Grouper - Store SmartLdapGroupStore (esup 4)

Cette page est destinée aux intégrateurs.

Données utilisées

Dans les paragraphes suivants, la configuration est présentée pour un annuaire LDAP configuré comme suit :

- dc=example,dc=org
  - ou=personnes
    - uid=user1
    - uid=user2
    - ....
  - ou=groups
    - cn=groupe1
    - cn=groupe2
    - ...

Les groupes sont des groupes exportés par grouper. Le cn est donc de la forme : folder1:folder2:nom. Par exemple : cn = admin:appli:appli1

Les groupes ont pour objectClass eduMember. Les membres des groupes sont connus via l'attribut member.

```
dn: cn=admin:appli:appli1,ou=groups,dc=aquitaine,dc=fr
objectClass: eduMember
cn: admin:appli:appli1
member: uid=user1,ou=personnes,dc=example,dc=org
member: uid=user2,ou=personnes,dc=example,dc=org
member: cn=admin:appli:appli2,ou=groups,dc=aquitaine,dc=fr
```

Au sein d'une entrée de personnes, on connait les groupes dont la personne est membre en interrogeant l'attribut isMemberOf ou memberOf selon la configuration de l'annuaire.

Deux configuration de l'annuaire sont possibles :

Cas 1 : Le isMemberOf/memberOf contient le cn du groupe. (Configuration par défaut de grouper).

```
dn: uid=user1,ou=personnes,dc=example,dc=org
uid: user1
objectClass: ...
isMemberOf: admin:appli:appli1
```

Cette configuration est décrite dans le paragraphe Configuration du SmartLdapGroupStore avec un cn dans l'entrée de personne.

Cas 2: Le isMemberOf/memberOf contient le dn du groupe. (Configuration par défaut de l'overlay MemberOf d'openLDAP).

```
dn: uid=user1,ou=personnes,dc=example,dc=org
uid: user1
objectClass: ...
memberOf: cn=admin:appli:appli1,ou=groups,dc=aquitaine,dc=fr
```

Cette configuration est décrite dans le paragraphe Configuration du SmartLdapGroupStore avec un dn dans l'entrée de personne.

Prise en compte de l'attribut d’appartenance par le portail

- Le portail doit être en mesure d'analyser les appartenances aux groupes. Il faut ajouter l'attribut qui contient les groupes de l'utilisateurs (isMemberOf / memberOf) à la liste des attributs pris en compte.

Fichier uportal-war/src/main/resources/properties/contexts/personDirectoryContext.xml

```
<bean id="uPortalLdapAttributeSource" class="org.jasig.services.persondir.support.ldap.LdapPersonAttributeDao">
  <property name="contextSource" ref="defaultLdapContext" />
  <property name="queryAttributeMapping">
    <map>
      <entry key="username" value="${ldap.uidAttr}"/>
    </map>
  </property>

  <property name="resultAttributeMapping">
    <map>
      <entry key="eduPersonPrimaryAffiliation"> <value>eduPersonPrimaryAffiliation</value></entry>
      <entry key="eduPersonAffiliation"> <value>eduPersonAffiliation</value></entry>
      <entry key="cn"> <value>cn</value></entry>
      <entry key="description"> <value>description</value></entry>
      <entry key="displayName"> <value>displayName</value></entry>
      <entry key="facsimileTelephoneNumber"> <value>facsimileTelephoneNumber</value></entry>
      <entry key="givenName"> <value>givenName</value></entry>
      <entry key="mail"> <value>mail</value></entry>
      <entry key="postalAddress"> <value>postalAddress</value></entry>
      <entry key="sn"> <value>sn</value></entry>
      <entry key="telephoneNumber"> <value>telephoneNumber</value></entry>
      <entry key="${ldap.uidAttr}">
        <set>
          <value>${ldap.uidAttr}</value>
          <value>username</value>
          <value>user.login.id</value>
        </set>
      </entry>
      <entry key="supannCodeINE"> <value>supannCodeINE</value></entry>
      <entry key="supannEtuId"> <value>supannEtuId</value></entry>
      <entry key="supannEmpId"> <value>supannEmpId</value></entry>
      <entry key="eduPersonAffiliation"> <value>eduPersonAffiliation</value></entry>
      <entry key="supannaffectation"> <value>supannaffectation</value></entry>
      <entry key="objectclass"> <value>objectclass</value></entry>
      <entry key="supannorganisme"> <value>supannorganisme</value></entry>

      <!-- Add attribute to determine if user belongs to a group. -->
      <entry key="isMemberOf"> <value>isMemberOf</value></entry>
    </map>
  </property>
</bean>
```

Activation du SmartLdapGroupStore

- Activer le SmartLdapGroupStore dans uportal-war/src/main/resources/properties/groups/compositeGroupServices.xml en dé-commentant le service smartldap :

```
<service>
    <name>smartldap</name>
    <service_factory>org.jasig.portal.groups.ReferenceIndividualGroupServiceFactory</service_factory>
    <entity_store_factory>org.jasig.portal.groups.smartldap.SmartLdapEntityStore$Factory</entity_store_factory>
    <group_store_factory>org.jasig.portal.groups.smartldap.SmartLdapGroupStore$Factory</group_store_factory>
        <entity_searcher_factory>org.jasig.portal.groups.smartldap.SmartLdapEntitySearcher$Factory<
/entity_searcher_factory>
    <internally_managed>false</internally_managed>
    <caching_enabled>true</caching_enabled>
  </service>
```

Description de la configuration du SmartLdapGroupStore

La configuration du SmartLdapGroupStore s'effectue dans le fichier "uportal-war/src/main/resources/properties/groups/SmartLdapGroupStoreConfig.xml"

Les propriété à configurer sont les suivants :

- url : l'URL de connexion à l'annuaire LDAP.
- userDn : le login de connexion à l'annuaire LDAP.
- password : le mot de passe de connexion à l'annuaire LDAP.

- baseDn : la branche de l'annuaire où les groupes sont stockées.
- filter : le filtre utilisé pour séléctionner tous les groupes à mapper.

- childGroupKeyRegex : une expression régulière qui permet de déterminer si les membres d'un groupe sont eux-même des groupes. L'expression régulière doit matcher si le membre est un groupe.
- keyIndexMatchingGroup : cet index est utilisé conjointement avec le childGroupKeyRegex pour extraire la clé du groupe.

- memberOfAttributeName : le nom de l'attribut, dans une entrée de personne, qui référence les groupes auxquels la personne appartient.

- keyAttributeName : le nom de l'attribut, dans l'entrée de groupe, qui référence la clé du groupe. Si on souhaite utiliser le dn comme clé, cette propriété doit être commentée.
- groupNameAttributeName: le nom de l'attribut, dans l'entrée de groupe, qui référence le nom du groupe.
- membershipAttributeName : : le nom de l'attribut, dans l'entrée de groupe, qui référence les membres du groupe.

Le nom du groupe est utilisé pour l'affichage. La clé du groupe est utilisée pour appeller le groupe. La clé est utilisée pour déterminer si une personne fait partie d'un groupe (en comparant avec l'attribut LDAP définit sous la propriété memberOfAttributeName).

Configuration du SmartLdapGroupStore avec un cn dans l'entrée de personne

La configuration attendue est la suivante :

```
<!--
 | This bean is the ContextSource instance that will be used to connect to LDAP.
 +-->
<bean id="ldapContext" class="org.springframework.ldap.core.support.LdapContextSource">
    <property name="url" value="ldap://ldap.univ.fr:389"/>
    <property name="userDn" value="uid=root,dc=univ,dc=fr"/>
    <property name="password" value="XXXX"/>
</bean>

<!--
 | Period, in seconds, after which SmartLdap will drop and re-init the groups
 | tree.  A value of zero or less (negative) disables this feature.
 +-->
<bean id="groupsTreeRefreshIntervalSeconds" class="java.lang.Long">
    <constructor-arg><value>900</value></constructor-arg>
</bean>

<!--
 | BaseDn that will be passed to the search (not to the context).
 |
 | WARNING:  If you get an error like this...
 |   ...PartialResultException: [LDAP: error code 10...
 | it probably means your baseDn isn't correct!
 +-->
<bean id="baseDn" class="java.lang.String">
    <constructor-arg><value>ou=groups,dc=univ,dc=fr</value></constructor-arg>
</bean>

<!--
 | ESUP Parameter.
 | This parameter is used to extract the id path of the groups
 | from their dn.
 | The id path is caught from the nth group of the regex where n is the value
 | of keyIndexMatchingGroup.
 | (http://docs.oracle.com/javase/7/docs/api/java/util/regex/Matcher.html#group%28int%29)
-->
<bean id="childGroupKeyRegex" class="java.lang.String">
    <constructor-arg><value>cn=(.*),ou=groups,dc=univ,dc=fr</value></constructor-arg>
</bean>
<bean id="keyIndexMatchingGroup" class="java.lang.Integer">
    <constructor-arg><value>1</value></constructor-arg>
</bean>

<!--
 | NOTE:  The remaining examples in this file are configured correctly for
 | Active Directory servers.
 +-->
```

```
<!--
 | LDAP query string that will be passed to the search.
 +-->
<bean id="filter" class="java.lang.String">
    <constructor-arg><value>(objectClass=groupOfNames)</value></constructor-arg>
</bean>

<!--
 | These beans tell smartLdap whether to gather additional groups that are
 | members of groups returned by the first baseDn and filter, and where to
 | look if so.
 |
 |   - resolveMemberGroups=[true|false]
 |   - resolveDn={a different, broader baseDn than the one above}
 |
 | Here's how it works:  smartLdap will first collect all groups under the
 | baseDn specified above.  If 'resolveMemberGroups' is enabled, it will
 | also search for additional groups (found within the 'resolveDn' specified
 | here) that are members of groups in the first collection.
 +-->
<bean id="resolveMemberGroups" class="java.lang.Boolean">
    <constructor-arg><value>false</value></constructor-arg>
</bean>
<bean id="resolveDn" class="java.lang.String">
    <constructor-arg><value>changeme</value></constructor-arg>
</bean>

<!--
 | This bean identifies the name of the Person Attribute that
 | lists the SmartLdap groups each person is a member of.
 +-->
<bean id="memberOfAttributeName" class="java.lang.String">
    <constructor-arg><value>isMemberOf</value></constructor-arg>
</bean>

<!--
 | This bean identifies the org.springframework.ldap.core.AttributesMapper
 | implementation used in reading the groups records from LDAP.
 +-->
<bean id="contextMapper" class="org.jasig.portal.groups.smartldap.SimpleContextMapper">
    <!--
     | Name of the group attribute that tells you its key.
     | If ommited, will use the dn.
     +-->
    <property name="keyAttributeName">
        <value>cn</value>
    </property>
    <!--
     | Name of the group attribute that tells you its name.
     +-->
    <property name="groupNameAttributeName">
        <value>cn</value>
    </property>
    <!--
     | Name of the group attribute that lists its members.
     +-->
    <property name="membershipAttributeName">
        <value>member</value>
    </property>
</bean>
```

Le code utilise les attributs comme suit :

1. Recherche de tous les groupes
2. Pour chaque groupe
    a. Extraire l'attribut qui servira clé soit le cn (keyAttributeName)
    b. Extraire l'attribut qui servira de nom soit le cn (groupNameAttributeName)
    c. Extraire la liste des membres en interrogeant l'attribut member (membershipAttributeName)
    d. Pour chaque membre

i. Vérifier s'il s'agit d'un groupe en vérifiant que l'expression matche l'expression cn=(.*),ou=groups,dc=univ,dc=fr (childGroupKeyRegex)
ii. S'il s'agit bien d'un groupe, extraire la clé en utilisant le groupe dont l'index est 1 (keyIndexMatchingGroup). Cela renvoie donc le cn du groupe.
iii. Insérer le sous-groupe en tant qu'enfant du groupe

Configuration du SmartLdapGroupStore avec un dn dans l'entrée de personne

La configuration attendue est la suivante :

```
<!--
 | This bean is the ContextSource instance that will be used to connect to LDAP.
 +-->
<bean id="ldapContext" class="org.springframework.ldap.core.support.LdapContextSource">
    <property name="url" value="ldap://frmp0165.frml.bull.fr:389"/>
    <property name="userDn" value="uid=root,dc=univ,dc=fr"/>
    <property name="password" value="secret"/>
</bean>

<!--
 | Period, in seconds, after which SmartLdap will drop and re-init the groups
 | tree.  A value of zero or less (negative) disables this feature.
 +-->
<bean id="groupsTreeRefreshIntervalSeconds" class="java.lang.Long">
    <constructor-arg><value>900</value></constructor-arg>
</bean>

<!--
 | BaseDn that will be passed to the search (not to the context).
 |
 | WARNING:  If you get an error like this...
 |   ...PartialResultException: [LDAP: error code 10...
 | it probably means your baseDn isn't correct!
 +-->
<bean id="baseDn" class="java.lang.String">
    <constructor-arg><value>ou=groups,dc=univ,dc=fr</value></constructor-arg>
</bean>

<!--
 | ESUP Parameter.
 | This parameter is used to extract the id path of the groups
 | from their dn.
 | The id path is caught from the nth group of the regex where n is the value
 | of keyIndexMatchingGroup.
 | (http://docs.oracle.com/javase/7/docs/api/java/util/regex/Matcher.html#group%28int%29)
-->
<bean id="childGroupKeyRegex" class="java.lang.String">
    <constructor-arg><value>cn=(.*),ou=groups,dc=univ,dc=fr</value></constructor-arg>
</bean>
<bean id="keyIndexMatchingGroup" class="java.lang.Integer">
    <constructor-arg><value>0</value></constructor-arg>
</bean>

<!--
 | NOTE:  The remaining examples in this file are configured correctly for
 | Active Directory servers.
 +-->

<!--
 | LDAP query string that will be passed to the search.
 +-->
<bean id="filter" class="java.lang.String">
    <constructor-arg><value>(objectClass=groupOfNames)</value></constructor-arg>
</bean>

<!--
 | These beans tell smartLdap whether to gather additional groups that are
 | members of groups returned by the first baseDn and filter, and where to
```

```
 | look if so.
 |
 |   - resolveMemberGroups=[true|false]
 |   - resolveDn={a different, broader baseDn than the one above}
 |
 | Here's how it works:  smartLdap will first collect all groups under the
 | baseDn specified above.  If 'resolveMemberGroups' is enabled, it will
 | also search for additional groups (found within the 'resolveDn' specified
 | here) that are members of groups in the first collection.
 +-->
<bean id="resolveMemberGroups" class="java.lang.Boolean">
    <constructor-arg><value>false</value></constructor-arg>
</bean>
<bean id="resolveDn" class="java.lang.String">
    <constructor-arg><value>changeme</value></constructor-arg>
</bean>

<!--
 | This bean identifies the name of the Person Attribute that
 | lists the SmartLdap groups each person is a member of.
 +-->
<bean id="memberOfAttributeName" class="java.lang.String">
    <constructor-arg><value>isMemberOf</value></constructor-arg>
</bean>

<!--
 | This bean identifies the org.springframework.ldap.core.AttributesMapper
 | implementation used in reading the groups records from LDAP.
 +-->
<bean id="contextMapper" class="org.jasig.portal.groups.smartldap.SimpleContextMapper">
    <!--
     | Name of the group attribute that tells you its key.
     | If ommited, will use the dn.
     +-->
            <!--
    <property name="keyAttributeName">
        <value>cn</value>
    </property>
            -->
    <!--
     | Name of the group attribute that tells you its name.
     +-->
    <property name="groupNameAttributeName">
        <value>cn</value>
    </property>
    <!--
     | Name of the group attribute that lists its members.
     +-->
    <property name="membershipAttributeName">
        <value>member</value>
    </property>
</bean>
```

Le code utilise les attributs comme suit :

1. Recherche de tous les groupes
2. Pour chaque groupe
    a. Extraire l'attribut qui servira clé soit le dn en l'absence de keyAttributeName
    b. Extraire l'attribut qui servira de nom soit le cn (groupNameAttributeName)
    c. Extraire la liste des membres soit les member (membershipAttributeName)
    d. Pour chaque membre
        i. Vérifier s'il s'agit d'un groupe en vérifiant que l'expression matche l'expression cn=(.*),ou=groups,dc=univ,dc=fr (childGroupKeyRegex)
        ii. S'il s'agit bien d'un groupe, extraire la clé en utilisant le groupe dont l'index est 0 (keyIndexMatchingGroup). Cela renvoie donc le dn du groupe.
        iii. Insérer le sous-groupe en tant qu'enfant du groupe

Prise en compte du SmartLdapGroupStore dans les groupes uPortal

- Ajouter les groupes smartLDAP à votre arborescence des groupes. Par exemple, on peut modifier le fichier Everyone.group-membership.xml comme suit :

```
<group script="classpath://org/jasig/portal/io/import-group_membership_v3-2.crn">
  <name>Everyone</name>
  <entity-type>org.jasig.portal.security.IPerson</entity-type>
  <creator>system</creator>
  <description>All Users</description>
  <children>
    <group>Authenticated Users</group>
    <group>PAGS Root</group>
    <group>Faculty</group>
    <group>Guests</group>
    <group>Portal System</group>
    <group>Staff</group>
    <group>Students</group>
    <!-- Ajout des groupes SmartLdap -->
    <group>SmartLdap ROOT</group>
  </children>
</group>
```

- L'import du fichier modifié s'effectue via la tâche d'import du portail. L'arborescence des groupes est alors construite. Dans les logs, on peut lire l'ensemble des groupes importés.

```
ant data-import -Dfile=uportal-war/src/main/data/quickstart_entities/group_membership/Everyone.group-membership.
xml
>>>
...
     [java]  INFO SmartLdap adding record for group:  esup:etablissements:etablissement_5:composantes:
composante_17:tous with key esup:etablissements:etablissement_5:composantes:composante_17:tous

     [java]  INFO SmartLdap adding record for group:  esup:etablissements:etablissement_5:composantes:
composante_18:tous with key esup:etablissements:etablissement_5:composantes:composante_18:tous

     [java]  INFO SmartLdap adding record for group:  esup:etablissements:etablissement_5:composantes:
composante_19:tous with key esup:etablissements:etablissement_5:composantes:composante_19:tous

     [java]  INFO SmartLdap adding record for group:  esup:etablissements:etablissement_5:composantes:
composante_20:tous with key esup:etablissements:etablissement_5:composantes:composante_20:tous
..
```

- On peut aussi vérifier les groupes importés via l'outil de gestion des groupes du portail.

Appel d'un groupe provenant du SmartLdapGroupStore

On cible un groupe en le désignant par son nom (et non par la clé). Par exemple, dans un fichier de channel, on écrira :

```
<group>admin:appli:appli1</group>
```

Il faudra ensuite réimporter le fichier via la commande d'import pour qu'il soit pris en compte par le portail.