Object Template

Introduction

Object template is an object that holds a set of rules how a specific object type (e.g. user) should be constructed. It usually contains mappings tha t either compute user properties or assign accounts and roles to the user (usually with a condition).

User template can be used for a variety of very interesting identity management configuration. For example:

- Rule-based RBAC (RB-RBAC) configuration assigns a role to user automatically. The assignment is usually based on values of some user properties, e.g. employee type or organizational unit. This can be implemented as a mapping that gives user an assignment with a condition that it triggered by the desired property value.
- Creating full name of the user using a mapping. The mapping is usually based on given name and family name of the user. If the mapping is weak then it provides a default value for full name (it will **not** overwrite existing value). If the mapping is strong then it will enforce consistency of `fullName` property with `givenName` and `familyName` (in this case).

Examples

You can check defined templates for a definition through GUI via Configuration->Repository objects->Object template (from List objects). Also advanced resources samples from midpoint development master branch (here) contain examples of templates used in a process of synchronization.

The user template object definition could look like this one:

```
<objectTemplate oid="c0c010c0-d34d-b33f-f00d-777222222333">
        <name>User Template CSV sync</name>

        <description>
            Alternative User Template Object.
            This object is used when creating a new account, to set it up as needed.
        </description>

  <mapping>
   <description>
                Property mapping.
                Defines how properties of user object are set up.
                This specific definition sets a full name as a concatenation
                of givenName and familyName.
            </description>
   <strength>weak</strength>
   <source>
    <path>$user/givenName</path>
   </source>
   <source>
    <path>$user/familyName</path>
   </source>
   <expression>
    <script>

<language>http://midpoint.evolveum.com/xml/ns/public/expression/language#Groovy</langu
age>
     <code>
      givenName + ' ' + familyName
     </code>
    </script>
   </expression>
   <target>
    <path>$user/fullName</path>
   </target>
  </mapping>

    </objectTemplate>
```

TODO: configuration examples

Global User Template

A user template may be applied globally by including the following snippet in system configuration **just after the "logging" element**:

```
<defaultUserTemplateRef oid="10000000-0000-0000-0000-000000000222"/>
```

The system configuration object is accessible through GUI via Configuration->Repository objects->System configuration(from List objects).
Templates available for use could be listed in similar way Configuration->Repository objects->Object template(from List objects).

Includes

Object template can include another object template. The include is a simple `includeRef` clause at the beginning of a template definition.

Object template include

```
<objectTemplate oid="10000000-0000-0000-0000-000000000222">
   <name>Complex User Template</name>

   <includeRef oid="10000000-0000-0000-0000-000000000223"/>

   <mapping>
       ...
```

Limitations and FAQ

Object template works just with a single object which is typically a user. Therefore it only has the data from this object and no other objects. It means that variables such as `$account` cannot be used in an object template. The reason for this is the separation of concerns principle. We try to design each component or mechanism in midPoint to do a single thing. This allows us a significant development advantage (debugging, testing) and also provides a better code reusability. The power of midPoint is to have simple principles (such as object template) that used over and over again and combined with other simple principles (such as inbound/outbound mappings) to create a flexible and comprehensive solution.

There is also another reason for **not** including account in object templates. The object template may be used even if the account is not available, e.g. when user is changed from the GUI. MidPoint is using relative changes therefore it is not always required to read all the accounts to process a change. And in fact the account may not even be available (e.g. because resource is temporarily down). Therefore it would be a very inconvenient and inefficient if account attributes are used in object templates.

Therefore if a property from an account (or other object) is needed in an object template there are several ways how to do it:

- Use inbound mappings instead. Inbound mappings can compute the value and place it directly in the user property. Inbound mappings can see both the account and the user. And midPoint can make sure that these mappings are executed at the right moment and are not executed when not needed. E.g. the mappings will be executed only when midPoint detects a change on the resource.
- Use combination of inbound mappings, extended user properties and object template. Use object template to the computation. Use inbound mappings to copy data from account to the user extended properties (`extension` part of user object). Then use the extended properties as input in the user template. This works around the separation of concerns boundaries. And as the extended properties are stored with the user in midPoint repository they are always available without the need to read the account all the time. Inbound mappings will make sure that these data are as fresh as possible.

See Also

- Mappings and Expressions
- Synchronization Examples