3.14 Authentification

Esup-commons n'authentifie pas les utilisateurs en tant que tel. Il est en revanche capable, en s'appuyant sur un service d'authentification externe, de savoir quel est l'utilisateur connecté à l'application web en cours.

Plusieurs moyens sont possibles pour cela. *esup-commons* s'appuie sur un service d'authentification externe, le *bean* **authenticationService**, pour authentifier les utilisateurs. Ce *bean*, qui doit implémenter l'interface **AuthenticationService**, possède une méthode **getCurrentUserId()**, qui renvoie l'identifiant de l'utilisateur courant.Le *bean* **authenticationService** est défini dans **/properties/auth/auth.xml**. Plusieurs classes implémentant l'interface **AuthenticationService** sont disponibles dans le package **org.esupportail.commons.services.authentication**, elles sont décrites ci après.

Dans les applications d'exemples ESUP-Commons ainsi que dans les archétypes, en plus d'une implémentation de **AuthenticationService**, on va aussi trouvé un bean métier, nommé **authenticator**, auquel on va injecter l'**authenticationService** dont le but sera de transformer l'objet **AuthInfo** obtenu par l'authenticationService (méthode **getAuthInfo**) en un objet métier (**User**).

Ex : AuthenticatorImpl.java dans l'archétype EC2 0.2.8

Sommaire :

Modes d'authentification disponibles

Authentification CAS

La classe **CasFilterAuthenticator**, renvoie l'identifiant de l'utilisateur authentifié par le filtre J2EE CAS :

```
<bean
    id="authenticationService"
    class="org.esupportail.commons.services.authentication.CasFilterAuthenticationService" />
```

Pour que cela fonctionne, il faut bien sûr que le filtre *CAS* soit correctement configuré.

Portail (JSR-168)

Esup-commons V2 propose (en mode portlet) deux classes permettant d'authentifier l'utilisateur du portail dans lequel se trouve l'application (remplaçant ainsi la classe **PortalAuthenticator** proposée en V1 qui ne proposait que du CAS) : **CasifiedPortalAuthenticationService** et **ShibbolizedPortalAuthenticationService** à choisir en fonction de la méthode d'authentification utilisée par le portail
L'une comme l'autre renvoie l'identifiant de l'utilisateur connecté au portail, selon *JSR*-168 :
La propriété **uidPortalAttribute** est le nom de l'attribut du portail qui sera considéré comme l'identifiant de l'utilisateur. Il doit être déclaré dans le fichier **/webapp/WEB-INF/portlet.xml**. Par défaut il prendra "uid".

```
<portlet-app>
[...]
        <user-attribute>
                <name>uid</name>
        </user-attribute>
        <user-attribute>
                <name>displayName</name>
        </user-attribute>
</portlet-app>
```

Cette implémentation ne fonctionne qu'en mode *portlet*.

Portail cassifié

```
<bean
    id="authenticationService"
    class="org.esupportail.commons.services.authentication.CasifiedPortalAuthenticationService" >
  <property name="uidPortalAttribute" value="uid" />
</bean>
```

Portail shibbolisé

```
<bean
    id="authenticationService"
    class="org.esupportail.commons.services.authentication.ShibbolizedPortalAuthenticationService" >
  <property name="uidPortalAttribute" value="uid" />
</bean>
```

Authentification Shibboleth

La classe **ShibbolethApacheModuleAuthenticationService** propose une authentification shib.

```
<bean id="authenticationService" lazy-init="true"
               class="org.esupportail.commons.services.authentication.
ShibbolethApacheModuleAuthenticationService">
            <property name="idHeader" value=""/>
            <property name="attributeHeaders">
               <list>
                   <value></value>
               </list>
            </property>
       </bean>
```

L'authentification shibboleth est faite par un module apache qui fonctionne en amont de l'application Java. Par l'option "**ShibUseHeaders On**" le module transmet à l'application Java, sous forme d'entêtes HTTP, l'ensemble des attributs de utilisateur connecté (reçus depuis le fournisseur d'identités). Ce sont ces attributs qui sont utilisés par l'implémentation **ShibbolethApacheModuleAuthenticationService**. L'objet **AuthInfo** renvoyé par la méthode **getAuthInfo()** de **AuthenticationService** implémente 3 méthodes :

- **getId()** : Renvoie l'identifiant de l'utilisateur. C'est en fait l'entête http ayant pour nom la valeur de la propriété **idHeader** de **ShibbolethApacheModuleAuthenticationService**
- **getType()** : Renvoie **AuthUtils.SHIBBOLETH**
- **getAttributes()** : Renvoie une Map d'attributs correspondant à aux noms des entêtes http ciblées par la valeur de la propriété **attributeHeaders** de **ShibbolethApacheModuleAuthenticationService**

Authentification par le Remote User

La classe **RemoteUserAuthenticator** s'appuie sur la variable **REMOTE_USER** de HTTP :

```
<bean
    id="authenticationService"
    class="org.esupportail.commons.services.authentication.RemoteUserAuthenticator" />
```

L'utilisation de cette classe n'est pas recommandée.

Authentification hors connexion

La classe **OfflineFixedUserAuthenticator** renvoie toujours le même identifiant.

```
<bean
    id="authenticationService"
    class="org.esupportail.commons.services.authentication.OfflineFixedUserAuthenticator" >
  <property name="userId" value="paubry" />
</bean>
```

Elle peut être utilisée lorsque l'on travaille à la mise au point hors connexion (pratique dans le train).

Authentification Nulle

La classe **NullAuthenticationService** renvoie toujours null.

```
<bean
    id="authenticationService"
    class="org.esupportail.commons.services.authentication.NullAuthenticationService" >
</bean>
```

Elle peut être utilisée lorsque l'on travaille sur une application qui ne nécessite pas d'authentification ou en phase de développement.

Authentification Mixte

Esup-commons propose une implémentation **delegatingAuthenticationService** qui permet de chaîner plusieurs des services d'authentification cités précédemment.

```
<bean id="authenticator" lazy-init="true"
                class="org.esupportail.example.services.authentication.AuthenticatorImpl">
                <property name="authenticationService" ref="delegatingAuthenticationService" />
        </bean>

        <bean id="delegatingAuthenticationService" lazy-init="true"
                class="org.esupportail.commons.services.authentication.DelegatingAuthenticationService">
            <property name="authenticationServices">
               <list>
                   <ref bean="portletAuthenticationService" />
                   <ref bean="casFilterAuthenticationService" />
                   <ref bean="offlineFixedUserAuthenticationService" />
               </list>
            </property>
        </bean>

        <bean id="casFilterAuthenticationService" lazy-init="true"
                class="org.esupportail.commons.services.authentication.CasFilterAuthenticationService">
        </bean>

        <bean id="portletAuthenticationService" lazy-init="true"
                class="org.esupportail.commons.services.authentication.CasifiedPortalAuthenticationService">
                <property name="uidPortalAttribute" value="uid" />
        </bean>

        <bean id="offlineFixedUserAuthenticationService"
                class="org.esupportail.commons.services.authentication.OfflineFixedUserAuthenticationService">
                <property name="authId" value="guest" />
                <property name="authType" value="cas" />
        </bean>
```

A la connexion de l'utilisateur *esup-commons* tentera le premier service d'authentification indiqué, si celui-ci échoue (ne retourne pas d'utilisateur) il passera au suivant et ainsi de suite dans l'ordre des services paramétrés et ce jusqu'à ce qu'une authentification réussisse.

Dans *esup-commons v1* on avait la classe **PortalOrCasFilterAuthenticator** s'appuie d'abord sur *JSR*-168, puis sur le filtre *CAS* qui fonctionnait à la fois en mode *portlet* ou *servlet*) :

```
<bean
    id="authenticationService"
    class="org.esupportail.commons.authentication.PortalOrCasFilterAuthenticator" >
  <property name="uidPortalAttribute" value="uid" />
</bean>
```

Cette implémentation est désormais dépréciée au profit de la classe **delegatingAuthenticationService**.

Utilisation du service d'authentification

Le service d'authentification s'appelle de la manière suivante :

```
String uid = authenticationService.getCurrentUserId();
```

Si **uid** est **null**, alors l'utilisateur n'est pas authentifié.