

Introduction à Spring Ldap 1.2

Introduction

L'objet de ce tutorial est de montrer comment accéder simplement aux données d'un serveur Ldap via Spring Ldap 1.2.

Il n'est pas nécessaire d'utiliser ou de connaître Spring pour utiliser ce tutorial.

Il n'est pas non plus nécessaire d'installer un serveur Ldap, car nous utiliserons le serveur public de www.openldap.com.

Vous trouverez les sources complètes de ce tutorial à la fin du document.

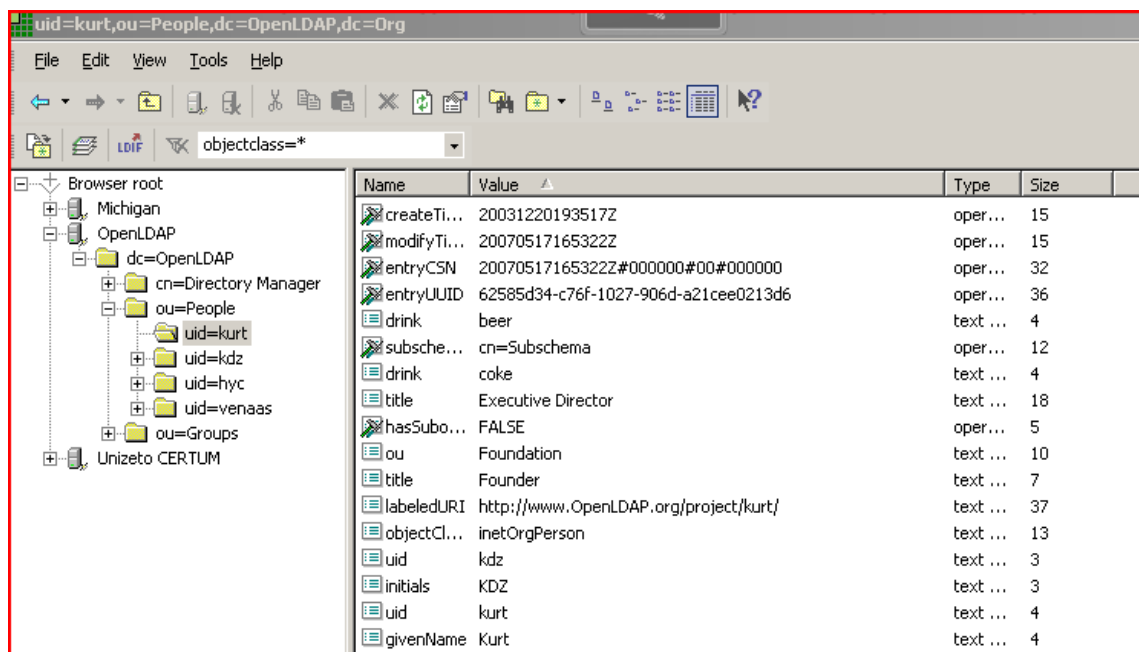
Remerciements

Merci à Baptiste Wicht pour sa relecture attentive.

Installation d'un browser Ldap

Afin de pouvoir consulter le contenu du serveur Ldap, nous avons besoin d'un browser Ldap. Je vous propose d'utiliser Softera Ldap Browser.

- Télécharger et installer Softera Ldap Browser 2.6 sur le site de Softerra : <http://www.ldapadministrator.com/download.htm>. et en faire une installation complète.
- Utiliser le browser pour aller sur le serveur préenregistré de openLdap : www.openldap.com sur le port 389.
- Ouvrir successivement les Nœuds :OpenLdap / cn = DirectoryManager / ou=People .Sélectionner le Nœud avec le uid = Kurt.



The screenshot shows the Softera Ldap Browser interface. The left pane displays a tree view of the LDAP hierarchy, with the entry 'uid=kurt' selected under 'ou=People'. The right pane shows a table of attributes for this entry.

Name	Value	Type	Size
createTi...	20031220193517Z	oper...	15
modifyTi...	20070517165322Z	oper...	15
entryCSN	20070517165322Z#000000#00#000000	oper...	32
entryUUID	62585d34-c76f-1027-906d-a21cee0213d6	oper...	36
drink	beer	text ...	4
subsche...	cn=Subschema	oper...	12
drink	coke	text ...	4
title	Executive Director	text ...	18
hasSubo...	FALSE	oper...	5
ou	Foundation	text ...	10
title	Founder	text ...	7
labeledURI	http://www.OpenLDAP.org/project/kurt/	text ...	37
objectCl...	inetOrgPerson	text ...	13
uid	kdz	text ...	3
initials	KDZ	text ...	3
uid	kurt	text ...	4
givenName	Kurt	text ...	4

Un clic droit dans les propriétés de Kurt permet de voir l'adresse complete de Kurt sur le serveur :

ldap://www.openldap.com:389/uid=kurt,ou=People,dc=OpenLDAP,dc=Org

Cette adresse se compose de 2 parties :

Partie 1 : www.openldap.com:389 est l'adresse internet et le port du serveur ldap.

Partie 2 : **uid=kurt,ou=People,dc=OpenLDAP,dc=Org** est le Distinguish Name de l'enregistrement kurt. Il représente le chemin d'accès dans l'arbre ldap à kurt.

Maintenant que nous avons vu comment obtenir les informations sur Kurt via un Browser Ldap, nous allons faire la même chose avec du Java !

Téléchargement de Spring Ldap

Télécharger la bibliothèque **spring-ldap-bin-with-dependencies-1.2.1.zip** sur le site <http://www.springframework.org/ldap>

Décompresser cette archive.

Dans le répertoire **dist** se trouve **spring-ldap-1.2.1.jar** qui doit être placé dans le CLASSPATH du projet pour pouvoir faire fonctionner ce tutorial.

Dans le repertoire **lib** se trouve **commons-lang.jar,commons-loggin.jar,spring-beans.jar,spring-core.jar**, toutes ces bibliothèques doivent aussi être ajoutées au CLASSPATH.

Création d'une connexion avec le serveur ldap.

Spring Ldap utilise la notion de ContextSource pour définir une connection au serveur Ldap.

Créons une factory pour récupérer un ContextSource correct. ContextSource étant une interface, nous utiliserons LdapContextSource qui est une implémentation fournie par Spring Ldap

```
public class LdapContextSourceFactory {  
  
    public static ContextSource getLdapContextSource() throws Exception {  
        LdapContextSource ldapContextSource = new LdapContextSource();  
        ldapContextSource.setUrl("ldap://www.openldap.com:389");  
        ldapContextSource.setBase("dc=OpenLDAP,dc=org");  
        ldapContextSource.afterPropertiesSet();  
        return ldapContextSource;  
    }  
}
```

Création de l'Objet Person

Spring Ldap permet de lire des enregistrements de personnes sous forme d'objet Person.

Voici notre class Person qui nous permettra d'obtenir une instance de Kurt. Uid est l'identifiant unique définit sur le serveur ldap pour les Person.

```
package com.neoneto.demo.springLdap1_2;
```

```

public class Person {

    private String uid;
    private String firstName;
    private String lastName;
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getUid() {
        return uid;
    }
    public void setUid(String uid) {
        this.uid = uid;
    }
}

```

Récupération d'un objet Person via un DAO.

Il est possible d'utiliser le pattern DAO pour accéder aux objets Person. Voici comment procéder en 4 étapes

- 1 Créer un Mapper de Ldap vers Java

Un `AttributMapper` permet d'instancier correctement notre classe Person. C'est lui qui effectue le mappage entre la Structure Ldap Person et le model Objet Java Person.

Par exemple l'attribut Ldap «givenName» sera placé dans l'attribut `Person.firstName` de l'objet Java : `p.setFirstName(attrs.get("givenName").get().toString());`

Ce mapper est déclaré comme interne interne du DAO, car il doit pas être utilisé ailleurs que dans cette Dao.

```

public class PersonDao {

    private static class PersonAttributMapper implements AttributMapper {

        public Person mapFromAttributes(Attributes attrs)
            throws javax.naming.NamingException {
            Person p = new Person();
            p.setFirstName(attrs.get("givenName").get().toString());
            p.setLastName(attrs.get("sn").get().toString());
            p.setUid(attrs.get("uid").get().toString());
            p.setEmail(attrs.get("mail").get().toString());
            return p;
        }
    }
}

```

- 2 Utiliser un LdapTemplate

L'exécution de requêtes Ldap se fait via un LdapTemplate.

```
public class PersonDao {
....
    private LdapTemplate ldapTemplate;

    public void setLdapTemplate(LdapTemplate ldapTemplate) {
        this.ldapTemplate = ldapTemplate;
    }
....
}
```

- 3 Définir le Distinguish Name

Le Distinguish Name est le chemin d'accès à l'enregistrement de Person sur le serveur. La fonction BuildDn permet d'obtenir un Distinguish Name correcte.

```
public class PersonDao {
....
    private Name buildDn(String uid) {
        DistinguishedName dn = new DistinguishedName();
        dn.add("ou", "People");
        dn.add("uid", uid);
        return dn;
    }
....
}
```

- 4 Obtenir l'instance de la class Person

Il suffit d'utiliser le methode lookup du ldapTemplate pour obtenir l'instance correspondant à la clef primaire, c'est à dire l'uid dans le cas des Objet Person.

```
public class PersonDao {
....
    public Person findByPrimaryKey(String uid) {
        Name dn = buildDn(uid);
        return (Person) ldapTemplate.lookup(dn, new
PersonAttributMapper());
    }
....
}
```

Test de la DAO

- Créons une classe exécutable pour tester notre DAO.

```
public class TestLdap {

    /** Retrieve a Kurt Person from ldap server and display Kurt in
Standard Out */
    public static void main(String[] args) {
    }
}
```

Cette classe exécutable ne prend en compte aucun argument.

- Ajoutons dans la méthode main le Context Source

```
// 1 Retrieve a LdapContextSource
ContextSource ldapContextSource = null;
```

```

    try {
        ldapContextSource =
LdapContextSourceFactory.getLdapContextSource();
    } catch (Exception e) {
        System.out.println("Impossible to get a
LdapContextSource.");
        e.printStackTrace();
    }

```

- Il faut aussi obtenir un LdapTemplate.

```

// 2 Instanciate a LdapTemplate
LdapTemplate ldapTemplate = new LdapTemplate();
ldapTemplate.setContextSource(ldapContextSource);

```

- Instancions un Dao et fournissons lui le Ldap Template.

```

// 3 instanciate a PersonDao
PersonDao dao = new PersonDao();
dao.setLdapTemplate(ldapTemplate);

```

- La récupération de Kurt se fait simplement en utilisant la DAO.

```

// 4 retrieve a Person and display it
Person person = dao.findByPrimaryKey("kurt");
System.out.println("Uid: " + person.getUid());
System.out.println("FirstName: " + person.getFirstName());
System.out.println("LastName: " + person.getLastName());
System.out.println("Email: " + person.getEmail() + "\n");

```

Lorsque nous exécutons cette classe nous obtenons dans la sortie standard :

```

Uid: kurt
LastName: Kurt
LastName: Zeilenga
Email: kurt@OpenLDAP.org

```

Ce qui correspond bien aux enregistrements contenue dans le serveur Ldap.

Amélioration de la DAO

Ajoutons une méthode dans la DAO qui permet de récupérer une liste de personne à partir de leur nom de famille.

La classe LikeFilter permet d'utiliser des filtres de recherches avec des WildCard :

```

public List getPersonNamesByLastName(String lastName) {
    AndFilter filter = new AndFilter();
    filter.and(new EqualsFilter("objectclass", "person"));
    filter.and(new LikeFilter("sn", lastName));
    return ldapTemplate.search("", filter.encode(),
        new PersonAttributMapper());
}

```

Par exemple getPersonNamesByLastName("E*"); retournera la liste des Personnes dont le nom de famille commence par E.

Complétons la classe main avec un exemple d'utilisation de cette nouvelle fonctionnalité.

```
// 5 retrieve a list of person
List listPerson = dao.getPersonNamesByLastName ("*e*");
for (Object object : listPerson) {
    System.out.println("Person: " + object);
}
```

L'exécution de la classe main retourne 2 personnes dans la sorties Standard.

Conclusion

Spring Ldap est une librairie J2EE qui permet de récupérer simplement des enregistrements Ldap sous forme d'objets java et d'affranchir le développeur Java de la connaissance des mécanismes de Ldap.

Spring Ldap permet aussi d'écrire sur le serveur Ldap. Reportez-vous à la documentation de référence de Spring Ldap qui explique très bien comment faire cela.

Liens et références

La doc de Spring Ldap : <http://static.springframework.org/spring-ldap/docs/1.1.2/reference/>

L'api de Spring Ldap : <http://static.springframework.org/spring-ldap/docs/1.1.2/api/>

En savoir plus sur Ldap : <http://www-sop.inria.fr/semir/personnel/Laurent.Mirtain/ldap-livre.html>

Liste de serveurs Ldap public : <http://www.emailman.com/ldap/public.html>

Blog de l'auteur : <http://getj2ee.over-blog.com/>

Contraintes techniques

Doc Version 1.0.4 - Publié le 19 Janvier 2008 par David Gimelle, Revue le 3 Avril 2008.

Ce tutorial a été testé avec Windows Vista, Eclipse 3.2 et le jdk 1.5.0_06 le 17 Janvier 2008.

Les Sources de ce tutorial :

1- Person.java :

```
package com.neoneto.demo.springLdap1_2;
```

```
public class Person {

    private String uid;
    private String firstName;
    private String lastName;
    private String email;

    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getFirstName() {
        return firstName;
    }
}
```

```

    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getUid() {
        return uid;
    }
    public void setUid(String uid) {
        this.uid = uid;
    }
    public String toString(){
        return uid+" - "+firstName+" "+lastName;
    }
}

```

2- PersonDao.java :

```

package com.neoneto.demo.springLdap1_2;

import java.util.List;

import javax.naming.Name;
import javax.naming.directory.Attributes;

import org.springframework.ldap.core.AttributesMapper;
import org.springframework.ldap.core.DistinguishedName;
import org.springframework.ldap.core.LdapTemplate;
import org.springframework.ldap.filter.AndFilter;
import org.springframework.ldap.filter.EqualsFilter;
import org.springframework.ldap.filter.LikeFilter;

public class PersonDao {

    private LdapTemplate ldapTemplate;

    private static class PersonAttributMapper implements AttributesMapper
    {

        public Person mapFromAttributes(Attributes attrs)
            throws javax.naming.NamingException {
            Person p = new Person();
            p.setFirstName(attrs.get("givenName").get().toString());
            p.setLastName(attrs.get("sn").get().toString());
            p.setUid(attrs.get("uid").get().toString());
            p.setEmail(attrs.get("mail").get().toString());
            return p;
        }
    }

    public Person findByPrimaryKey(String uid) {
        Name dn = buildDn(uid);
        return (Person) ldapTemplate.lookup(dn, new
        PersonAttributMapper());
    }

    private Name buildDn(String uid) {

```

```

        DistinguishedName dn = new DistinguishedName();
        dn.add("ou", "People");
        dn.add("uid", uid);
        return dn;
    }

    public void setLdapTemplate(LdapTemplate ldapTemplate) {
        this.ldapTemplate = ldapTemplate;
    }

    public List getPersonNamesByLastName(String lastName) {
        AndFilter filter = new AndFilter();
        filter.and(new EqualsFilter("objectclass", "person"));
        filter.and(new LikeFilter("sn", lastName));
        return ldapTemplate.search("", filter.encode(),
            new PersonAttributMapper());
    }
}

```

3 – LdapContextSourceFactory.java :

```

package com.neoneto.demo.springLdap1_2;

import org.springframework.ldap.core.ContextSource;
import org.springframework.ldap.core.support.LdapContextSource;

public class LdapContextSourceFactory {

    public static ContextSource getLdapContextSource() throws Exception {
        LdapContextSource ldapContextSource = new LdapContextSource();
        ldapContextSource.setUrl("ldap://www.openldap.com:389");
        ldapContextSource.setBase("dc=OpenLDAP,dc=org");
        ldapContextSource.afterPropertiesSet();
        return ldapContextSource;
    }
}

```

4 – TestLdap.java :

```

package com.neoneto.demo.springLdap1_2;

import java.util.List;

import org.springframework.ldap.core.ContextSource;
import org.springframework.ldap.core.LdapTemplate;

public class TestLdap {

    /** Retrieve a Kurt Person from ldap server and display Kurt in
    Standard Out */
    public static void main(String[] args) {

        // 1 Retrieve a LdapContextSource
        ContextSource ldapContextSource = null;
        try {
            ldapContextSource =
LdapContextSourceFactory.getLdapContextSource();
        } catch (Exception e) {
            System.out.println("Impossible to get a
LdapContextSource.");
            e.printStackTrace();
        }
    }
}

```

```

// 2 Instanciate a LdapTemplate
LdapTemplate ldapTemplate = new LdapTemplate();
ldapTemplate.setContextSource(ldapContextSource);

// 3 instanciate a PersonDao
PersonDao dao = new PersonDao();
dao.setLdapTemplate(ldapTemplate);

// 4 retrieve a Person and display it
Person person = dao.findByPrimaryKey("kurt");
System.out.println("Uid: " + person.getUid());
System.out.println("FirstName: " + person.getFirstName());
System.out.println("LastName: " + person.getLastName());
System.out.println("Email: " + person.getEmail() + "\n");

// 5 retrieve a list of person
List listPerson = dao.getPersonNamesByLastName("*e*");
for (Object object : listPerson) {
    System.out.println("Person: " + object);
}
}
}

```