# Umsetzung

Beispieltext und kurze Worte zur Umsetzung

## Authentifizierungsstelle LDAP

Um Benutzer in der SmartVal API zu authentifizieren, wurde LDAP auf einem Server installiert. Die SmartVal API stellt bei einer Anfrage eine Verbindung mit dem Server her, um eine Authentifizierung durchzuführen mittels Benutzername und Passwort. Sind die Zugangsdaten richtig und hat der Benutzer die Benötigten Rechte, so wird ein entsprechender Response vom Server zurückgegeben.

## LDAP Server

Zur Installation des LDAP wurde OpenLDAP verwendet. Diese Software wurde auf folgendem Server installiert:

|  |  |
| --- | --- |
| Host: | landsteiner.fh-salzburg.ac.at |
| IP: | 193.170.119.66 |
| User: | Administrator |
| Passwort: | 9HxAJ39g8EnudLS7 |

Tabelle 1: Informationen zu Authentifizierungsserver LDAP

## OpenLDAP konfiguration

Um eine Verbindung zu OpenLDAP herstellen zu können, musste in der Software die Verbindung angelegt werden. Hierzu wurden die folgenden Informationen vergeben:

|  |  |  |  |
| --- | --- | --- | --- |
| Host: | 193.170.119.66 | Port: | 389 |
| Base: | dc=maxcrc,dc=com | Version: | 3 |
| Username: | cn=Manager,dc=maxcrc,dc=com | Simple Authentication | |
| Password: | secret |  | |

Tabelle 2: Zugangsdaten zu OpenLDAP

Weiters musste ein Zertifikat „development“ installiert werden. Dieses Zertifikat wurde in der Management Konsole unter „Vertrauenswürdige Zertifikate“ und „persönliche Zertifikate“ im lokalen Computerkonto hinzugefügt.

Um nun eine Hierarchie im OpenLDAP einzurichten, wurden die Kategorien

* Groups,
* People und
* Smartmeter

hinzugefügt. Somit konnte eine erste Unterscheidung zwischen Gruppen, Benutzern und Smartmeter hergestellt werden. In der Kategorie Groups wurden des Weiteren die Gruppen

* Administrator,
* Energieberater,
* Forschungszentrum,
* Kunden und
* Netzbetreiber

hinzugefügt. Somit konnte die Hierarchie zwischen Netzbetreiber über Energieberater bis hin zum einzelnen Smartmeter ermöglicht werden.

Mit diesen Einstellungen war der Server fertig aufgesetzt.

## Schnittstelle zur Authentifizierung in SmartVal API

Die Schnittstelle zur Authentifizierung wurde in zwei Klassen (LDAPManager und LdapContextSourceFactory) sowie in einem Interface (ILDAPManager) unterteilt.

Wie schon erwähnt, wurde SmartVal API mithilfe von Spring erstellt. Mit Spring war es möglich, die Frameworks

* org.springframework.security:spring-security-ldap und
* org.springframework.ldap:spring-ldap-core

zu ergänzen, welche ein erleichtertes Arbeiten mit LDAP ermöglicht.

Im ersten Schritt muss eine Verbindung zum OpenLDAP Server hergestellt werden.

**LdapContextSourceFactory**

In der Klasse LdapContextSourceFactory wurde die Funktion

*public static LdapContextSource getContextSource();*

implementiert. Dieser Funktion wurden die Zugangsdaten ersichtlich aus *Tabelle 2* hinzugefügt. Bei Aufruf dieser Funktion wird so eine Verbindung direkt zum OpenLDAP hergestellt.

**LDAPManager**

Um alle Funktionen für die restliche API bereitzustellen, wurde das Interface ILDAPManager erstellt. Dieses Interface enthält alle Funktionen, welche betreffend die Authentifizierung als auch Erstellung oder Löschung von Daten auf dem LDAP Server liefert. Ausprogrammiert wurden alle Funktionen in der Klasse LDAPManager implementiert.

Alle folgenden Funktionen geben ein boolean true, wenn die Funktion problemlos durchgeführt wurde und false, falls es zu einem Fehler gibt zurück. Die Parameter IDataSourceContext als auch IUserContext werden aus der API mitgegeben, welche alle benötigten Daten mitliefern.

Von dieser Schnittstelle werden die folgenden Funktionen bereitgestellt:

* *boolean IsAllowedToAccess(IUserContext userContext, IDataSourceContext dataSourceContext);*

IsAllowedToAccess prüft, ob der mitgegebenen Benutzer die gewollten Berechtigungen besitzt. Wird false zurückgegeben, hat er keine Berechtigung. Hierbei kann direkt der Kunde, oder auch der Energieberater angegeben werden, welcher den Kunden des zugehörigen Smartmeters berät. Ebenso kann mit Mitgabe des Netzbetreibers ein Zugriff geprüft werden.

* *boolean CreateCustomer(IUserContext userContext, IDataSourceContext dataSourceContext);*

Diese Funktion bietet die Möglichkeit, einen neuen Kunden am LDAP Server anzulegen. Bei Erstellung eines Kunden muss ein Smartmeter mitgegeben werden.

* *boolean CreateConsultant(IUserContext source, IUserContext destination);*

Die Funktion CreateConsultant lässt einen Energieberater erstellen.

* *boolean CreateSmartMeter(IDataSourceContext dataSourceContext);*

Mit CreateSmartMeter wird ein neuer Smartmeter am OpenLDAP hinzugefügt.

* *boolean AddUserToGroup(IUserContext userContext, String Group);*

Diese Funktion fügt den mitgegebenen Benutzer einer Gruppe hinzu. Hiermit kann eine Hierarchie aufgebaut werden. Beispielsweise kann mit dieser Funktion ein Kunde X zur Gruppe Kunden hinzugefügt werden oder ein Energieberater Y zur Gruppe Netzbetreiber.

* *boolean AddUserToUser(IUserContext source, IUserContext destination);*

AddUserToUser bietet die Möglichkeit, beispielsweise einen Kunden zu einem Energieberater hinzuzufügen. Somit kann die Hierarchie besser umgesetzt werden.

* *boolean AddMeterToUser(IUserContext userContext, IDataSourceContext dataSourceContext);*

Mit dieser Funktion wird ein Smartmeter einem Kunden hinzugefügt.

* *boolean DeleteUser(IUserContext userContext);*

DeleteUser löscht einen Benutzer aus dem OpenLDAP.

* *boolean DeleteUserFromGroup(IUserContext userContext, String Group);*

Diese Funktion Löscht einen Benutzer aus einer spezifischen Gruppe.

* *boolean DeleteUserFromAll(IUserContext userContext);*

Diese Funktion ist eine Hilfsfunktion von DeleteUser. Diese Funktion löscht den gelöschten Benutzer aus allen Einträgen und hält die Einträge sauber.

* *boolean DeleteMeterFromUser(IUserContext userContext, IDataSourceContext dataSourceContext);*

DeleteMeterFromUser entfernt den mitgegebenen Smartmeter aus der Mitgliedschaft des mitgegebenen Users.

* *boolean DeleteSmartMeter(IDataSourceContext dataSourceContext);*

Mit dieser Funktion wird der mitgegebene Smartmeter aus dem OpenLDAP gelöscht.

* *boolean DeleteMeterfromAll(IDataSourceContext dataSourceContext);*

Diese Funktion ist eine Hilfsfunktion von DeleteSmartMeter und löscht den mitgegebenen Smartmeter aus allen Einträgen des OpenLDAP.

Mithilfe dieser Funktionen ist eine saubere Pflege des OpenLDAP möglich als auch eine Authentifizierungsoption, welche von der SmartVal API benötigt wird. Der Quellcode wurde im Anhang beigefügt.

Anhang

package at.ac.fh.salzburg.smartmeter.ldap;

import at.ac.fh.salzburg.smartmeter.access.IDataSourceContext;

import at.ac.fh.salzburg.smartmeter.access.IUserContext;

import org.springframework.ldap.core.AttributesMapper;

import org.springframework.ldap.core.LdapTemplate;

import org.springframework.ldap.query.LdapQueryBuilder;

import org.springframework.stereotype.Component;

import javax.naming.InvalidNameException;

import javax.naming.directory.\*;

import javax.naming.ldap.LdapName;

@Component

public class LDAPManager implements ILDAPManager {

private static LdapTemplate ldapTemplate = new LdapTemplate(LdapContextSourceFactory.getContextSource());

/\*

public static void main(String[] args) {

UserContext cons1 = new UserContext("consultant2","consultant2");

UserContext cust1 = new UserContext("customer5","customer5");

//IDataSourceContext sm1 = () -> "5";

LDAPManager manager = new LDAPManager();

//manager.CreateConsultant(cust1,cons1);

manager.AddUserToUser(cust1,cons1);

//manager.AddUserToGroup(cons1,"Energieberater");

}

\*/

@Override

public boolean CreateCustomer(IUserContext userContext, IDataSourceContext dataSourceContext){

try{

LdapName dn = new LdapName("uid="+userContext.userid()+",ou=People");

//User Attributes

Attribute userCn = new BasicAttribute("cn", userContext.userid());

Attribute userPassword = new BasicAttribute("userPassword",userContext.password());

Attribute UserID = new BasicAttribute("uidNumber","0");

Attribute Usergid = new BasicAttribute("gidNumber","0");

Attribute Userhome = new BasicAttribute("homeDirectory","/home/user/");

//ObjectClass attributes

Attribute oc = new BasicAttribute("objectClass");

oc.add("top");

oc.add("PosixAccount");

oc.add("PosixGroup");

Attributes entry = new BasicAttributes();

entry.put(userCn);

entry.put(Userhome);

entry.put(Usergid);

entry.put(UserID);

entry.put(userPassword);

entry.put(oc);

ldapTemplate.bind(dn, null, entry);

CreateSmartMeter(dataSourceContext);

AddMeterToUser(userContext,dataSourceContext);

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean CreateConsultant(IUserContext Customer, IUserContext Consultant){

try{

LdapName dn = new LdapName("uid="+Consultant.userid()+",ou=People");

//User Attributes

Attribute userCn = new BasicAttribute("cn", Consultant.userid());

Attribute userPassword = new BasicAttribute("userPassword",Consultant.password());

Attribute UserID = new BasicAttribute("uidNumber","0");

Attribute Usergid = new BasicAttribute("gidNumber","0");

Attribute Userhome = new BasicAttribute("homeDirectory","/home/user/");

//ObjectClass attributes

Attribute oc = new BasicAttribute("objectClass");

oc.add("top");

oc.add("PosixAccount");

oc.add("PosixGroup");

Attributes entry = new BasicAttributes();

entry.put(userCn);

entry.put(Userhome);

entry.put(Usergid);

entry.put(UserID);

entry.put(userPassword);

entry.put(oc);

ldapTemplate.bind(dn, null, entry);

AddUserToUser(Customer, Consultant);

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean CreateSmartMeter(IDataSourceContext dataSourceContext) {

try{

LdapName dn = new LdapName("uid="+dataSourceContext.MeterID()+",ou=Smartmeter");

//User Attributes

Attribute userCn = new BasicAttribute("cn", dataSourceContext.MeterID());

Attribute UserID = new BasicAttribute("uidNumber",dataSourceContext.MeterID());

Attribute Usergid = new BasicAttribute("gidNumber","0");

Attribute Userhome = new BasicAttribute("homeDirectory","/home/smartmeter/");

//ObjectClass attributes

Attribute oc = new BasicAttribute("objectClass");

oc.add("top");

oc.add("PosixAccount");

oc.add("PosixGroup");

Attributes entry = new BasicAttributes();

entry.put(userCn);

entry.put(Userhome);

entry.put(Usergid);

entry.put(UserID);

entry.put(oc);

ldapTemplate.bind(dn, null, entry);

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean DeleteSmartMeter(IDataSourceContext dataSourceContext) {

try{

LdapName dn = new LdapName("uid="+dataSourceContext.MeterID()+",ou=Smartmeter");

ldapTemplate.unbind(dn);

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean DeleteUser(IUserContext userContext) {

try{

LdapName dn = new LdapName("uid="+userContext.userid()+",ou=People");

ldapTemplate.unbind(dn);

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean IsAllowedToAccess(IUserContext userContext, IDataSourceContext dataSourceContext) {

try {

final String[] cntemp = new String[1];

final String[] gruppe = new String[1];

//Hol die größte Gruppenzugehörigkeit

ldapTemplate.search(

LdapQueryBuilder.query().where("memberUid").is(userContext.userid()),

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("cn");

if(nameAttr.contains("Kunden")){

gruppe[0] = "cn: Kunden";

}

if(nameAttr.contains("Energieberater")){

gruppe[0] = "cn: Energieberater";

}

if(nameAttr.contains("Netzbetreiber")){

gruppe[0] = "cn: Netzbetreiber";

}

return null;

});

if(gruppe[0].contains("cn: Netzbetreiber")){

LdapName dn = new LdapName("uid="+userContext.userid()+",ou=People");

List consultants = new ArrayList();

List customers = new ArrayList();

//Holt alle Energieberater des Netzbetreibers

ldapTemplate.lookup(

dn,

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("memberUid");

cntemp[0] = nameAttr.toString();

String[] parts = cntemp[0].split("[:,]");

for(int i = 1; i< parts.length; i++){

consultants.add(parts[i].toString());

}

return null;

});

for(int i = 0; i < consultants.size(); i++) {

LdapName dn2 = new LdapName("uid="+consultants.get(i).toString()+",ou=People");

//Holt alle Kunden aller Energieberater

ldapTemplate.lookup(

dn2,

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("memberUid");

cntemp[0] = nameAttr.toString();

String[] parts = cntemp[0].split("[:,]");

for(int n = 1; n< parts.length; n++){

customers.add(parts[n].toString());

}

return null;

});

//Abfrage, ob deren Mitglieder den angegebenen Smartmeter besitzen

int check = 0;

for(int n = 0; n < customers.size(); n++) {

ldapTemplate.search( LdapQueryBuilder.query().where("memberUid").is(dataSourceContext.MeterID()),

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("uid");

cntemp[0] = nameAttr.toString();

return null;

});

try {

if (cntemp[0].equals("uid:" + customers.get(n))) {

check = 1;

break;

} else {

check = 0;

}

} catch (NullPointerException ne) {

//

}

}

if(check == 1){

return true;

}

else{

return false;

}

}

}

else if(gruppe[0].contains("cn: Energieberater")){

LdapName dn = new LdapName("uid="+userContext.userid()+",ou=People");

List mitglieder = new ArrayList();

//Holt alle Mitglieder des Energieberaters

ldapTemplate.lookup(

dn,

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("memberUid");

cntemp[0] = nameAttr.toString();

String[] parts = cntemp[0].split("[:,]");

for(int i = 1; i< parts.length; i++){

mitglieder.add(parts[i].toString());

}

return null;

});

//Abfrage, ob deren Mitglieder den angegebenen Smartmeter besitzen

int check = 0;

for(int i = 0; i < mitglieder.size(); i++) {

ldapTemplate.search( LdapQueryBuilder.query().where("memberUid").is(dataSourceContext.MeterID()),

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("uid");

cntemp[0] = nameAttr.toString();

return null;

});

try {

if (cntemp[0].equals("uid:" + mitglieder.get(i))) {

check = 1;

break;

} else {

check = 0;

}

} catch (NullPointerException ne) {

//

}

}

if(check == 1){

return true;

}

else{

return false;

}

}

else if(gruppe[0].contains("cn: Kunden")){

ldapTemplate.search( LdapQueryBuilder.query().where("memberUid").is(dataSourceContext.MeterID()),

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("uid");

cntemp[0] = nameAttr.toString();

return null;

});

try {

if (cntemp[0].equals("uid: " + userContext.userid())) {

return true;

} else {

return false;

}

}

catch(NullPointerException ne){

//

}

}

} catch (Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean AddMeterToUser(IUserContext userContext, IDataSourceContext dataSourceContext) {

try {

LdapName dn = new LdapName("uid="+userContext.userid()+",ou=People");

Attribute MeterID = new BasicAttribute("memberUid", dataSourceContext.MeterID());

ModificationItem ID = new ModificationItem(

DirContext.ADD\_ATTRIBUTE, MeterID);

ldapTemplate.modifyAttributes(dn, new ModificationItem[]{ID});

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean AddUserToUser(IUserContext Customer, IUserContext Consultant) {

try {

LdapName dn = new LdapName("uid=" + Consultant.userid() + ",ou=People");

Attribute UserID = new BasicAttribute("memberUid", Customer.userid());

ModificationItem ID = new ModificationItem(

DirContext.ADD\_ATTRIBUTE, UserID);

ldapTemplate.modifyAttributes(dn, new ModificationItem[]{ID});

return true;

} catch (InvalidNameException e) {

e.printStackTrace();

}

return false;

}

@Override

public boolean AddUserToGroup(IUserContext userContext, String Group) {

try {

LdapName dn = new LdapName("cn="+Group+",ou=Groups");

Attribute UserID = new BasicAttribute("memberUid", userContext.userid());

ModificationItem ID = new ModificationItem(

DirContext.ADD\_ATTRIBUTE, UserID);

ldapTemplate.modifyAttributes(dn, new ModificationItem[]{ID});

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean DeleteMeterFromUser(IUserContext userContext, IDataSourceContext dataSourceContext) {

try {

LdapName dn = new LdapName("uid="+userContext.userid()+",ou=People");

Attribute MeterID = new BasicAttribute("description", dataSourceContext.MeterID());

ModificationItem ID = new ModificationItem(

DirContext.REMOVE\_ATTRIBUTE, MeterID);

ldapTemplate.modifyAttributes(dn, new ModificationItem[]{ID});

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean DeleteMeterfromAll(IDataSourceContext dataSourceContext) {

try {

final String[] test = new String[1];

ldapTemplate.search(

LdapQueryBuilder.query().where("memberUid").is(dataSourceContext.MeterID()),

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("cn");

System.out.printf("%s%n",

nameAttr == null ? "" : nameAttr.get()

);

test[0] = nameAttr.toString();

final String[] parts = test[0].split(":");

String cntemp = parts[1];

try {

LdapName dn = new LdapName("uid="+cntemp.substring(1)+",ou=People");

Attribute MeterID = new BasicAttribute("memberUid", dataSourceContext.MeterID());

ModificationItem ID = new ModificationItem(DirContext.REMOVE\_ATTRIBUTE, MeterID);

ldapTemplate.modifyAttributes(dn, new ModificationItem[]{ID});

}

catch(Exception e){

e.printStackTrace();

}

return null;

});

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean DeleteUserFromGroup(IUserContext userContext, String Group) {

try {

LdapName dn = new LdapName("cn="+Group+",ou=Groups");

Attribute UserID = new BasicAttribute("memberUid", userContext.userid());

ModificationItem ID = new ModificationItem(

DirContext.REMOVE\_ATTRIBUTE, UserID);

ldapTemplate.modifyAttributes(dn, new ModificationItem[]{ID});

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

@Override

public boolean DeleteUserFromAll(IUserContext userContext) {

try {

final String[] test = new String[1];

ldapTemplate.search(

LdapQueryBuilder.query().where("memberUid").is(userContext.userid()),

(AttributesMapper<Void>) attrs -> {

Attribute nameAttr = attrs.get("cn");

System.out.printf("%s%n",

nameAttr == null ? "" : nameAttr.get()

);

test[0] = nameAttr.toString();

final String[] parts = test[0].split(":");

String cntemp = parts[1];

try {

LdapName dn = new LdapName("cn="+cntemp.substring(1)+",ou=Groups");

Attribute UserID = new BasicAttribute("memberUid", userContext.userid());

ModificationItem ID = new ModificationItem(DirContext.REMOVE\_ATTRIBUTE, UserID);

ldapTemplate.modifyAttributes(dn, new ModificationItem[]{ID});

}

catch(Exception e){

e.printStackTrace();

}

return null;

});

return true;

}

catch(Exception e){

e.printStackTrace();

}

return false;

}

}