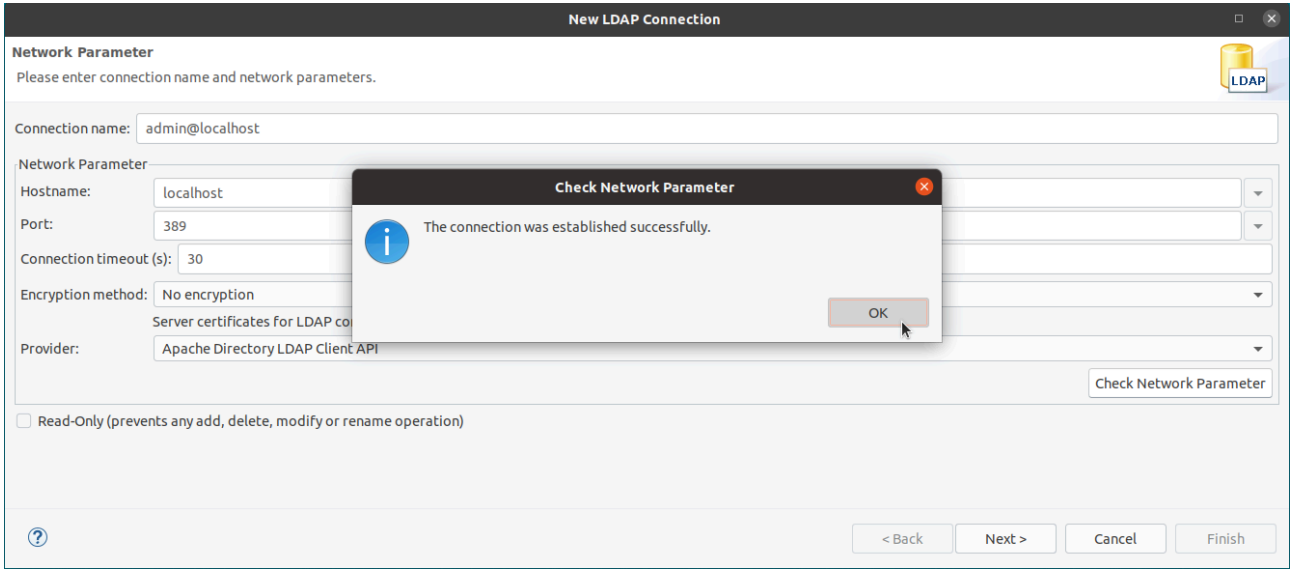
Administrator access to your DIT



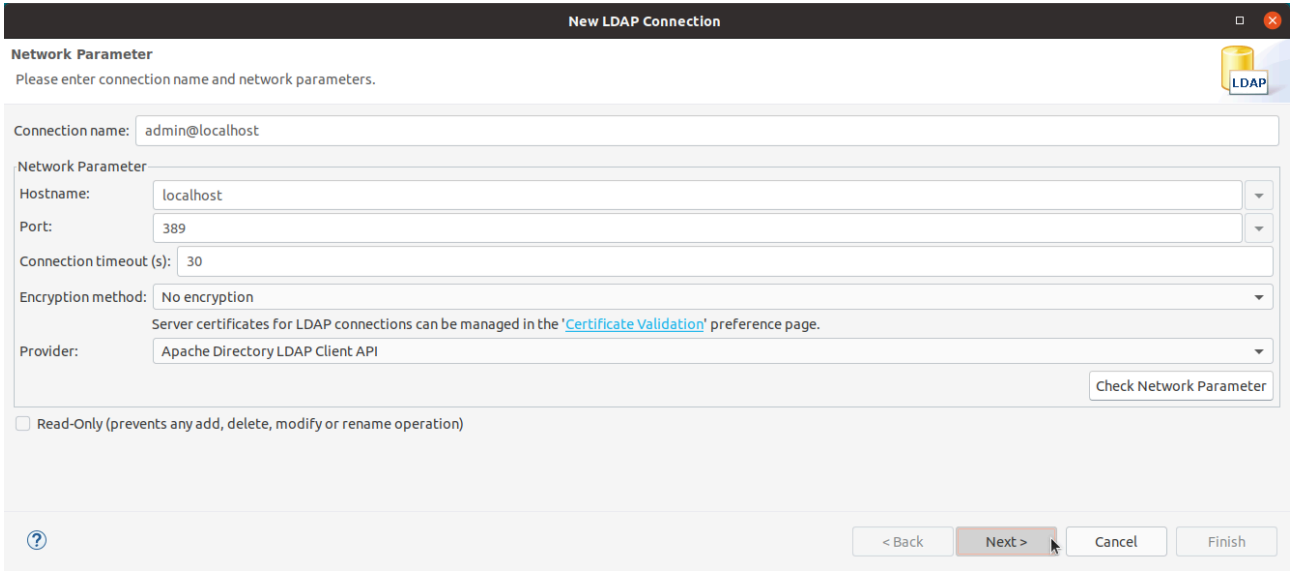
Administrator access to your DIT



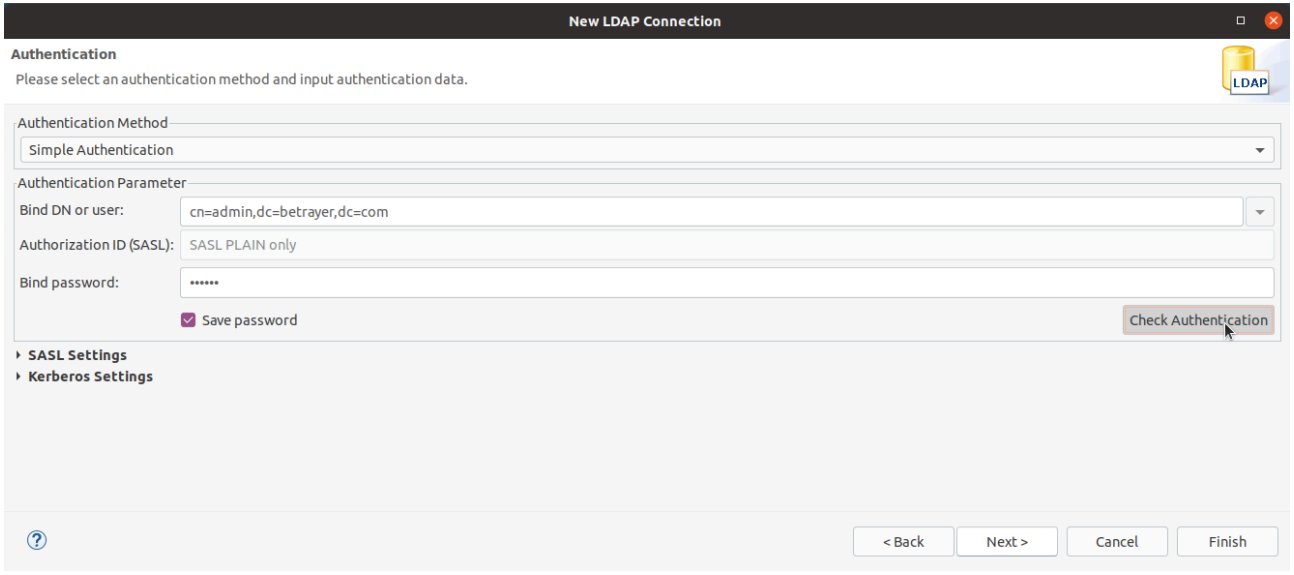
Administrator access to your DIT



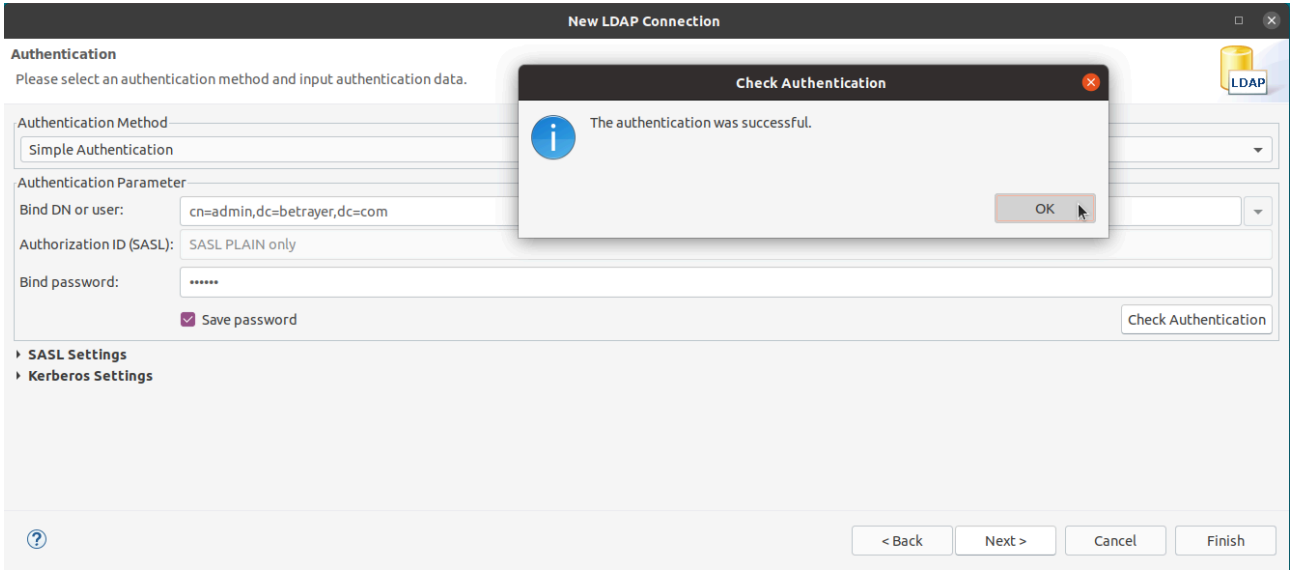
Administrator access to your DIT



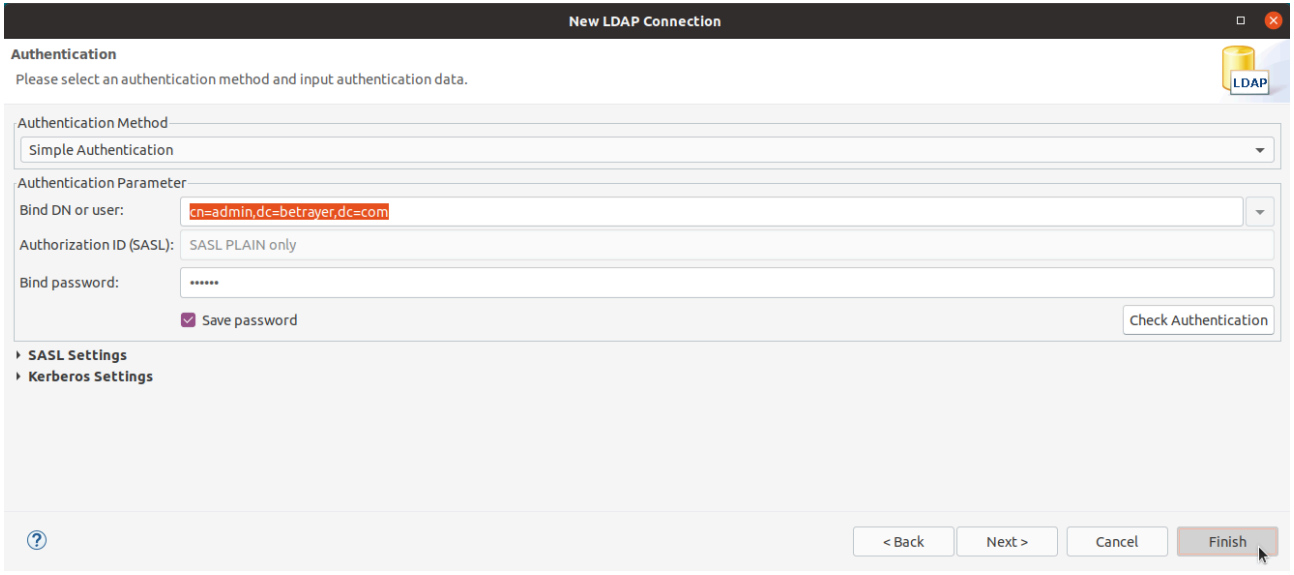
Administrator access to your DIT



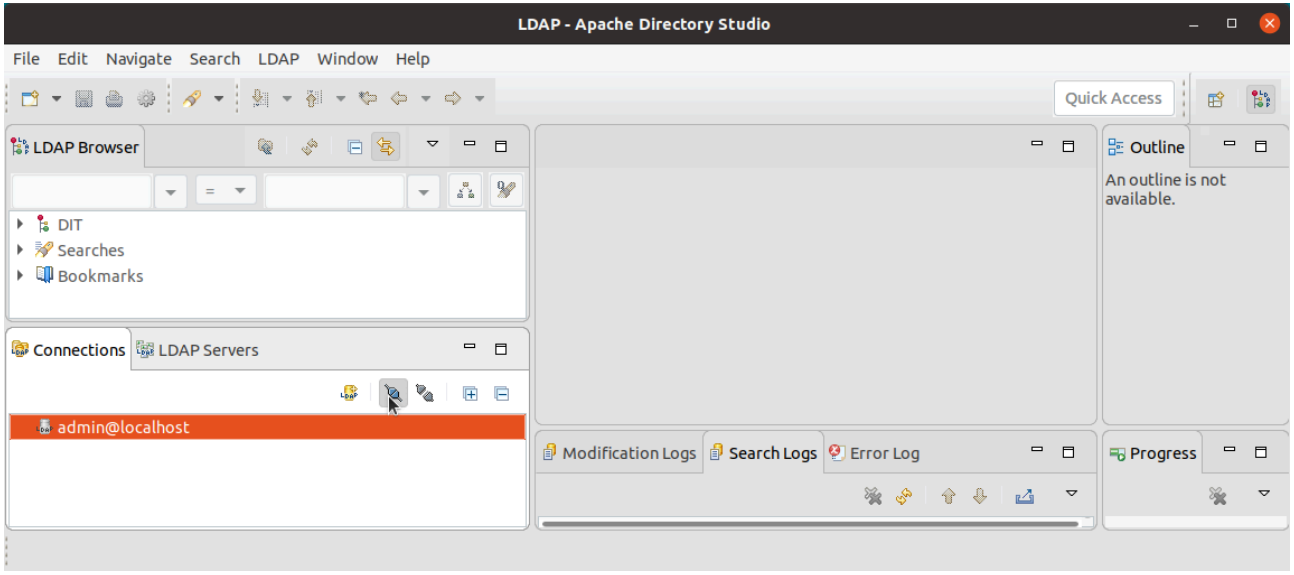
Administrator access to your DIT



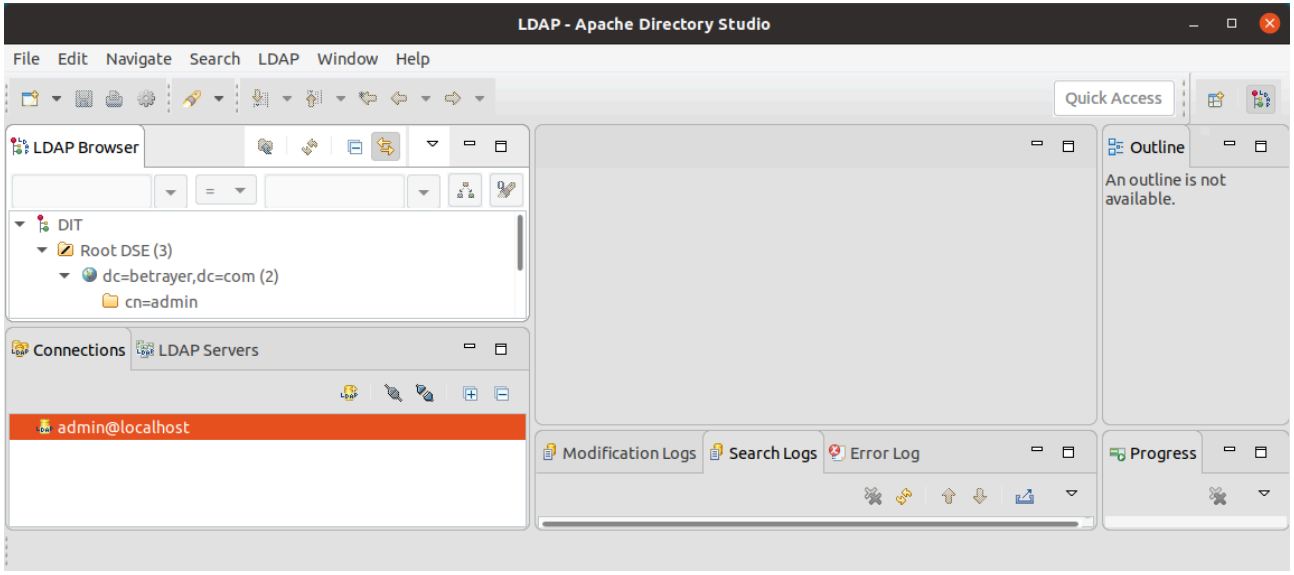
Administrator access to your DIT



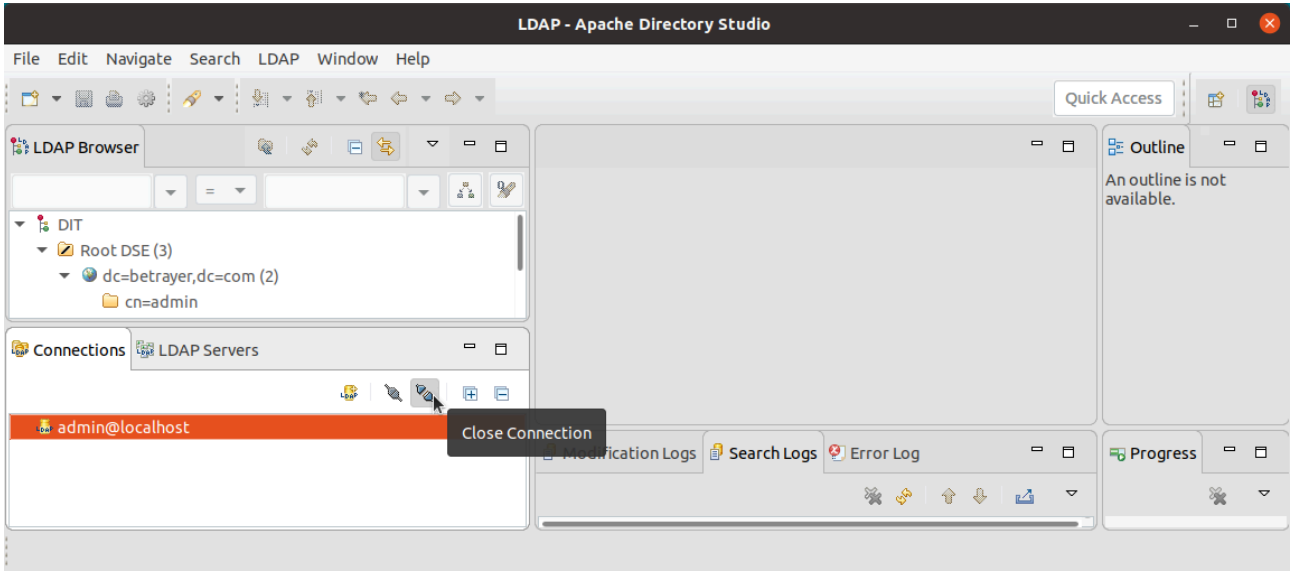
Administrator access to your DIT



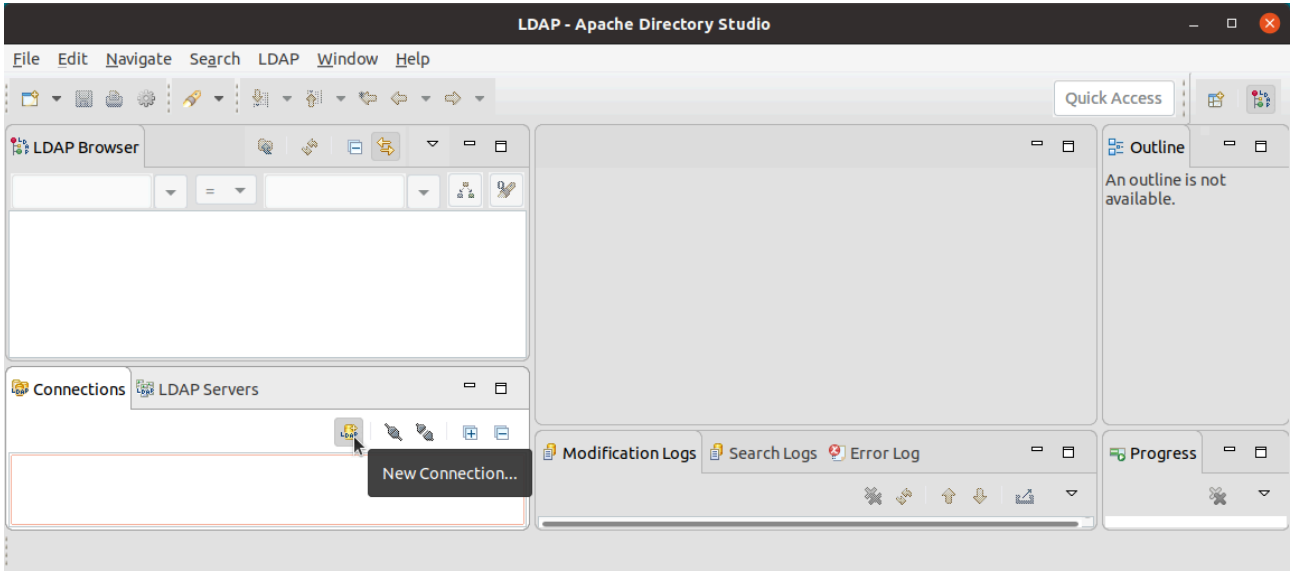
Administrator access to your DIT



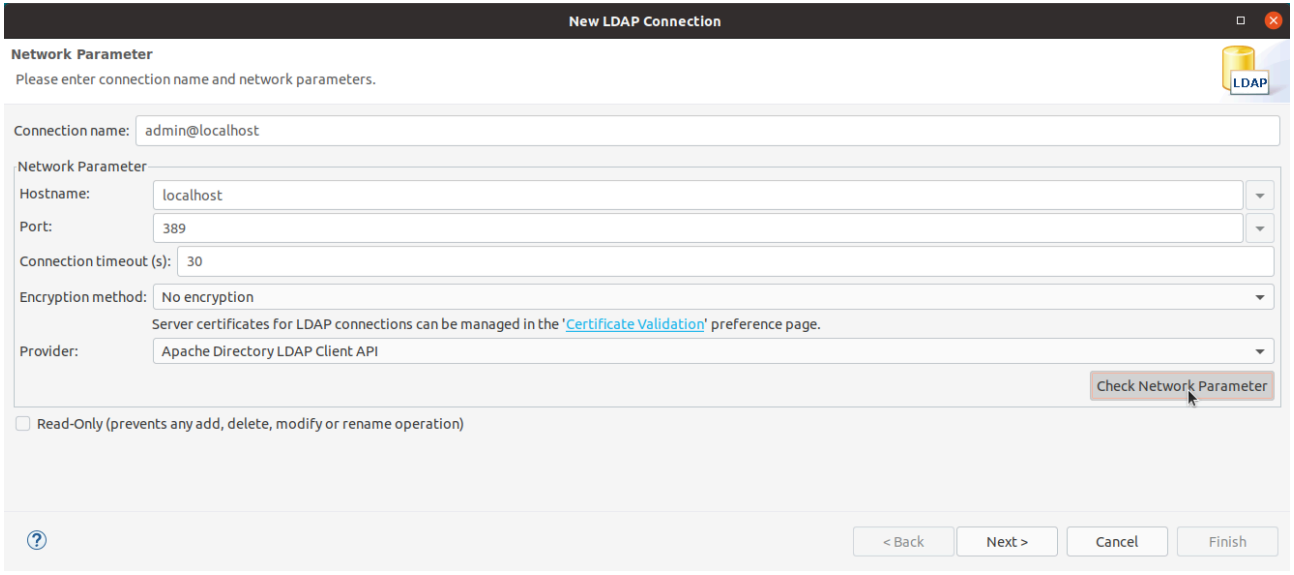
Administrator access to your DIT



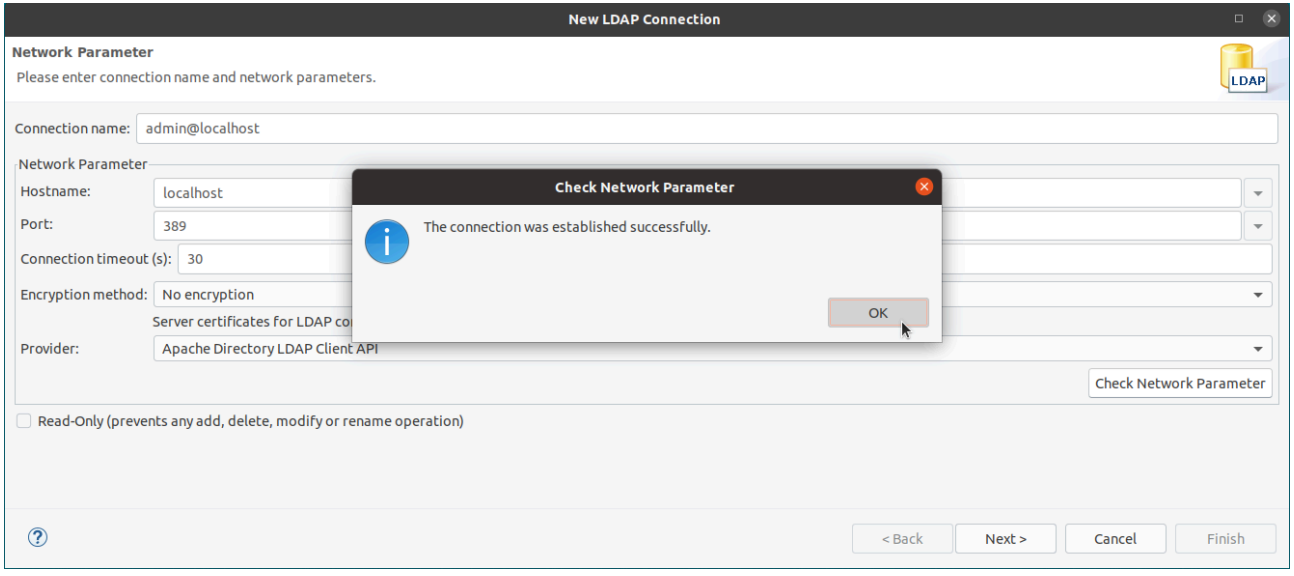
Administrator access to your server's data tree



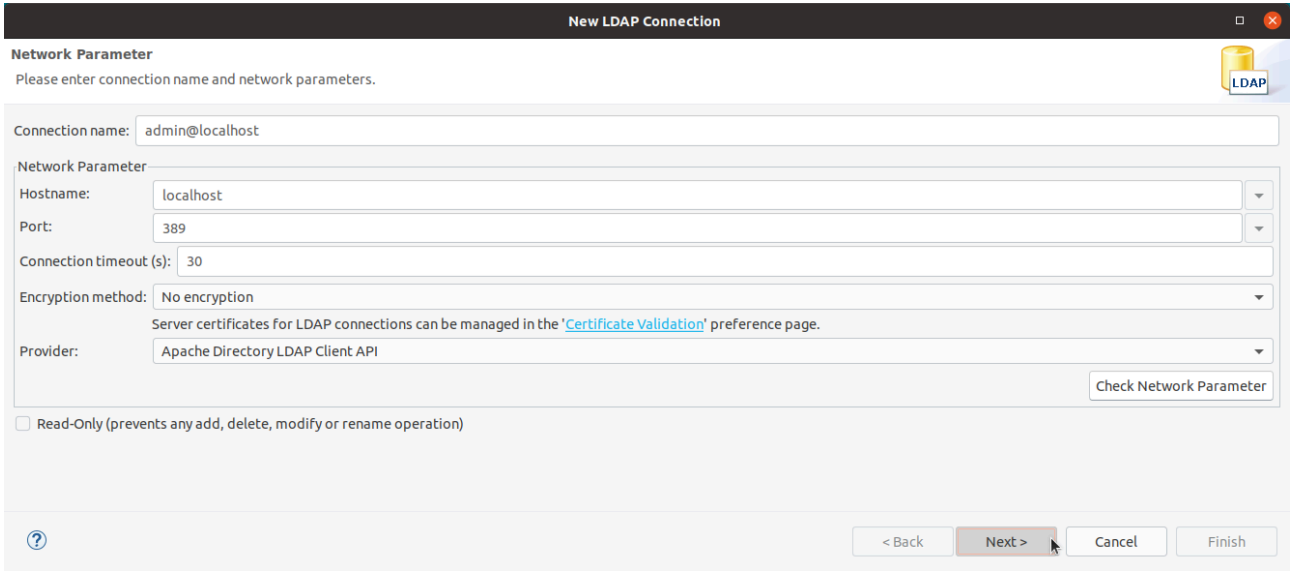
Administrator access to your server's data tree



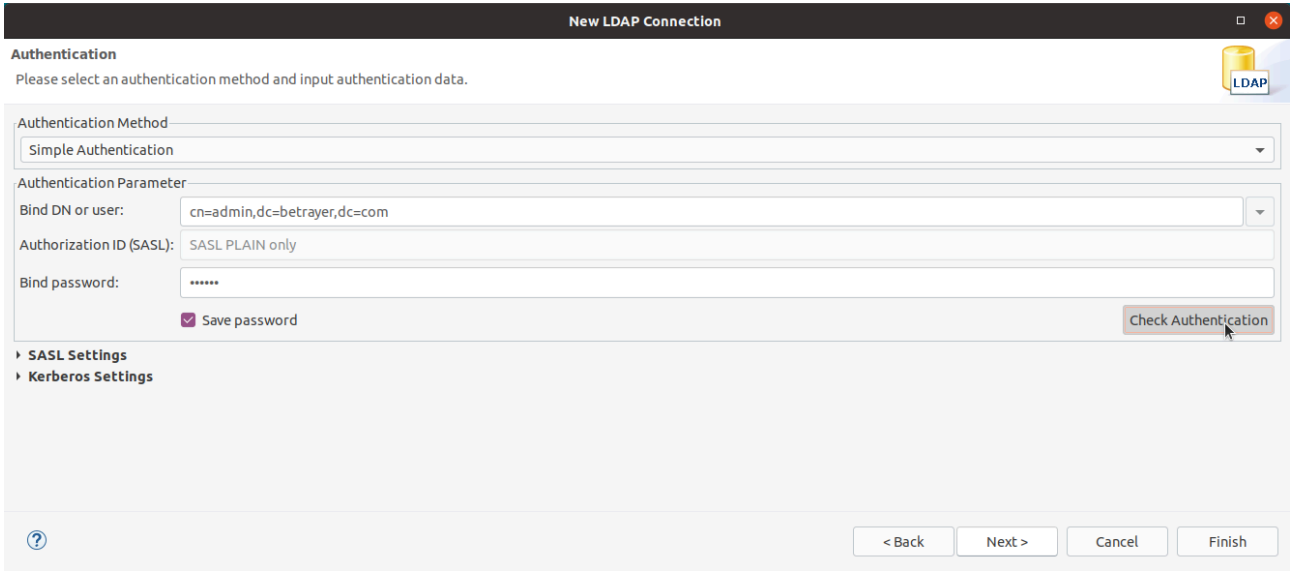
Administrator access to your server's data tree



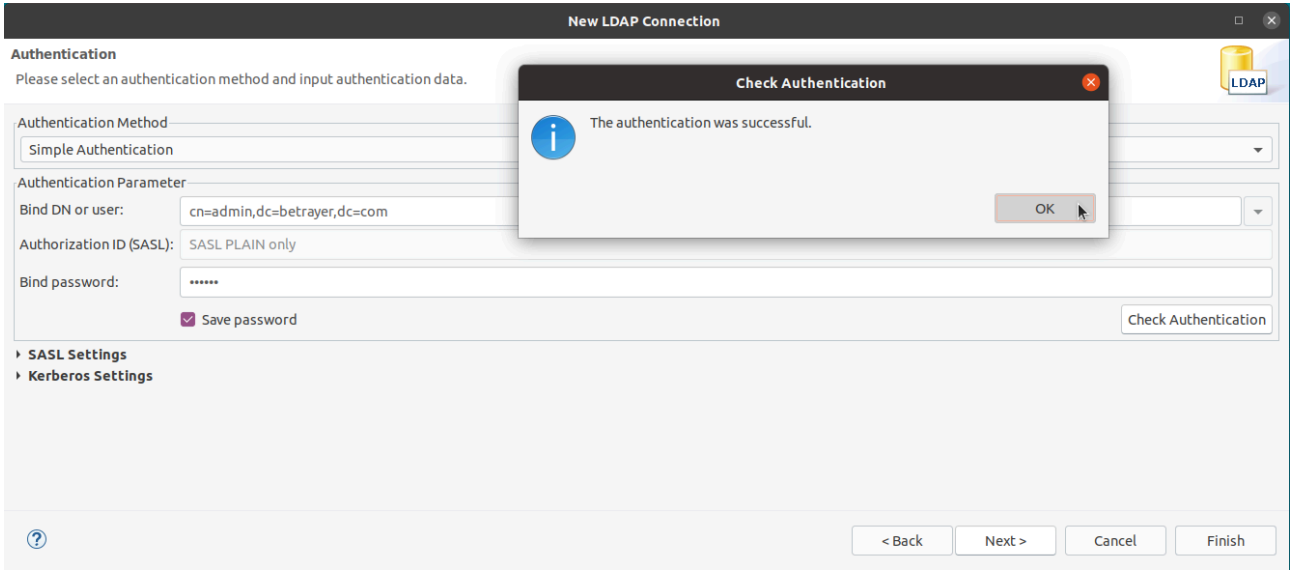
Administrator access to your server's data tree



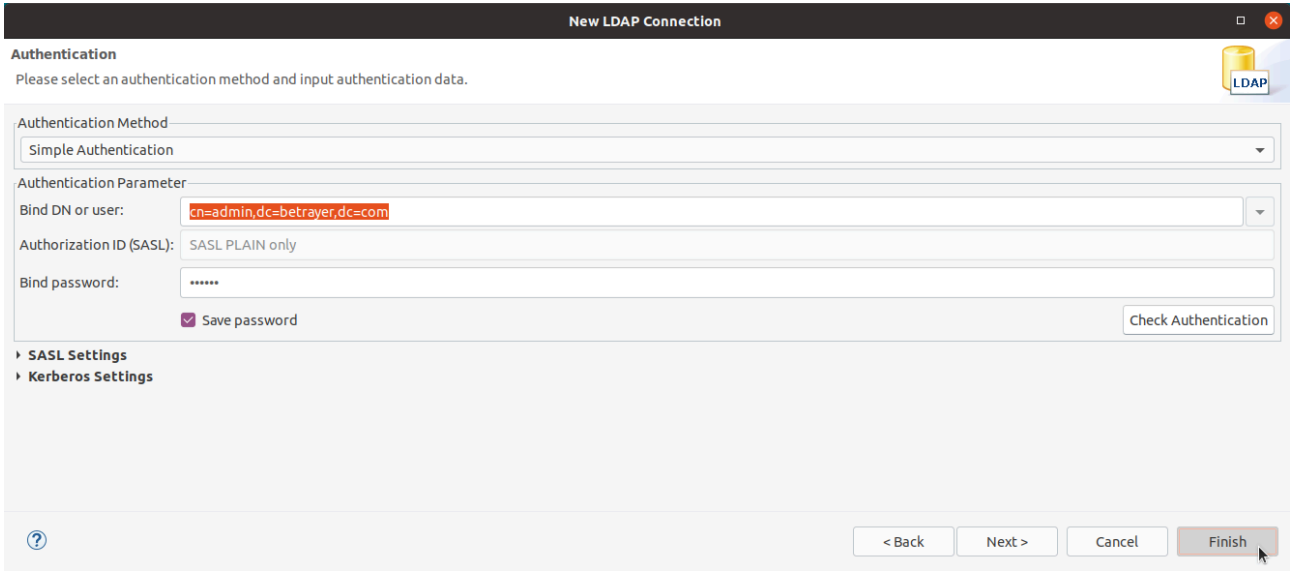
Administrator access to your server's data tree



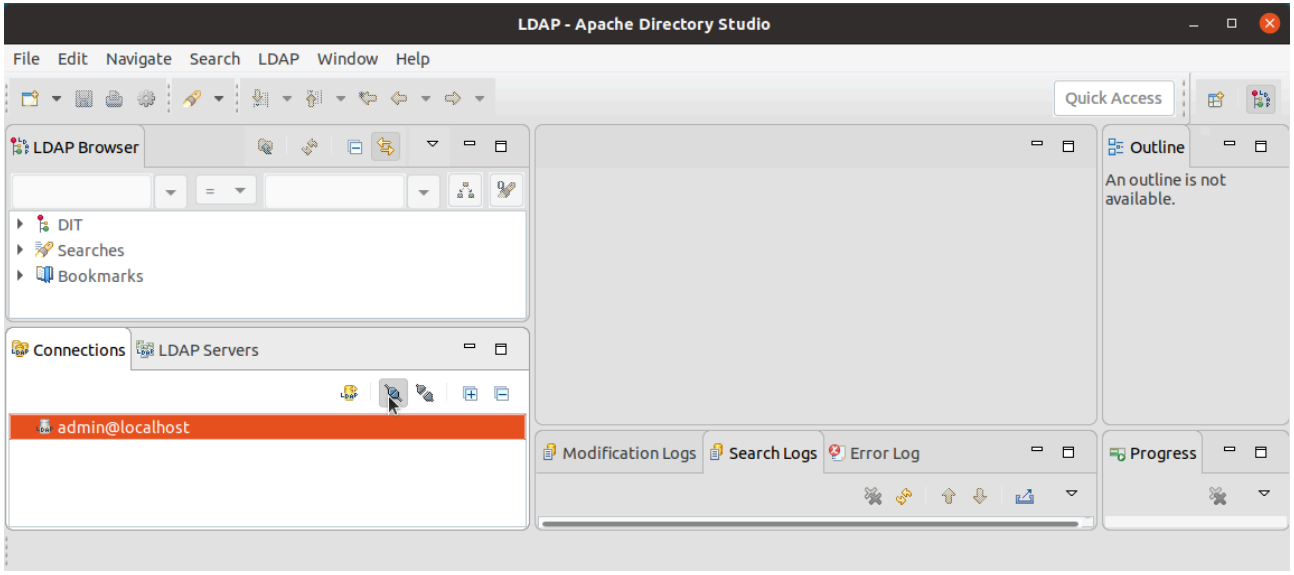
Administrator access to your server's data tree



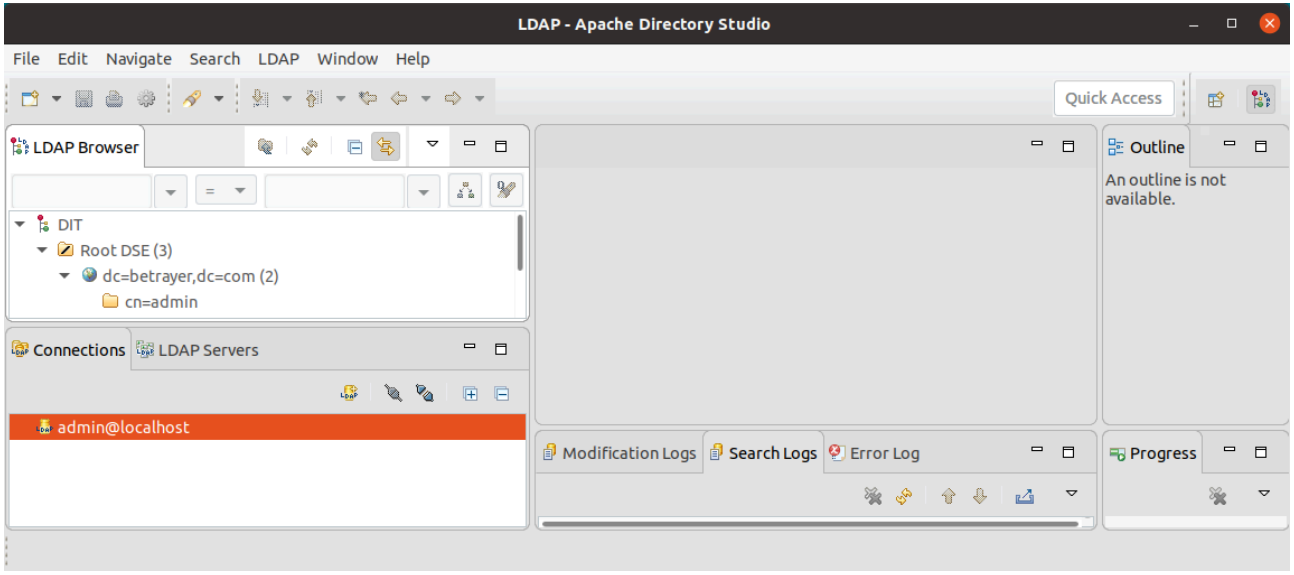
Administrator access to your server's data tree



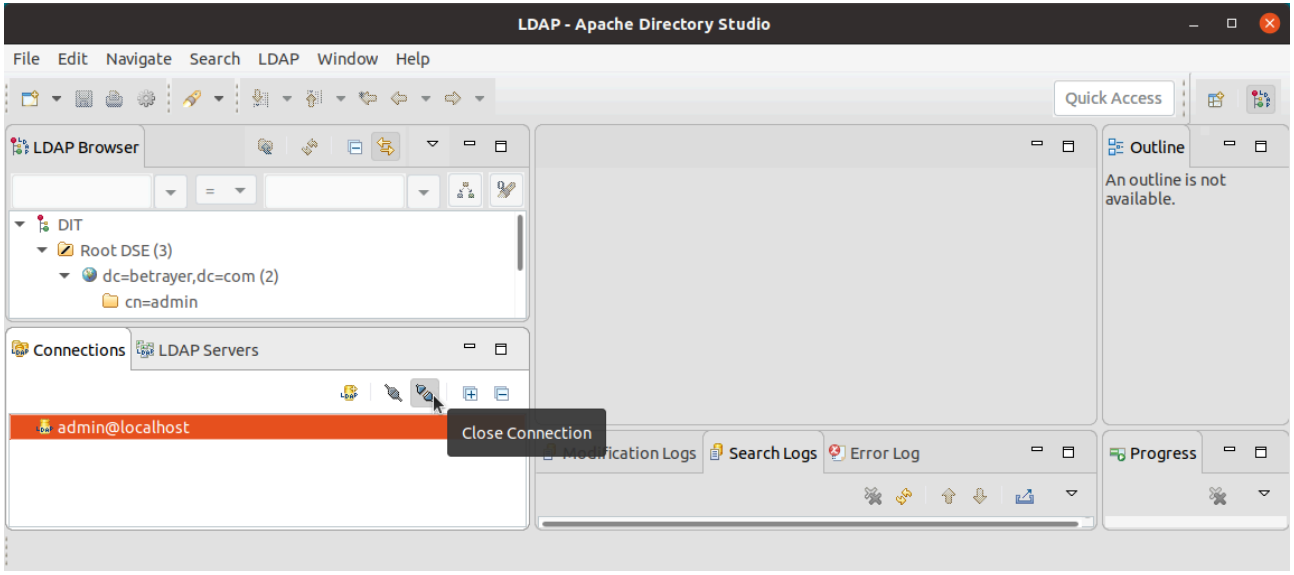
Administrator access to your server's data tree



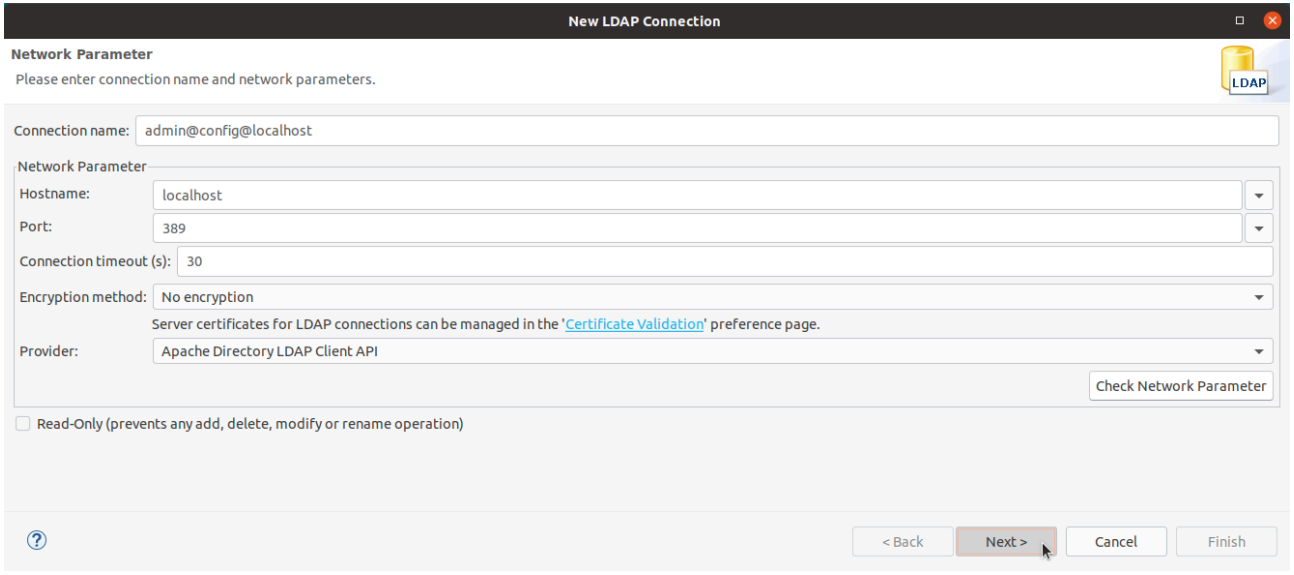
Administrator access to your server's data tree



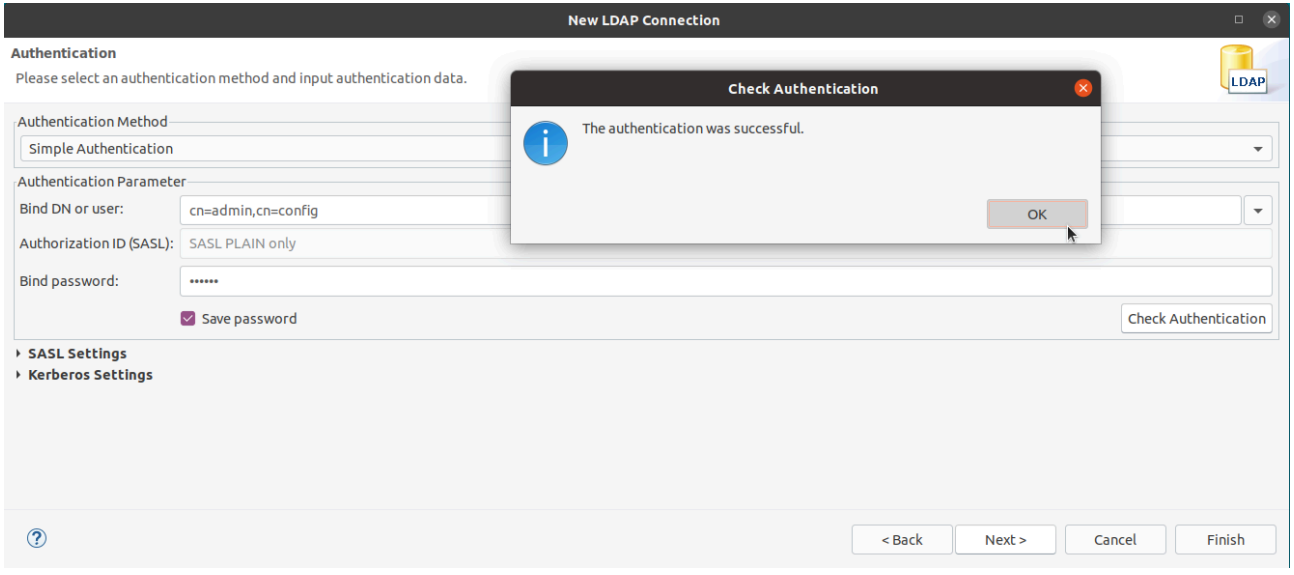
Administrator access to your server's data tree



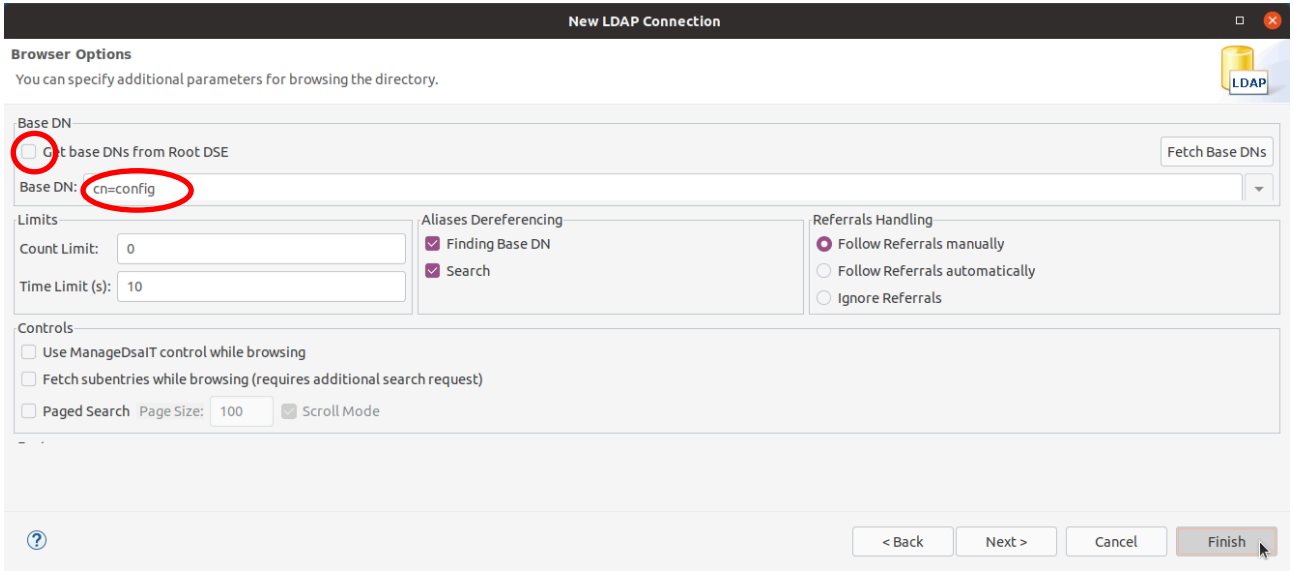
Administrator access to your server's configuration



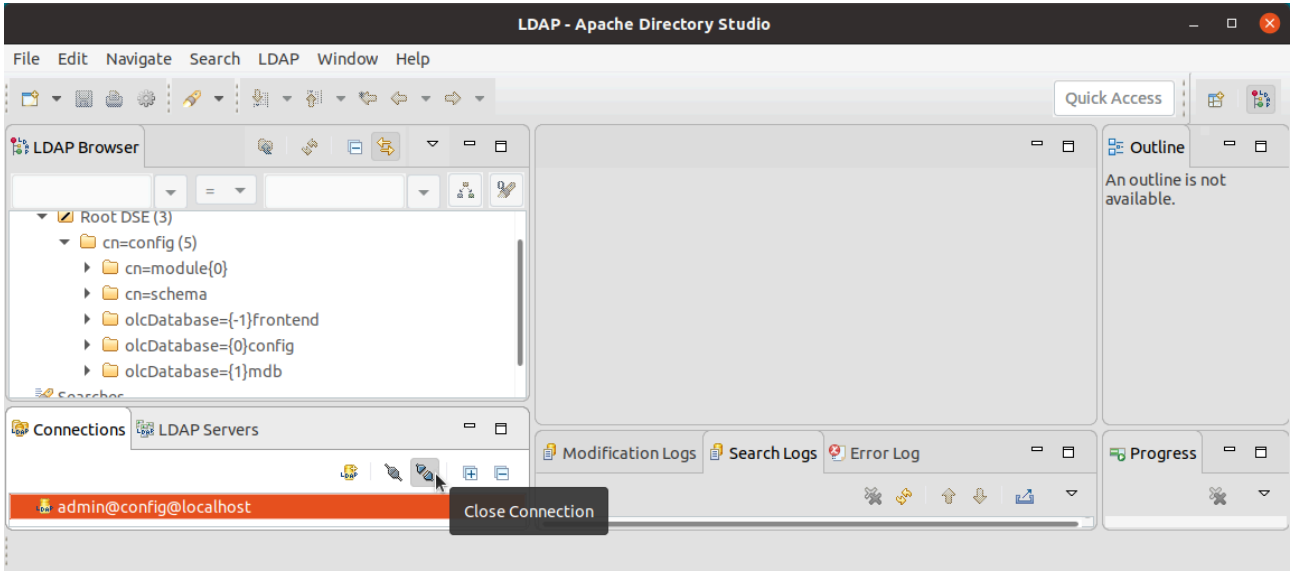
Administrator access to your server's configuration



Administrator access to your server's configuration



Administrator access to your server's configuration



Terminology

- DIT:Document information tree
- DN: An entries distinguished name. Unique identifier.
- RDN: Relative distinguished name. Unique identifier with respect to a given context node.
- Bind operation: Connect to an LDAP service.
  1. anonymous
  2. authenticated

Adding an entry

```
dn: uid=smith ❶,dc=betrayer,dc=com ❷
changetype: add ❸
objectClass: inetOrgPerson ❹
objectClass: organizationalPerson ❹
objectClass: Person ❹
objectClass: top ❹
uid: smith ❺
cn: Jill Smith ❻
sn: Smith ❻
```

Adding a new attribute

Operation	Result
dn: uid=smith,dc=betrayer,dc=com ❶ changetype: modify ❷ add: description ❸ description: New employee ❹	dn: uid=smith,dc=betrayer,dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top cn: Jill Smith sn: Smith uid: smith description: New employee

Replacing an attribute value

Operation	Result
dn: uid=smith,dc=betrayer,dc=com ❶ changetype: modify ❷ replace: description ❸ description: Long term employee ❹	dn: uid=smith,dc=betrayer,dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top cn: Jill Smith sn: Smith uid: smith description: Long term employee

Deleting an attribute entirely

Base state	Operation	Result
dn: uid=smith,dc=betrayer,dc=com ... uid: smith commonName: Jill Smith surname: Smith description: Long term employee	dn: uid=smith,dc=... changetype: modify delete: description	dn: uid=smith,dc=betrayer,dc=com ... uid: smith commonName: Jill Smith surname: Smith

Multi valued attributes

Operation	Result
dn: uid=smith,dc=betrayer,dc=com changetype: modify add: mail ❶ mail: smith@company.com ❷ mail: jsmith@privateaccount.org ❸	dn: uid=smith,dc=betrayer,dc=com ... sn: Smith mail: jsmith@privateaccount.org ❷ mail: smith@company.com ❸

Set semantics of multivalued attributes

Base state	Operation
dn: uid=smith,dc=betrayer,dc=com objectClass: inetOrgPerson objectClass: organizationalPerson objectClass: person objectClass: top uid: smith commonName: Jill Smith surname: Smith	dn: uid=smith,dc=betrayer,dc=com changetype: modify add: mail ❶ mail: smith@company.com ❷ mail: jsmith@privateaccount.org ❸ mail: smith@company.com ❹ ERR_13207_VALUE_ALREADY_EXISTS ❹ The value 'smith@company.com' already exists in the attribute (mail)

Deleting selected attribute values

Base state	Operation	Result
dn: uid=smith,dc=betrayer,dc=com ... cn: Jill Smith mail: jsmith@privateaccount.org mail: smith@company.com mail: anonymous@keeput.org	dn: uid=smith,dc=... changetype: modify delete: mail mail: smith@company.com mail: anonymous@keeput.org	dn: uid=smith,dc=betrayer,dc=com ... cn: Jill Smith mail: jsmith@privateaccount.org

Query scope

- BaseObject
- SingleLevel
- WholeSubtree
- SubordinateSubtree (not yet officially standardized)

Query filter

- Presence: `(cn=*)`
- Equality: `(uid=xy123)`
- Comparison: `(1000 < uidNumber)`
- Substring: `(surname=K*)`
- Approximate Match: `(sn~=gok)`

  (matches `Gack`, `Keck`, `Kubiac`, `Goik`, `Kabbeck`, `Nguyen Quoc`, `Gubic` , `Koc`)

- AND: `(&(5 < uidNumber)(sn=*k*)(gidNumber=1000))`
- OR: `(|(5 < uidNumber)(sn=*k*)(gidNumber=1000))`
- NOT: `(!(5 < uidNumber))`
- Extensible match filters, see Extensible Match Search Filter as well.

Schema support

- Objectclass definitions
- Objectclass inheritance. Types:
  - Abstract
  - Structural
  - Auxiliary
- Dynamic instance schema changes
- Single and multivalued attributes
- Attribute data types
- Matching rules (Case sensitive / insensitive, string match, numeric)

Implementations

```
access to attrs=matrikelNr
      by dn="uid=goik,ou=userlist,dc=hdm-stuttgart,dc=de" read
      by dn="uid=kuhn,ou=userlist,dc=hdm-stuttgart,dc=de" read
      by self read
      by * none

access to attrs=userPassword,shadowLastChange,passwordClear
      by dn="uid=Administrator,ou=people,ou=MI,ou=domainlist,dc=hdm-stuttgart,dc=de" read
      by anonymous auth
      by * none
```

Implementations

- JNDI
- Ldaptive

Exercises

1. Work through the exercises the section called “Browse an existing LDAP Server” and the section called “Populating your DIT.” to the section called “Extending an existing entry”.

   Tip

   When logging in as a non - admin user i.e. using a bind DN like `uid=petra,ou=MIB,ou=MI,dc=betrayer,dc=com` you will not be able to browse your tree. This action requires a permission setting to be changed in `olcDatabase={1}mdb,cn=config of your server's configuration tree`. Follow these steps:

   a. Log in to your server's configuration using `cn=admin,cn=config` as in Figure 19.9, “Administrator access to your server's configuration”.

   b. Select your database backend node below `cn=config`.

   c. Replace:

      ```
      to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage by * break
      ```

      By:

      ```
      to * by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage by * read
      ```

2. Find your LDAP servers database back end. Install the lmdb-utils package and dump your back end's data.

3. Create an application reading `ldap.hdm-stuttgart.de`'s data and writing corresponding student entries back to your local LDAP server.

Why MongoDB?

- Document oriented (hierarchy support).
- Horizontal scaling (Sharding)
- Large user community
- Major programming languages API support.
- Open Source

Running a Docker container

```
docker run -d \
  --name localMongoDb ❶\
  -e MONGO_INITDB_ROOT_USERNAME=admin ❷\
  -e MONGO_INITDB_ROOT_PASSWORD=secret ❷\
  -e MONGO_INITDB_DATABASE=admin ❸\
  -v ~/Data/Mongo:/data/db ❹\
  -p 27017:27017 ❺\
  mongo:4.4.1 ❻
```

Using docker-compose

docker-compose.yml

```
version: '3.7'

services:
    mongodb:
        image: mongo:4.4.1
        container_name: mongodb
        restart: always
        environment:
            MONGO_INITDB_ROOT_USERNAME: admin
            MONGO_INITDB_ROOT_PASSWORD: secret
        ports:
            - 27017:27017
        volumes:
            - ./mongo-init.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
```

mongo-init.js

```
db.createUser(
    {
        user: "explorer",
        pwd: "secret",
        roles: [
            {
                role: "readWrite",
                db: "exploredb"
            }
        ],
        passwordDigestor: "server"
    }
);
```

```
docker-compose up --build -d
```

Manual user creation (mongo-init.js fail)

```
> mongo -u admin -p secret admin
...
 db.createUser(
...     {
...         user: "explorer",
...         pwd: "secret",
...         roles: [
...             {
...                 role: "readWrite",
...                 db: "exploredb"
...             }
...         ],
...         passwordDigestor: "server"
...     }
... );
Successfully added user: { "user" : "explorer"...
```

Log in as user explorer

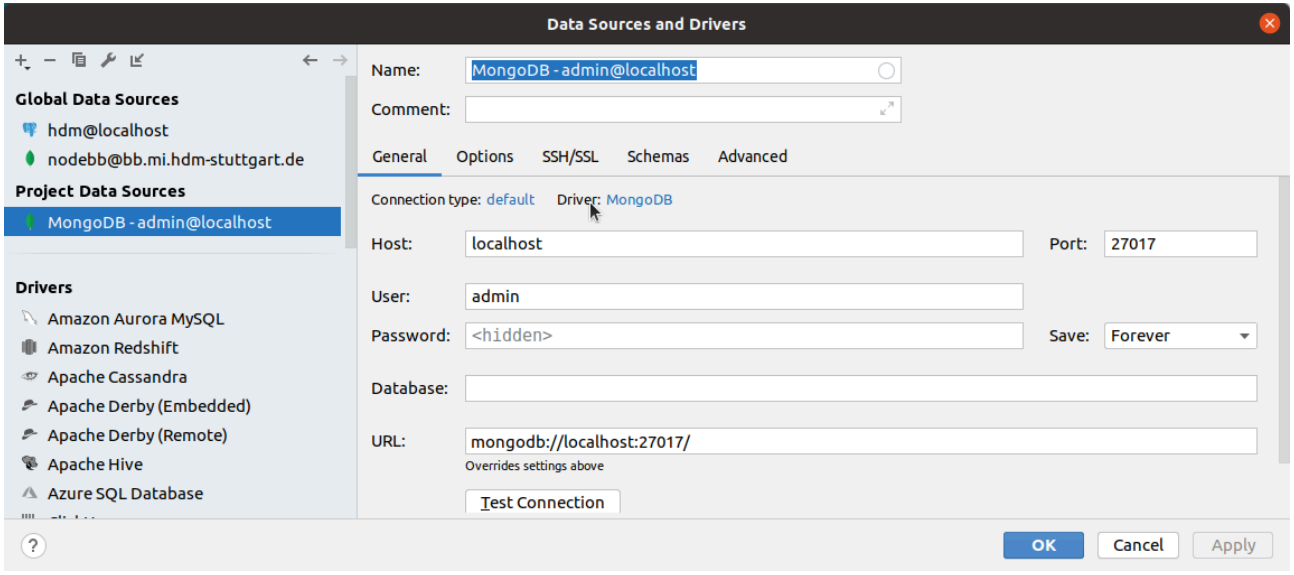
```
> mongo -u explorer -p secret admin
...
> use exploredb
switched to db exploredb

> db.user.insert(
...     { cname: "Eve Gardener",
...         uid: "gardener",
...         email: "gardener@betrayer.com"
...     }
... )
WriteResult({ "nInserted" : 1 })
>
> db.user.find()
{ "_id" : ObjectId("5fa1c79d661a55242658f135"),
  "cname" : "Eve Gardener", "uid" : "gardener", "email" : "gardener@betrayer.com" }
```

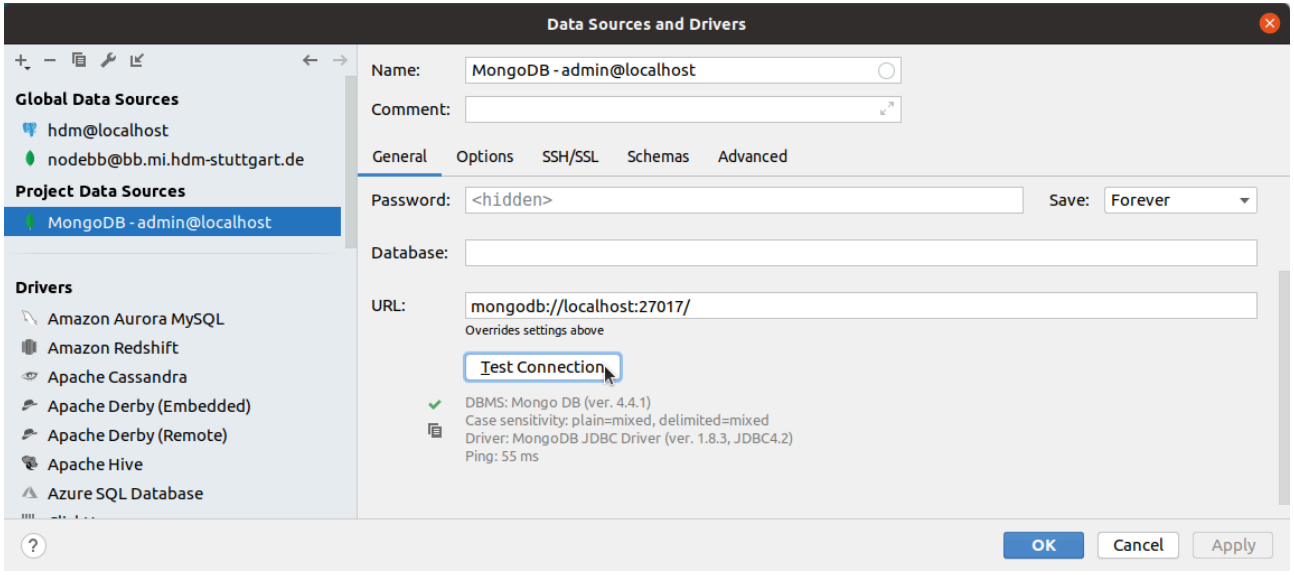
Using IntelliJ

- View --> Tool Windows --> Database
- Data Source --> MongoDB

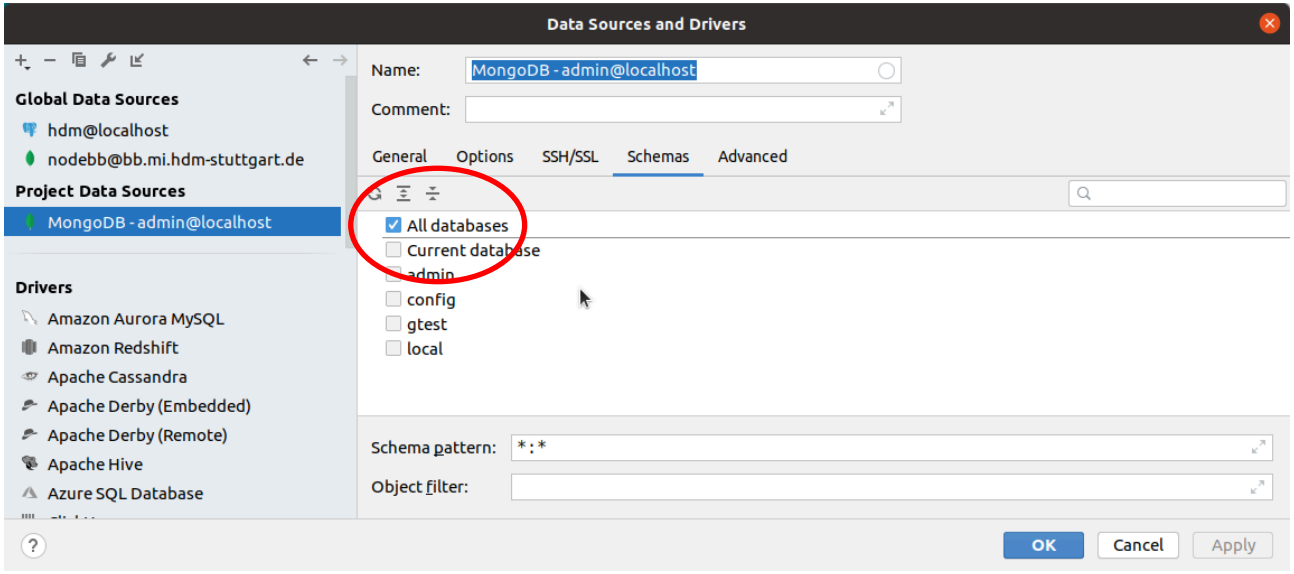
Idea show all databases



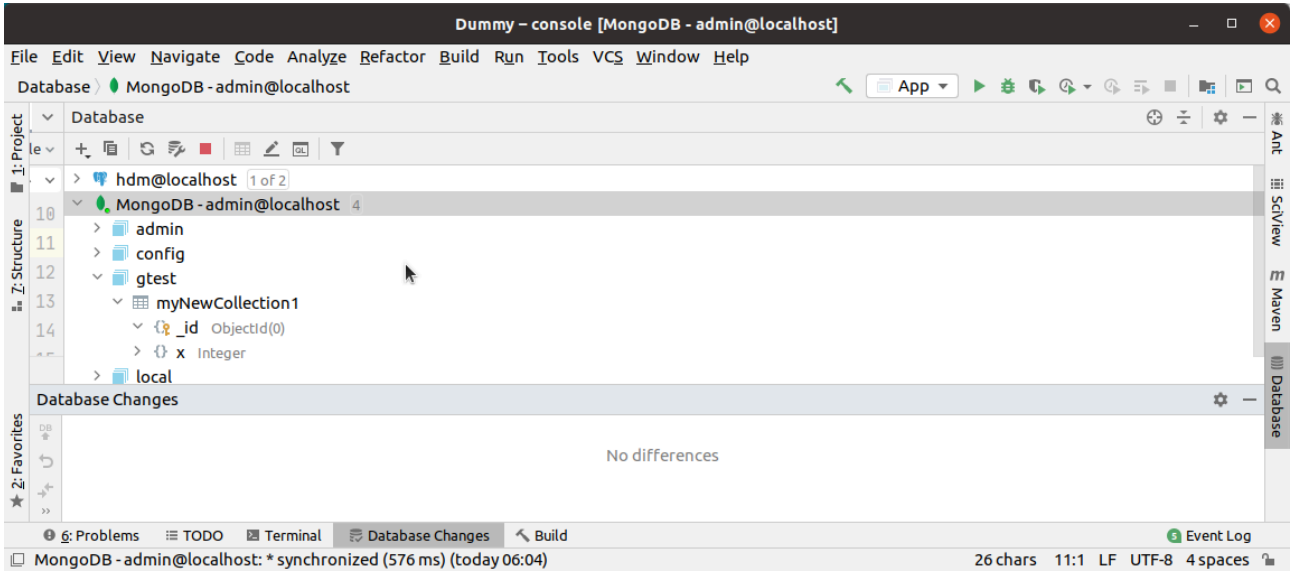
Idea show all databases



Idea show all databases



Idea show all databases



Terminology / Hierarchy

- MongoDB Server
  - **Database**
    - **Collection** : Similar to tables in SQL.
      - **Document**: Similar to records in SQL.

```
> mongo -u explorer -p secret admin

> use exploredb

> db.user.insert(
    { cname: "Eve Gardener",
      uid: "gardener",
      email: "gardener@betrayer.com"
    }
 )
```

Adding a document

Code	Result

db.group.insert(
    {
        cname: "All users",
        gid: "users",
    }
)
	db.group.find()
	
	[
	  {
	    "_id": {"$oid": "5fa3035932b87a0c60a6ed1a"},
	    "cname": "New users",
	    "gid": "users"
	  }
	]

Updating attributes

Code	Result

db.group.update(
{_id: ObjectId(
"5fa3035932b87a0c60a6ed1a")},
{ $set:
 	
{
cname: "New users" ❶,
gidNumber: 1000 ❷
}
}
)
	db.group.find()
	
	[
	{
	"_id": {"$oid":
	"5fa3035932b87a0c60a6ed1a"},
	cname: "New users" ❶,
	"gid": "users",
	"gidNumber": 1000 ❷
	}
	]

Deleting a document

Code	Result
`db.group.deleteOne( {"_id": ObjectId( "5fa3035932b87a0c60a6ed1a")});`	`[ { "acknowledged": true, "deletedCount": 1 } ]`

Deleting multiple documents

Code	Result

`db.group.deleteMany({});`	`[`
	`  {`
	`  "acknowledged": true,`
	`  "deletedCount": 23`
	`  }`
	`]`

Multi valued attributes

Set semantics of multivalued attributes

Deleting selected attribute values

Deleting an attribute

Code	Result

`db.group.update(`
`  {_id: ObjectId(`
`  "5fa3035932b87a0c60a6ed1a")},`
 	
`  { $unset:`
`  {`
`  gidNumber: 42 ❶`
`  }`
`  }`
`)`
	`db.group.find()`
	
	`[`
	`  {`
	`  "_id": {"$oid":`
	`  "5fa3035932b87a0c60a6ed1a"},`
	`  "cname": "My users",`
	`  "gid": "users"`
	`  }`
	`]`

Query filter

Schema validation support

```
db.runCommand( {
  collMod: "group",
  validator: { $jsonSchema: {
     bsonType: "object",
     required: [ "cname", "gid" ],
        properties: {
          cname: {
              bsonType: "string",
              description:
              "A group's common name"
          },
          gid: {
             bsonType: "string",
             description:
             "A group's short name"
          }
       }
     }
  },
  validationLevel: "moderate"
})
```

Violating required field

```
db.group.insert(
 {
   cname: "Extra users"
 }
```

```
... Bulk write operation error on server localhost:27017.
Write errors: [BulkWriteError{index=0, code=121,
    message='Document failed validation', details={}}].
```

Schema types

See BSON Types for reference.

Enforcing unique keys

```
db.group.createIndex({
   "cname": 1
  },
  {
      unique: true
})

db.group.insert({cname: "Extra users",
  ...
  }
)
```

```
com.mongodb.MongoBulkWriteException:
Bulk write operation error on server localhost:27017.
Write errors: [BulkWriteError{index=0, code=11000,
message='E11000 duplicate key error collection:
exploredb.group index: cname_1 dup key:
  { cname: "Extra users" }', details={}}].
```

On the downside

- No way to enforce referential integrity rules.

Implementations

Implementations

-
-

Sharding rationale

- Problem: Large datasets / high throughput
- Two alternatives:
  - Vertical scaling: RAM, cpu,...
  - Horizontal scaling: Load distribution by multiple nodes.

Sharding rationale

See sharded-cluster for details.

Exercises

1.

2.