Shibboleth IdP v3.2.1

Davide Vaghetti – davide.vaghetti@garr.it
Coordinatore Comitato Tecnico Scientifico Federazione IDEM - GARR

Data Sources multiple

Davide Vaghetti – davide.vaghetti@garr.it

Data Connector

- I data connector producono set di `IdPAttribute` (oggetti java) che sono consumati dal `Attribute resolver` per la definizione degli attributi.
- Distinguiamo tra:
  - definzione dei data source
    - ad es. in ldap.properties
    - come bean (di solito in global.xml)
  - uso dei data source per funzioni specialistiche
    - autenticazione (idp.authn...)
    - persistentId (idp.persistentId...)
    - storage general-purpose (sessione e consenso) (vedi global.xml)
  - definzione dei data connector
    - in `attribute-resolver.xml`
    - tipi:
      - LDAPConnector
      - RelationDatabaseConnector
      - altri... (vedi tutorial successivi)

Davide Vaghetti – davide.vaghetti@garr.it

RelationalDatabaseConnector

- Definibili interamente in `attribute-resolver.xml`:

```
<resolver:DataConnector id="myDatabase"
xsi:type="dc:RelationalDatabase">
  <resolver:FailoverDataConnector ref="BackupDataseConnector"/>
  <dc:ApplicationManagedConnection
     jdbcDriver="org.hsqldb.jdbc.JDBCDriver"
jdbcURL="jdbc:hsqldb:mem:RDBMSDataConnectorStore"
     jdbcUserName="SA" jdbcPassword="secret" />
  <dc:QueryTemplate>
     <![CDATA[
       SELECT * FROM people WHERE
userid='$resolutionContext.principal'
     ]]>
  </dc:QueryTemplate>

  <dc:Column columnName="homephone"
attributeID="phonenumber" />

  <dc:ResultCache elementTimeToLive="PT10S"/>
</resolver:DataConnector>
```


Davide Vaghetti – davide.vaghetti@garr.it

RelationalDatabaseConnector - 1

BeanManagedConnection: definibili richiamando un `bean` gia' definito:

```
<resolver:DataConnector id="myDatabase" xsi:type="dc:RelationalDatabase"
mappingStrategy="MappingBeanId">
  <dc:BeanManagedConnection>DataConnectorBeanId</dc:BeanManagedConnection>
  <dc:QueryTemplate>
  [...]
</resolver:DataConnector>
```

Ad esempio puo' essere riutilizzato il `bean` definito per il `persistentId` e lo storage general-purpose (sessione e consenso). Elementi:

```
  * <dc:QueryTemplate>: standard query SQL racchiusa tra <![CDATA[ ... ]]>;
    * `$resolutionContext.principal` = uid (authn...);
  * <dc:Column>
    * i campi (`columnName`) selezionati dalla query devono essere
associate ad un `attributeID`
    * l'attributo referenziato **deve** essere definito in `attribute-
resolver`
    * si puo' referenziare un attributo esistente o definirlo ex-novo
  * <dc:ResultCache>:
    * elementTimeToLive: permanenza in cache del campo (default 4H)
    * maximumCachedElements: numero massimo elementi in cache (default 500)
```


Davide Vaghetti – davide.vaghetti@garr.it

Riutilizzo di un data source

Abbiamo gia' un data source definito, riutilizziamolo!

Il nostro data source, da `global.xml`:

```
[...]
    <bean id="MyDataSource"
class="org.apache.commons.dbcp.BasicDataSource"
      p:driverClassName="com.mysql.jdbc.Driver"
[...]
```

Il bean id e' "MyDataSource", per richiamarlo in `attribute-resolver`:

```
[...]
<resolver:DataConnector id="myDatabase"
xsi:type="dc:RelationalDatabase" mappingStrategy="MappingBeanId">

<dc:BeanManagedConnection>DataConnectorBeanId</dc:BeanManagedConnec
tion>
```


Davide Vaghetti – davide.vaghetti@garr.it

RuoliOrganizzativi

Tabella `shibboleth.RuoliOrganizzativi`:

```
(
id MEDIUMINT NOT NULL,
uid VARCHAR(255) NOT NULL,
ruolo VARCHAR(255) NOT NULL,
PRIMARY KEY (id)
);
```

Contenuto:

```
-------------------------------
| id | uid   | ruolo           |
-------------------------------|
| 0  | mario | Tecnico         |
| 1  | pino  | Amministrativo  |
| 2  | pina  | Docente         |
-------------------------------
```


Davide Vaghetti – davide.vaghetti@garr.it

Configuriamo la query

In `attribute-resolver.xml`:

```
[...]

<dc:BeanManagedConnection>MyDataSource</dc:BeanManage
dConnection>
        <dc:QueryTemplate>
            <![CDATA[
                SELECT * FROM
shibboleth.RuoliOrganizzativi WHERE uid =
'$resolutionContext.principal'
            ]]>
        </dc:QueryTemplate> [...]
```

E associamo il campo ruolo all'attributo `myRuolo`:

```
        <dc:Column columnName="ruolo"
attributeID="myRuolo" />
```

Finito? NO, manca la definizione dell'attributo!

Davide Vaghetti – davide.vaghetti@garr.it

Definizione e uso attributo myRuolo

In `attribute-resolver.xml`:

```
  <resolver:AttributeDefinition xsi:type="ad:Simple"
id="myRuolo" sourceAttributeID="myRuolo">
        <resolver:Dependency ref="myRdbms" />
        <resolver:AttributeEncoder
xsi:type="enc:SAML1String" name="urn:x-
corsoidp:dir:attribute-def:myRuolo" encodeType="false" />
        <resolver:AttributeEncoder
xsi:type="enc:SAML2String"
name="urn:oid:1.3.6.1.4.1.99999999.1.1.1.1"
friendlyName="myRuolo" encodeType="false" />
  </resolver:AttributeDefinition>
```

Per poter rilasciare l'attributo dobbiamo configurare anche `attribute-filter.xml`. E' una AttributeRule come le altre:

```
[...]
    <AttributeRule attributeID="myRuolo">
            <PermitValueRule xsi:type="ANY" />
    </AttributeRule> [...]
```


Davide Vaghetti – davide.vaghetti@garr.it

Esercizio

- configurare un `RelationalDatabase DataConnector` completando il file di configurazione:
  - attribute-resolver-full.xml: `/opt/shibboleth-idp/conf/attribute-resolver-myRdbms.xml`
  - connessione:
    - `<dc:BeanManagedConnection>` = MyDataSource
  - query:
    - select da tabella shibboleth.RuoliOrganizzativi, chiave `uid`
  - campi e attributi:
    - campo da trasformare in attributo: `ruolo`
    - attributo da referenziare: `myRuolo`
- configurare una `AttributeDefinition` per `myRuolo`:
  - fonte (dependency): `DataConnector` definito;
- aggiungere il nostro nuovo resolver a `services.xml`:

```
<util:list id ="shibboleth.AttributeResolverResources">
    <value>%{idp.home}/conf/attribute-resolver.xml</value>
    AGGIUNGERE QUI IL NOSTRO RESOLVER CUSTOM
</util:list>
```

- configurare una `AttributeRule` per `myRuolo` in `attribute-filter.xml`

Davide Vaghetti – davide.vaghetti@garr.it