Chapter 1

Using LDAP

1.1 Présentation du protocole LDAP

Le terme LDAP (Lightweight Directory Access Protocol) désigne un protocole permettant l'accès en lecture et en écriture à des annuaires d'entreprises. Ces bases de données, optimisées pour la lecture, sont des référentiels d'informations sur les composants d'une organisation (individus, ressources, organisation fonctionnelle). Le plus souvent, les annuaires LDAP sont exploités pour l'identification des utilisateurs aux services de messagerie ou aux applications intranet. Dans le cadre du développement d'applications e-business, les annuaires LDAP sont des composants techniques essentiels et critiques : il est le plus souvent obligatoire d'interfacer un logiciel avec ces annuaires.

Pharo dispose de différents frameworks pour le développement web (Seaside, AIDAweb, etc.) ainsi que pour l'accès aux bases de données (DBXTalk). Quelle est la situation avec LDAP? Peut on créer des applications capables d'identifier un utilisateur ou de récupérer des informations dans un annuaire LDAP?

Installation de LDAPlayer

Pour communiquer avec un annuaire LDAP, Pharo dispose de la bibliothèque de classes LDAPlayer créée par Ragnar Hojland Espinosa. Elle disponible via Squeaksource à l'adresse suivante:http://www.squeaksource.org/LDAPlayer/

Pour l'installer, lancez Pharo et définissez un nouveau dépôt HTTP à l'aide de Monticello. MCHttpRepository location: 'http://www.squeaksource.com/LDAPlayer' user: '' password: ''

1.2 Les principales classes

La plupart des classes que nous allons découvrir sont stockées dans le paquet LDAP-Core. Une connexion à un annuaire LDAP instancie un objet LDAPConnection. Des requŘtes vers l'annuaire retourne des objets LDAPResult. La classe LDAPAttrModifier fournie plusieurs méthodes de classe permettant de modifier les attributs d'une entrée de l'annuaire. La classe LDAPFilter a pour finalité de définir des filtres de recherche et de limiter ainsi le volume d'informations retourné au client.

Définir une connexion. Créons une nouvelle catégorie nommée 'AppLDAP' pour découvrir les fonctionnalités de LDAPlayer. Notre objectif est de mettre en situation les principales fonctionnalités de la bibliothèque LDAPlayer et de vous permettre de les réutiliser facilement dans une véritable application. Cette catégorie contient une classe unique nommée 'Ldap' qui renfermera l'ensemble de nos méthodes d'instances.

Pour définir la connexion, nous allons utiliser cinq variables d'instance qui représentent le nom du serveur LDAP (hostname), le port TCP de connexion (port), le compte utilisateur (bindDN), le mot de passe de l'utilisateur (password) et la base de la recherche (baseDN) qui sera utilisée pour construire un DN (Distinguished Name).

```
Object subclass: #Ldap
   instanceVariableNames: 'baseDN bindDN hostname password port'
   classVariableNames: ''
   poolDictionaries: ''
   category: 'Ldap'
```

Stéf ►*I do not like DN*◄ Stéf ►*We should explain that basically everything resolves around the call.* ◄

```
Ldap>>baseDN
   ↑ baseDN
Ldap>>baseDN: anObject
   baseDN := anObject
Ldap>>bindDN
   ↑bindDN
Ldap>>bindDN: anObject
   bindDN := anObject
Ldap>>hostname
   ↑hostname
Ldap>>hostname: anObject
   hostname := anObject
Ldap>>password
   ↑password
Ldap>>password: anObject
```

```
    password := anObject
Ldap>>port
    ↑ port
Ldap>>port: anObject
    port := anObject
```

Pour initialiser ces variables d'instances, créons une méthode initialize dans le protocole initalize-release.

```
Ldap>>initialize
    super initialize.
    self hostname: 'ldap.mondomaine.org'.
    self port: 389.
    self bindDN: 'cn=admin,dc=domaine,dc=org'. self password: 'mysecretpassword'. self
        baseDN: 'ou=people,dc=domaine,dc=org'.
```

Nous avons également besoin d'une variable d'instance permettant de sauvegarder un objet décrivant la connexion vers l'annuaire LDAP.

```
Object subclass: #Ldap instanceVariableNames: 'connection baseDN bindDN hostname
        password port' classVariableNames: '' poolDictionaries: '' category: 'Ldap'
```

Bien évidemment, nous avons besoin de deux accesseurs pour écrire et lire dans cette variable d'instance. Créons un protocole 'accessing' et définissons les deux méthodes suivantes :

```
Ldap>>connection
    ↑ connection
Ldap>>connection: anObject
    connection := anObject
```