LDAP

Contents

Als LDAP Server wird Active Directory verwendet.

Um die Security zu aktivieren, müssen folgende Änderungen an der *WEB-INF/web.xml* vorgenommen werden.

Der Security Filter muss hinzugefügt werden:

```
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Die korrekte Spring Konfiguration muss eingebunden werden:

```
Folgende Zeilen:

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>de.latlon.xplan.manager.web.spring.ManagerWebSpringConfig</param-value>
</context-param>

zu folgenden Zeilen abändern:

<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>de.latlon.xplan.manager.web.spring.ManagerWebSpringConfigWithAdLdapSecurity ↩
      </param-value>
</context-param>
```

Um die Security zu deaktivieren, müssen der oben genannte Filter entfernt und die zuerst beschriebene Spring Konfiguration eingebunden werden.

Die Verbindung zum Active Directory Server kann in der Webapp des XPlanManagers in der Datei *WEB-INF/classes/security.properties* konfiguriert werden:

```
ldap.server.domain=adserver.lat-lon
ldap.server.url=ldap://adserver.lat-lon:389
```

Weiterhin kann dort die Konfiguration für das dynamische Auslesen der Gruppenhierarchien erfolgen. Dies ist lediglich nötig, wenn dynamische Gruppenhierarchien genutzt werden (siehe weiter unten):

```
ldap.server.searchUser=ll-technical
ldap.server.searchPassword=ADServer!
ldap.server.searchNode=OU=lgvxplanisk,DC=adserver,DC=lat-lon
```

Hier muss ein (technischer) Nutzer konfiguriert werden, der lesenden Zugriff auf das Active Directory hat um die Rollen auslesen zu können (*ldap.server.searchUser* und *ldap.server.searchPassword*). Mit dem Schlüssel *ldap.server.searchNode* wird die Organisationseinheit mit OU, und die Domänen Komponenten mit DC in kommaseparierter Form notiert.

Die Active Directory-Gruppen mit den dazugehörigen Bezirken auf die die Gruppe zugreifen darf, die Superuser-Gruppen und die Editor-Gruppen können in der Webapp des XPlanManagers in der Datei *WEB-INF/classes/security-configuration.xml* konfiguriert werden:

```
<util:list id="groupsSuper" value-type="java.lang.String">
  <beans:value>SUPER</beans:value>
</util:list>

<util:list id="groupsEditor" value-type="java.lang.String">
  <beans:value>EDITOR</beans:value>
</util:list>
```

```
<util:map id="groupsTodistricts" key-type="java.lang.String" value-type="java.util.List">
  <beans:entry key="ALTONA" value-ref="districtsAltona" />
  <beans:entry key="HARBURG" value-ref="districtsHarburg" />
  <beans:entry key="HAMBURGNORD" value-ref="districtsHamburgNord" />
</util:map>

<util:list id="districtsAltona" value-type="java.lang.String">
  <beans:value>Altona</beans:value>
</util:list>

<util:list id="districtsHarburg" value-type="java.lang.String">
  <beans:value>Harburg</beans:value>
</util:list>

<util:list id="districtsHamburgNord" value-type="java.lang.String">
  <beans:value>Hamburg-Nord</beans:value>
</util:list>

<beans:bean id="grantedAuthoritiesMapper"
            class="de.latlon.xplan.manager.web.spring.security. ↩
               ActiveDirectoryGrantedAuthoritiesMapper">
  <beans:constructor-arg index="0">
    <beans:ref bean="groupsSuper" />
  </beans:constructor-arg>
  <beans:constructor-arg index="1">
    <beans:ref bean="groupsEditor" />
  </beans:constructor-arg>
  <beans:constructor-arg index="2">
    <beans:ref bean="groupsTodistricts" />
  </beans:constructor-arg>
  <beans:constructor-arg index="3">
    <beans:ref bean="roleHierarchy" />
  </beans:constructor-arg>
</beans:bean>
```

Details zur Konfiguration:

- Die Liste *groupsSuper* (momentan mit SUPER gefüllt) stellt eine Liste aller Superuser-Gruppen dar. Ein Nutzer, der einer Superuser-Gruppe zugeordnet ist, hat keinerlei Beschränkungen bei der Nutzung des XPlanManagers.
- Die Liste *groupsEditor* (momentan mit EDITOR gefüllt) stellt eine Liste aller Editor-Gruppen dar. Falls ein Nutzer einer Editor-Gruppe zugeordnet ist, kann dieser alle Pläne aus Bezirken editieren, für die der Nutzer Rechte hat (siehe nächste Zeile).
- Die Map *groupsTodistricts* (momentan mit ALTONA, HARBURG und HAMBURGNORD gefüllt) muss eine Active Directory-Gruppe als Key erhalten und als Value eine Liste aller Bezirke, auf welche die Gruppe zugreifen darf. Ein LDAP-Nutzer, der einer oder mehrerer dieser Gruppen zugeordnet ist, besitzt Rechte für die jeweils konfgurierten Bezirke.
- Die Listen der Bezirke *groupsTodistricts* (momentan mit Altona, Harburg und Hamburg-Nord gefüllt) stellen eigene Spring-Beans dar und werden von der zuvor beschriebenen Map referenziert.
- Der *grantedAuthoritiesMapper* nutzt die zuvor konfigurierten Rechte. Diese Bean darf im Normalfall nicht manipuliert werden! Sie sollte lediglich modifiziert werden, wenn Gruppenhierarchien deaktiviert werden sollen (siehe nächster Absatz).
- Achtung: In der Beispielkonfiguration wird eine Gruppenhierarchie genutzt. Der nächste Absatz muss zwingend beachtet werden.

Gruppenhierarchien

Es können Gruppenhierarchien konfiguriert werden, um hierarchische Abhängigkeiten zwischen Gruppen abzubilden. So kann eine Gruppen Mitglied einer anderen Gruppe sein und dabei die Eigenschaften der übergeordneten Gruppe übernehmen.

Beispiel: Gruppe "Hamburg" ist Mitglied der Gruppe "Editor". Dadurch hat die Gruppe "Hamburg" die Eigenschaften von "Hamburg" und "Editor". Die Gruppe "Editor" hat dagegen lediglich die Eigenschaften von "Editor".

Details zur Konfiguration:

- Das als viertes Konstruktorargument übergebene Argument der Bean *grantedAuthoritiesMapper* (siehe vorheriges Konfigurationsbeispiel) muss der Gruppenhierarchie entsprechen. Dies kann wie in Beispiel 1 in der Konfiguration direkt erfolgen (das Beispiel konfiguriert HARBURG als Mitglied der Gruppe EDITOR) oder von der Anwendung aus dem ActiveDirectory ausgelesen werden, wie in Beispiel 2 gezeigt.
- Sind keine Gruppenhierachien vorhanden, muss das vierte Konstruktorargument entfernt werden (dies sollte der einzige Fall sein, in dem der *grantedAuthoritiesMapper* manipuliert wird).
- Falls eine dynamische Rollenhierarchie wie in Beispiel 2 genutzt wird, müssen in der Datei *WEB-INF/classes/security.properties* zwingend der searchUser, das searchPassword und der searchNode angegeben werden (siehe weiter oben).

Beispiel 1 - Konfiguration einer statischen Rollenhierarchie:

```
<beans:bean id="roleHierarchy" class="org.springframework.security.access.hierarchicalroles ↩
    .RoleHierarchyImpl">
  <beans:property name="hierarchy">
   <beans:value>
     HARBURG > EDITOR
   </beans:value>
  </beans:property>
</beans:bean>
```

Beispiel 2 - Konfiguration einer dynamischen Rollenhierarchie:

```
<beans:bean id="roleHierarchy" class="org.springframework.security.access.hierarchicalroles ↩
    .RoleHierarchyImpl">
  <beans:property name="hierarchy">
    <beans:bean factory-bean="roleHierarchyScanner" factory-method="retrieveRoleHierarchy"  ↩
        />
  </beans:property>
</beans:bean>

<beans:bean id="roleHierarchyScanner"
            class="de.latlon.xplan.manager.web.spring.security. ↩
                ActiveDirectoryRoleHierarchyScanner">
  <beans:constructor-arg index="0" value="${ldap.server.url}" />
  <beans:constructor-arg index="1" value="${ldap.server.domain}" />
  <beans:constructor-arg index="2" value="${ldap.server.searchUser}" />
  <beans:constructor-arg index="3" value="${ldap.server.searchPassword}" />
  <beans:constructor-arg index="4" value="${ldap.server.searchNode}" />
  <beans:constructor-arg index="5">
    <beans:ref bean="groupsSuper" />
  </beans:constructor-arg>
  <beans:constructor-arg index="6">
    <beans:ref bean="groupsEditor" />
  </beans:constructor-arg>
  <beans:constructor-arg index="7">
    <beans:ref bean="groupsTodistricts" />
  </beans:constructor-arg>
</beans:bean>
```

Note
Sowohl die dynamische als auch die statische Gruppenhierarchie wird während des Starts der Webapp ausgewertet. Falls es nachträgliche Änderungen an den Hierarchien gibt, muss die Webapp neu gestartet werden, damit diese von der Software erkannt und genutzt werden.