RoleService

OMiLAB Technical Documentation

Software version: v0.2.8

Contents

Revision History

Revision	Date	Author(s)	Description
1.0	12.04.16	D Goetzinger	Created Document

1 Service Details

Prerequisites for the user

- None

Prerequisites for administrative user

- None

Prerequisites for the service operator

- SQL
- LDAP
- Tomcat server administration skills
- Familiarity with the concepts of the OMiLAB Microservice Infrastructure

Dependencies

- ActivityLoggingService
- OpenLDAP
- MySQL

Frameworks

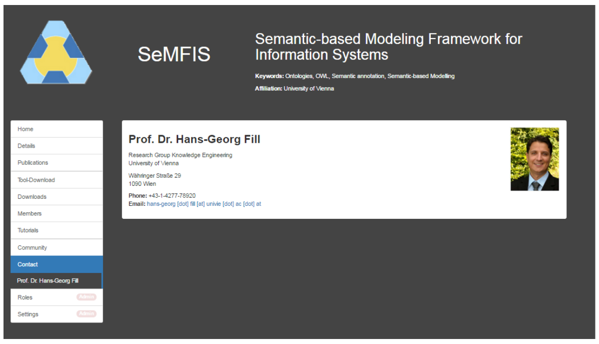
- Spring Boot 1.2.5.RELEASE and related
- Spring Security and Spring Security LDAP
- Apache Commons 3.4
- Google Guava 18

Summary
The RoleService provides functionality to assign roles to users within the context of projects. By means of this assignment it is possible give different rights to different users. There are a couple of roles, that are already pre-defined, although the RoleService does only provide the assignment. The decisions, whether somebody is allowed to perform a certain action have to be performed at the according service. The source of the users is the LDAP directory, which provides not only basic information like email and password, but also personal information about the user, like affiliation. This information is used to provide the contact overview and the member page.

2 User Manual

There are two main functionalities that are provided to the enduser by the RoleService. On the one hand there is the "Contact" page which provides an overview of the contacts assigned to a project, as in figure 2.1. It can be reached by accessing the endpoint *"contact"*.

On the other hand there is the "Member" page, which gives each user the possibility to join or leave project, see figure 2.2. It can be reached by accessing the endpoint *"member"*.


SeMFIS

Semantic-based Modeling Framework for Information Systems

Keywords: Ontologies, OWL, Semantic annotation, Semantic-based Modelling

Affiliation: University of Vienna

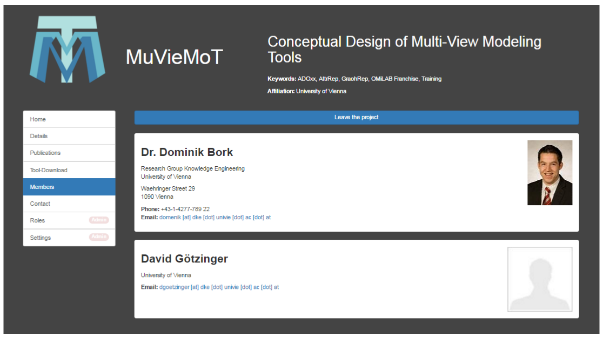
Prof. Dr. Hans-Georg Fill

Research Group Knowledge Engineering
University of Vienna

Währinger Straße 29
1090 Wien

Phone: +43-1-4277-78920
Email: hans-georg [dot] fill [at] univie [dot] ac [dot] at

Figure 2.1: Contact page


MuVieMoT

Conceptual Design of Multi-View Modeling Tools

Keywords: ADOxx, AttrRep, GraohRep, OMiLAB Franchise, Training

Affiliation: University of Vienna

Dr. Dominik Bork

Research Group Knowledge Engineering
University of Vienna

Waehringer Street 29
1090 Vienna

Phone: +43-1-4277-789 22
Email: domenik [at] dke [dot] univie [dot] ac [dot] at

David Götzinger

University of Vienna

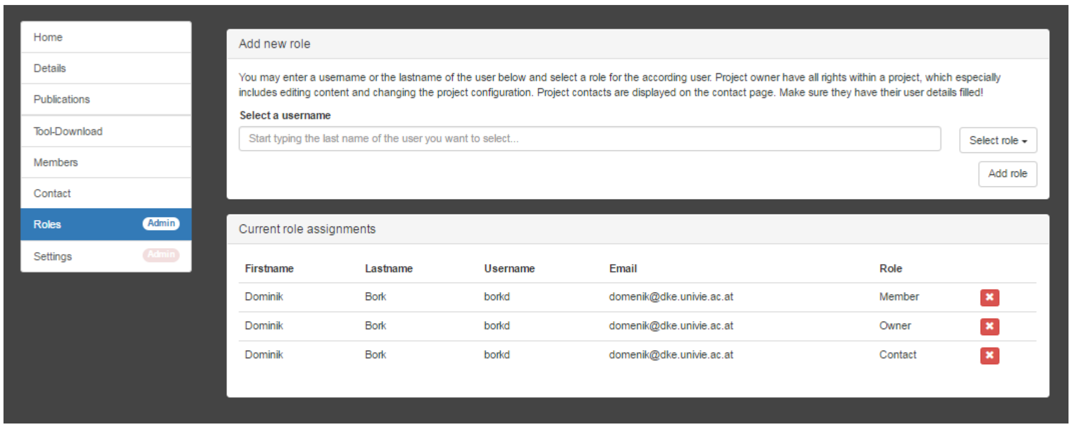
Email: dgoetzinger [at] dke [dot] univie [dot] ac [dot] at

Figure 2.2: Member page

The information, which is displayed in these two pages, is retrieved from the LDAP directory. For the OMiLAB there is a account management tool, customized based on PWM (https://github.com/pwm-project/pwm) available.

The administration of assignments can be found in the administrative view of the RoleService, as depicted in figure 2.3. You may directly enter the username of an OMiLAB user into the editable field, which says *"Select a username"*, select the according role from the selectbox and confirm the selection. If the entered username is valid and belongs to a OMiLAB user, the background of the editable field will turn green. If no user could be found, it will turn red. There is also the possibility to search for specific users, based on their lastname. Simply start typing the lastname into the editable field and some suggestions will be displayed, based on the input. If the suggestion is selected, the correct username will be put into the editfield. Furthermore the role assignments can be deleted by pressing the red cross next to each assignment. This applies particularly to automatically generated assignment, i.e. through the "Member" page.

Although the available roles can be easily extended, as described in the next chapter, a few have been predefined:



Add new role

You may enter a username or the lastname of the user below and select a role for the according user. Project owner have all rights within a project, which especially includes editing content and changing the project configuration. Project contacts are displayed on the contact page. Make sure they have their user details filled!

Select a username

Start typing the last name of the user you want to select...

Current role assignments

Firstname	Lastname	Username	Email	Role	
Dominik	Bork	borkd	domenik@dke.univie.ac.at	Member	✖
Dominik	Bork	borkd	domenik@dke.univie.ac.at	Owner	✖
Dominik	Bork	borkd	domenik@dke.univie.ac.at	Contact	✖

Figure 2.3: RoleService administration inteface

- **Project owner** The project owner has extensive rights regarding the administration of projects. Based on this assignment the PSM will allow the user to access the project settings, the role assignments and the administrative view of each instantiated service. The project owner is initially setup through the PSM Wizard.
- **Project contact** The project contact will be displayed at the page reachable by the endpoint *"contact"*. In order to have a valuable appearance, it would be good, if the according project contact has all information in his account fill in, e.g. photo, affiliation, ...
- **Project member** The project member assignments are made based on the *"join/leave"* functionality of the page, reachable by the endpoint *"member"*.

3 DevOps Manual

3.1 Installation

The installation of the RoleService may be performed similar to all other services. It can be built from source using maven as build system. The according target to create a deployable package, is *"package"*. The profile *"distribution"* provides a well-documented configuration with sane default values. So the command to build the RoleService using the command line interface can be *"mvn -Pdistribution package"*. This will create a deployable war archive in a new generated *"target"* folder.

Before a deployment on the application server is possible, the service has to be configured for the specific environment, what is performed in the file *"application.properties"*. If the configuration-section of the application-distribution.properties has not already been edited before building, the file can be found in the war archive at *"/WEB-INF/classes/application.properties"*.

```
# Address where the application is externally reachable
app.url = http://www.omilab.org/roleservice

# Credentials to access the database
spring.datasource.url = jdbc:mysql://localhost:3306/role
spring.datasource.username = role
spring.datasource.password = password

# Enter access to LDAP here, where the OMiLAB user data
   resides
ldap.url=ldap://omildap.dke.univie.ac.at:389
# Base for LDAP
ldap.base=dc=omilab,dc=org

# URL where the endpoint of ActivityLoggingService is
   reachable (without trailing slash)
omilab.activity = http://localhost:8080/logging
# SID (used for ActivityLoggingService) has to be unique
   across all services and is used to identify the service
omilab.sid = ROLE-VIE-01
```

What is left to do after the configuration, is the preparation of the database. When the option *"spring.jpa.hibernate.ddl-auto"* is set to *"validate"* the service expects to find valid schema at the database given and will exit, if it is not present. In order to create the necessary schema the file *"schema.sql"* has to be imported to the configured database. This file contains already some standard configuration for the RoleService, i.e. the roles "PROJECT_OWNER", "PROJECT_CONTACT" and "PROJECT_ MEMBER". The PSM expects a role with the internal representation "PROJECT_OWNER" and will not work properly, if it is not present

After the service is configured and the database schema is imported, the service may be deployed on the application server. The service has been tested extensively with Apache Tomcat 7 and JRE 7, as well as MySQL 5.5.

3.2 Configuration

The core of the RoleService is the table *"security_association"* which depicts the the ternary relation between the project, represented through the service instance, the role and the LDAP username. This is the table that is edited through the user interface.

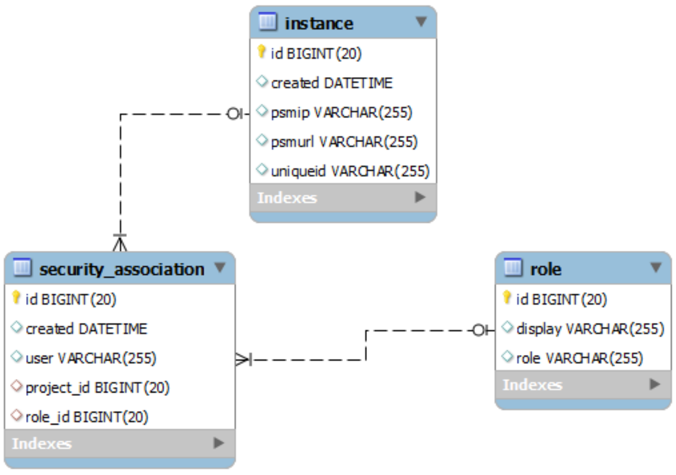


Figure 3.1: RoleService database scheme

In order to make another role available it has to be added to the table *"role"*. The column *"role"* should contain the internal representation of the role. It will not be visible to the enduser and should be unique in the whole infrastructure. The column *"display"* should contain a clean version of the name, that is also displayable to the user in the interface.

An example on how to add a new role looks like the following:

```
INSERT INTO role (display,role) VALUES ('New Role','NEW_ROLE
   ');
```

This is all it needs, to make the new role usable. After the according entry has been created in the database, it is possible to assign the *"New Role"* to users. Furthermore service developers will see in the OMiLAB Generic Interface, if users have this role assigned, and may make decisions based on this assignment.

If a certain role is still in use by some users, you may remove all the assignments using a certain role, e.g. "PROJECT_CONTACT" and the role itself with the following statements:

```
DELETE sa FROM security_association sa INNER JOIN role ON
   role.id = sa.role_id WHERE role = 'PROJECT_CONTACT' ;
DELETE FROM role WHERE role = 'PROJECT_CONTACT';
```

3.3 Interface and Implementation details

OMiLAB Generic Interface

Using the OMiLAB Generic Interface it is possible to create new role assignments. This is very feasible during the PSM Wizard Process, as it allows to automatically define a project contact and project owner during the creation of a project. This kind of assignments can be made be specifying the name of the user and the display name of the role. By using adequate numbering, multiple assignments can be made:

```
{
  "params": {
    "roleuser1": "$USERNAME",
    "rolename1": "Owner",
    "roleuser2": "$USERNAME",
    "rolename2": "Contact"
  },
  "roles": [
    "PROJECT_OWNER"
  ],
  "username": "$USERNAME"
}
```

OMiLAB Role Interface

As the PSM and the Repository, both depend on the RoleService to determine the allowed actions for each user, it employs other interfaces than the OMiLAB Generic Interfaces, to facilitate the queries of role information. In detail, it enables to query the roles of a user in the context of a project. In order to identify one can use the local instance id or the unique id of a project, which is saved during instantiation. The pattern on how to query the URL looks like the following:

```
http://<host>:<port>/<applicationcontext>/rest/role/instance
    /<localinstanceid>/user/<username>
http://<host>:<port>/<applicationcontext>/rest/role/id/<
    projectuniqueid>/user/<username>
```

A concrete example:

```
http://vienna.omilab.org/role/rest/role/instance/2/user/fill
http://vienna.omilab.org/role/rest/role/id/3760fcec-92f0-443
    e-ba76-575ca8903121/user/fill
```

Both queries will show the same response, as they refer to the same project. Based on the answer of this query, one can see that the user "fill" is "Owner" and "Contact" of the according project

```
[
    {
        "id": 2,
        "role": "PROJECT_OWNER",
        "display": "Owner"
    },
    {
        "id": 3,
        "role": "PROJECT_CONTACT",
        "display": "Contact"
    }
]
```

Additionally the images from the LDAP can be directly retrieved, if the username is known. The following shows the pattern, how the images can be retrieved directly and accordingly an example.

```
http://vienna.omilab.org/<servicecontext>/view/images/<
    username>.jpg
http://vienna.omilab.org/role/view/images/fill.jpg
```

LDAP attributes

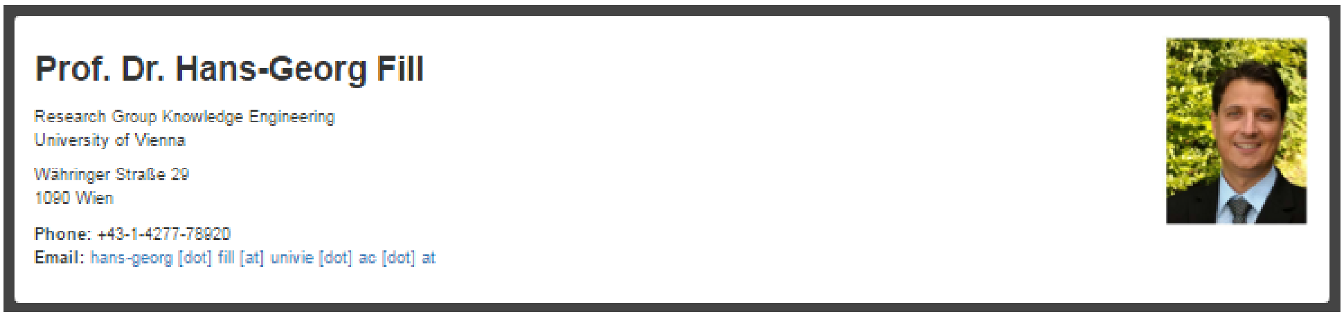
As discussed in the previous chapter, the information in the *"contact"* page and *"member"* page depend on the information in the LDAP. The following attributes are used:

LDAP attribute	Example	Description
mail	hans-georg.fill@univie.ac.at	This field denotes the email address of a user. Certain applications require the email attribute to be unique. This is also enforced by the PWM, so no users with the same email address may register. Although it is possible to enter this manually, as there are no contrains enforced by the LDAP directory.
cn	fill	This fields denotes the username and thus also the path in the LDAP directory
userPassword	(password)	This fields denotes the password, hashed using the SHA algorithm
givenName	Hans-Georg	This field denotes the first name of the user.
sn	Fill	This field denotes the lastname of a user
jpegPhoto	(binary)	This field saves the photo of the user binary in jpeg.
o Name	University of Vienna	This field denotes the affiliation of a user.
ou	Research Group Knowledge Engineering	If there is a sunit of the affiliation, it can be specified here.
postalAddress	1090 Wien	This field denotes zip code and the location.
street	Währinger Straße 29	This field denotes the street and the street number of the users office.
title	Prof. Dr.	This field denotes the academic title of the user.
telephoneNumber	+43-1-4277-78920	This field denotes telephone number of the users office.

The visualization of the attribute values, used in the example, can be seen in figure 3.2. What can be seen here, is that the email address is never available in plain text. For the display some special characters are written out. Also the *"mailto"* links are escaped accordingly (character entities).

AJAX autocompletion

As discussed in the previous chapter the RoleService provides functionality for autocompleting the lastnames. This is done by using JSONP, as the autocomplete features should not be routed through the PSM for performance reasons, but are rather directed



Prof. Dr. Hans-Georg Fill

Research Group Knowledge Engineering
University of Vienna

Währinger Straße 29
1090 Wien

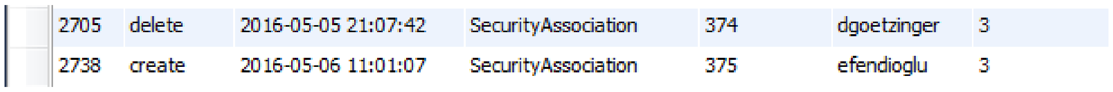
Phone: +43-1-4277-78920
Email: hans-georg [dot] fill [at] univie [dot] ac [dot] at

Figure 3.2: Example of contact

to the service itself. In order to protect the autocomplete feature, so that not every anonymous user can use it, a protection, similar to CSRF has been implemented. Each time, the page is accessed, a random will be generated, embedded into site and stored for 30 minutes at the server. This token will be sent to the server, whenever an AJAX call to the autocomplete API of the RoleService is triggered. Based on this information, the server may recognise, if the origin of the request is a user that has the appropriate rights in the PSM or not. As the tokens are invalidated after 30 minutes, no further access to the autocomplete feature is available after this specific time period. An according warning and a reminder to reload the page is displayed, if this time period is reached.

3.4 Logging Events

This section should take a closer look, on what events are sent to the LoggingService by the RoleService. The creations and the removal of the database concept *"SecurityAssociation"* are sent to the LoggingService.



2705	delete	2016-05-05 21:07:42	SecurityAssociation	374	dgoetzinger	3
2738	create	2016-05-06 11:01:07	SecurityAssociation	375	efendioglu	3

Figure 3.3: Example log entries

4 Contact

4.1 Service Developer

David Götzinger

Research Group Knowledge Engineering
Faculty of Computer Science
University of Vienna

Währinger Straße 29
1090 Vienna
AUSTRIA

Email: dgoetzinger@dke.univie.ac.at