3.5.2 Accès à l'annuaire LDAP en écriture

Bon pour relecture

Sommaire :

- Utilisation basique
- Utilisation du service LDAP depuis du code Java
  - Exceptions
  - Manipulation en écriture d'un utilisateur

Nous ne détaillons dans ce document que la manipulation en écriture des utilisateurs *LDAP*, qui est la seule manipulation en écriture implémenté actuellement dans esup-commons.

Un exemple de manipulation en écriture des utilisateur LDAP est présent dans le code de l'application esup-activaccount.

Utilisation basique

L'implémentation utilisée pour manipuler seulement les utilisateurs *LDAP* est **WriteableLdapUserServiceImpl**.

La déclaration d'un *bean* **writeableUserService** de cette classe ressemblera à :

```
<bean id="writeableLdapUserService"
        class="org.esupportail.activ.services.ldap.WriteableLdapUserServiceImpl" >
    <description>
        This bean provides LDAP write facilities to all the other beans.
        It must implement interface
        org.esupportail.commons.services.ldap.WriteableLdapUserService.
        This service is used to write in LDAP people branch dn.
    </description>
    <property name="ldapTemplate" ref="writeableLdapTemplate" >
        <description>
            The LDAP template used to access the LDAP directory.
            May not be set if you want to bind LDAP directory with user account.
        </description>
    </property>
    <property name="contextSource" ref="writeableContextSource">
        <description>
            The data source used by the LDAP template. Must be set if ldapTemplate
            is empty
        </description>
    </property>
    <property name="dnAuth" value="ou=people,dc=domain,dc=edu">
        <description>The DN path to use to connect user</description>
    </property>
    <property name="idAuth" value="uid">
        <description>The LDAP attribute that stores the unique identifier of
            users binding to the LDAP directory</description>
    </property>
    <property name="dnSubPath" value="ou=people">
        <description>The DN sub path. Used to create or delete entries</description>
    </property>
    <property name="idAttribute" value="uid">
        <description>
            The LDAP attribute that stores the unique identifier of
            users in the LDAP directory (optional, "uid" by
            default).
        </description>
    </property>
    <property name="attributes">
        <description>
            The LDAP attributes to update.
        </description>
        <list>
            <value>cn</value>
            <value>password</value>
            <value>shadowLastChange</value>
        </list>
    </property>
</bean>
```

- La propriété **dnAuth** donne le dn complet de la branche contenant les comptes amenés à se connecter.
- La propriété **idAuth** donne le nom de l'attribut *LDAP* qui contient l'identifiant unique des comptes amenés à se connecter.
- La propriété **dnSubPath** est le dn relatif de la branche contenant les utilisateurs.
- La propriété uidAttribute donne le nom de l'attribut LDAP qui contient l'identifiant unique des utilisateurs de l'annuaire.
- La propriété **attribute** donne les noms des attributs utilisateurs qui pourront potentiellement être modifiés.

> ⚠ Différencier les comptes amenés à se connecter des comptes utilisateurs permet par exemple de modifier des utilisateurs de la branche **people** avec un compte d'administration situé dans un autre branche.

Cette classe s'appuie, comme les service LDAP de recherche, sur la bibliothèque *LdapTemplate* :

```
<bean id="writeableLdapTemplate" class="org.springframework.ldap.LdapTemplate" >
    <description>
        The LDAP template used to acces the LDAP directory. See
        http://ldaptemplate.sourceforge.net.
    </description>
    <property name="contextSource" ref="writeableContextSource">
        <description>
            The data source used by the LDAP template.
        </description>
    </property>
</bean>

<bean id="writeableContextSource"
        class="org.springframework.ldap.support.LdapContextSource" scope="session">
    <description>
        This bean describes the physical acces to the LDAP
        directory. In the example below, anonymous and unsecured
        connections will be done to the LDAP directory running on
        server ldap.esup-portail.org on port 389, using the search
        base ou=people,dc=esup-portail,dc=edu, with scope "sub". See
        http://ldaptemplate.sourceforge.net.
    </description>
    <property name="url" value="ldap.domain.edu">
        <description>The LDAP URL of the directory.</description>
    </property>
    <property name="userName" value="">
        <description>
            The dn used to bind to the LDAP directory.
        </description>
    </property>
    <property name="password" value="">
        <description>
            The password used to bind to the LDAP directory.
        </description>
    </property>
    <property name="base" value="dc=domain,dc=edu">
        <description>The search base (mandatory).</description>
    </property>
    <!-- Pooled connections are initialized at the first LDAP r/w operation
                      and works despite credentials change  -->
    <property name="pooled" value="false" />
    <property name="baseEnvironmentProperties">
        <description>
            The environment properties, for instance to set the
            timeout.
        </description>
        <map>
            <entry key="com.sun.jndi.ldap.connect.timeout"
                value="5000" />
            <entry key="com.sun.jndi.ldap.connect.pool.debug" value="fine"/>
        </map>
    </property>
    <aop:scoped-proxy/>
</bean>
```

Il est préconisé d'utiliser le service LDAP d'écriture conjointement avec un service LDAP de recherche.
Le service de recherche permettra en effet de rechercher les comptes utilisateurs à modifier avec le service LDAP d'écriture.
Il faut donc instancier les deux dans votre contexte Spring, le service LDAP de recherche s'initialisant comme précédemment.

Il n'est pas nécessaire d'initialiser les variables **userName** et **password** lors de l'instanciation de l'objet **writeableContextSource**. En effet ces variables pourront être ininitailisées lors de l'exécution de l'applicationaprès avoir été, par exemple, saisis dans un formulaire présenté à l'utilisateur.

Ici **writeableLdapTemplate** est un singleton alors que **writeableContextSource** est instancié pour chaque session utilisateur (pour permettre par exemple à chaque utilisateur de modifier ses propres attributs avec son propre compte). Ceci est possible grâce à la balise :

```
<aop:scoped-proxy/>
```

Il est également nécessaire de définir un objet *LdapSchema* qui assurera une correspondance entre le nom des attributs LDAP de l'annuaire et les variables Java correspondantes. Cela est utile pour gérer notamment les nom d'attributs non normalisés par Supann ou Internet 2 :

```
<bean id="ldapSchema" class="org.esupportail.activ.services.ldap.LdapSchema" >
    <property name="displayName"><value>displayName</value></property>
    <property name="birthdate"><value>myBirthDay</value></property>
    <property name="birthdateFormat"><value>yyyyMMddHHmmss'Z'</value></property>
    <property name="uid"><value>uid</value></property>
    <property name="employeeId"><value>supannEmpId</value></property>
    <property name="cn"><value>cn</value></property>
    <property name="birthName"><value>myBirthName</value></property>
    <property name="password"><value>password</value></property>
    <property name="mail"><value>mail</value></property>
    <property name="shadowLastChange"><value>shadowLastChange</value></property>
</bean>
```

Utilisation du service LDAP depuis du code Java

Exceptions

Les exceptions lancées par les appels aux méthodes de **WriteableLdapUserService** peuvent lancer les exceptions suivantes :

- **LdapBindFailedException**, si les authentifiant du compte tentant de se connecter à *LDAP* sont incorrects,
- **LdapAttributesModificationException** si la tentative de modification des attributs utilisateurs échoue pour une autre raison.
  Une autre exception est mise à disposition des développeurs :
- **NotUniqueLdapAccountException**, si le compte utilisateur que l'on tente de modifier n'est pas unique.
  Il appartient au programmeur d'attraper ou non ces exceptions en fonction du contexte de l'application.

Manipulation en écriture d'un utilisateur

Pour modifier un utilisateur *LDAP* à l'aide d'un compte *LDAP* , il faut appeler le service *LDAP* de la manière suivante :

```
writeableLdapUserService.defineAuthenticatedContext("adminDN", "adminPassword");
```

Ainsi on va se connecter à l'annuaire avec les identifiants : **adminDN/adminPassword**. Ces identifiants peuvent provenir d'un formulaire de l'applcation créé avec JSF.

```
/* On recherche le compte à modifier dans l'annuaire LDAP */
LdapUser ldapUser = ldapUserService.getLdapUser("fjammes");

/* On efface les attributs que l'on ne souhaite pas réécrire */
ldapUser.getAttributes().clear();

List<String> listPasswordAttr = new ArrayList<String>();
listPasswordAttr.add(account.getPassword());
ldapUser.getAttributes().put(LdapSchema.getPassword(), listPasswordAttr);

/* Writing of shadowLastChange in LDAP */
List<String> listShadowLastChangeAttr = new ArrayList<String>();
Calendar cal = Calendar.getInstance();
String shadowLastChange = Integer.toString(
        (int) Math.floor(cal.getTimeInMillis()
        / (1000 * 3600 * 24)));
if (logger.isDebugEnabled()) {
        logger.debug("Writing shadowLastChange in LDAP: " + shadowLastChange);
}

listShadowLastChangeAttr.add(shadowLastChange);
ldapUser.getAttributes().put(LdapSchema.getShadowLastChange(), listShadowLastChangeAttr);

/* Writing of displayName in LDAP */
List<String> listDisplayNameAttr = new ArrayList<String>();
listDisplayNameAttr.add(account.getDisplayName());
ldapUser.getAttributes().put(LdapSchema.getDisplayName(), listDisplayNameAttr);
this.writeableLdapUserService.updateLdapUser(ldapUser);

ldapUser.getAttributes().clear();

/* For security reasons, all passwords are erased */
account.setPassword(null);

logger.info("Activated account [" + account.getId() + "]");

this.writeableLdapUserService.defineAnonymousContext();
```