Repository

OMiLAB Technical Documentation

Software version: v0.2.1

Contents

Revision History

Revision	Date	Author(s)	Description
1.0	12.07.16	D Goetzinger	Created Document

1 Service Details

Prerequisites for the user

- None

Prerequisites for administrative user

- None

Prerequisites for the service operator

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

Dependencies

- Jackrabbit
- CAS
- RoleService
- PSM
- ActivityLoggingService

Frameworks

- Apache Jackrabbit 2.8
- Jasig CAS Library 3.2
- Google Guava 18.0
- FileManager https://github.com/simogeo/Filemanager
- C5Connector https://github.com/th-schwarz/C5Connector.Java

Summary
The Repository provides functionality similar to a file explorer. It can be used to upload all kinds of reuseable assets, like images, ADOxx libraries, modelling toolkits, etc. It facilitates the basic management and download of the uploaded content and integrates with the PSM and RoleServices. Please be aware, that the downloads are not restricted and everybody, knowing the right link, can download anything.

2 User Manual

The Repository can be accessed from different locations and using different user interface components.

Figure 2.1: Access from project settings

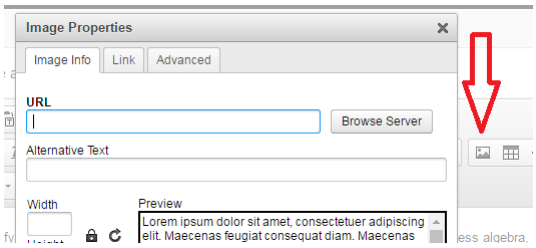

Figure 2.2: Access from WYSIWYG editor

In the figures 2.1 and 2.2 you can access the Repository by clicking on the "Browse server". Figure 2.1 shows an option that is often used when a dedicated value is needed, for example the "Settings" page in the PSM or in the DownloadService. Figure 2.2 shows the option than can be used within the WYSIWYG editor, that is widely used within the OMiLAB Portal to embed images.

After clicking the "Browse Server" a new window similar to the one depicted in figure 2.3 will appear. OMiLAB Projects are represented as folders, that may be accessed to store further data. You will only see folders, that you have permission to access. If you are missing the folder of your project, please make sure that you have the role "Owner" in the respective project and that the respective service is instantiated for the project. It is not possible to create a folder manually or upload a file to this main page. You will have to use a PSM to instantiate the Repository in order to create a new folder.

After you have entered a concrete folder you will see view similar to figure 2.4. This is the place where you may create new folders and files at will. Please be aware, that there is no further access restriction and that anybody with the correct link may download the files.

Selecting one item will present several options to further modify the name, as depicted in figure 2.5. The file may be deleted, renamed and replaced. Usually there will also be a button to "select" the according item. If you click this, the window of the Repository will be closed and the URL will be filled into the according edit field. Please note, that you might have to adjust other links, after you have changed something like the name, if you have used the file multiple times. This will not be done automatically.

Adding a new file can be done through the "Upload" dialog. You will be presented with an interface similar to what is depicted in figure 2.6. At this stage the FileManager

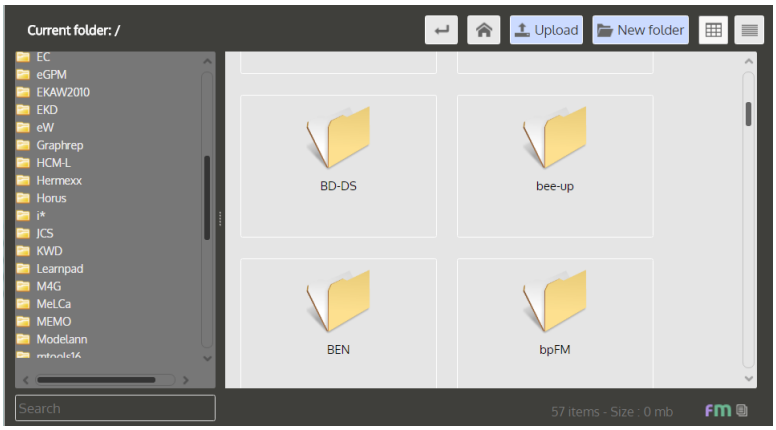

Figure 2.3: Repository overview

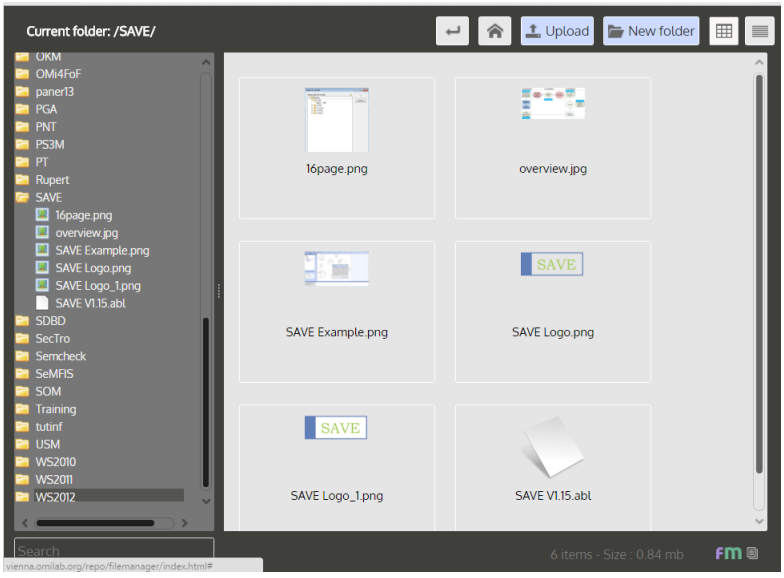

Figure 2.4: Repository folder

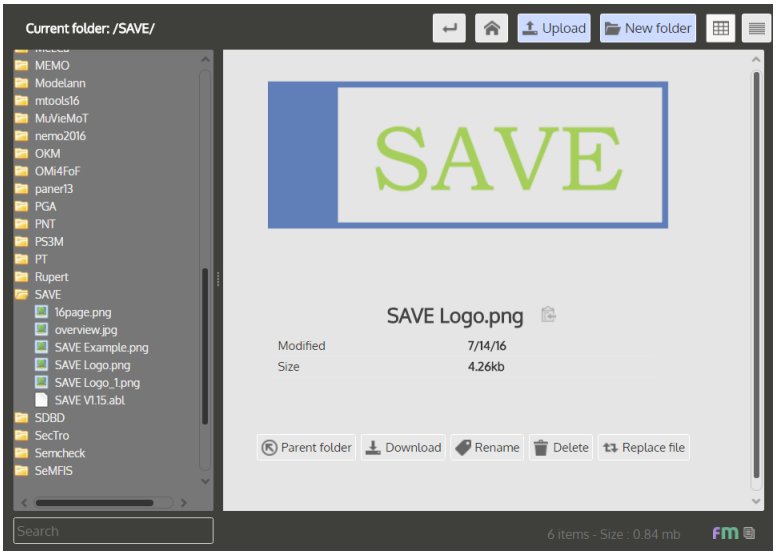

Figure 2.5: Repository file

supports only one simultaneous upload. It can be either drag and dropped into the marked are, or clicking on this area will open the regular file browsing dialog of your operating system for selection.

Figure 2.7 depicts how an upload in progress looks like. After the green progress bar has filled and the actual upload has completed, please keep the window open until you receive a message similar to the one depicted in figure 2.8. The time in between is needed for the internal reorganisation of the Repository.

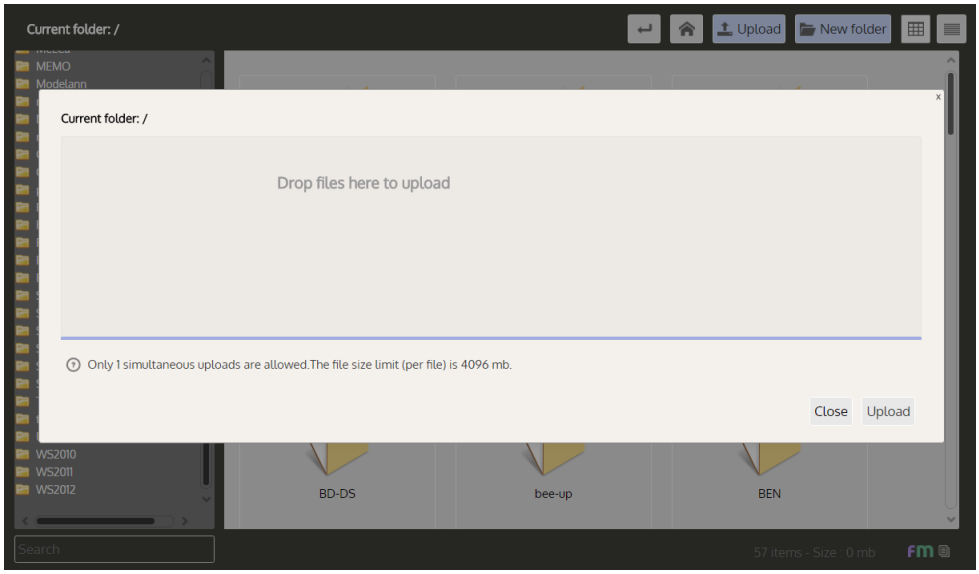

Figure 2.6: Repository file

Figure 2.7: Upload in progress

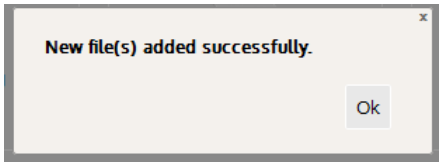

Figure 2.8: Upload confirmation

3 DevOps Manual

3.1 Installation and Configuration

The installation of the Repository differs from the installation process of the other services, as it is not based on Spring Boot but on other open source projects. The deployment of this application is similar to the deployment of standard java web applications.

It is strongly suggested to use the simplified installation for this package. Nevertheless this section will discuss how a manual installation can be performed.

Before building

In the case of the Repository several configuration variables will be set during the build process based on the settings on the main configuration file. The Repository will have to be rebuilt for every configuration change and it is under no circumstances sufficient to simply some binary packages. As of this there is no binary package distributed officially through jenkins.dke.univie.ac.at/release

The different configuration that are set based on the according maven profile are located at: *"/src/main/resources/INSTALL/config/default-distribution.properties"*. For this guide we will assume the profile "distribution".

```
# Location of the role service (without trailing slash)
omilab.role = http://www.omilab.org/role
# Define location of PSM (without trailing slash)
# Needed to dynamically resolve project names
omilab.psm = http://www.omilab.org/psm

# Responses from the role service can be cached. Please specifiy here, if
# the responses should be cached and how long (in minutes).
# Changes in user roles may take up to this "cachetime" until they take
# effect
omilab.role.cache = true
omilab.role.cachetime = 1

# Responses from the psm can be cached. Please specifiy here, if
# the responses should be cached and how long (in minutes).
```

```
16 # Changes in project abbreviation may take up to this "
      cachetime" until they take
17 # effect
18 omilab.psm.cache = true
19 omilab.psm.cachetime = 1
20
21 # URL where the endpoint of ActivityLoggingService is
      reachable (without trailing slash)
22 omilab.activity = http://localhost:8080/logging
23 # SID (used for ActivityLoggingService) has to be unique
      across all services and is
24 # used to identify the service
25 omilab.sid = REPO-01
26
27 # User listed here are not queried to the role service, but
      instead are assumed to be owner
28 # in all projects, thus they see all projects
29 omilab.admin = dgoetzinger
30 # should under no circumstances be used, as you might
      destroy the jackrabbit scheme!!
31 omilab.poweruser =
```

Building

The build process can be started regularly using the command *"mvn -Pdistribution package"*. This will create a deployable war archive in a new generated *"target"* folder.

After building

Finally the tomcat installation has to be adjusted. Within the *"INSTALL"* directory a file called *"jcr-2.0.jar"* can be found. This file has to be copied to the shared lib directory of the Tomcat installation. In the case of Ubuntu 14.04 it is located at *"jcr-2.0.jar"*. Additionally in the root of the Tomcat installation a directory *"jackrabbit"* has to be created (absolute path in Ubuntu 14.04 is *"/var/lib/tomcat7/jackrabbit"*). Within the directory *"INSTALL"* two example configuration files (repository.xml.X) can be found. The second one has to be copied to the previously created jackrabbit folder and renamed to *"repository.xml"*. The file has to be filled with database credentials. Please use the same database and credentials at **both** of the **two** settings. Also make sure that all of these files and folders have the correct permissions (*"jcr2.jar"* and the *"jackrabbit"* folder). Commands similar to the following might be used:

```
1 $ chown -R tomcat7:tomcat7 <file or folder>
2 $ chmod -R 755 <file or folder>
```

3.2 Interface and Implementation details

Architecture

The "FileManager" is a pure JavaScript application, that composes the frontend of the Repository through REST calls. The source of this application can be found at https://github.com/simogeo/Filemanager. The documentation of the interfaces between the "'FileManager" and the backend application "C5Connector" can be found at: https://github.com/simogeo/Filemanager#api. Further documentation is available at: https://github.com/simogeo/Filemanager/wiki. The "C5Connector" is responsible for mediating between the inputs of the "FileManager" and "Apache Jackrabbit" which comprises the backend database. Documentation regarding the the "C5Connector" can be found at: https://github.com/th-schwarz/C5Connector.Java

Figure 3.1 visualizes the relation between these three components.

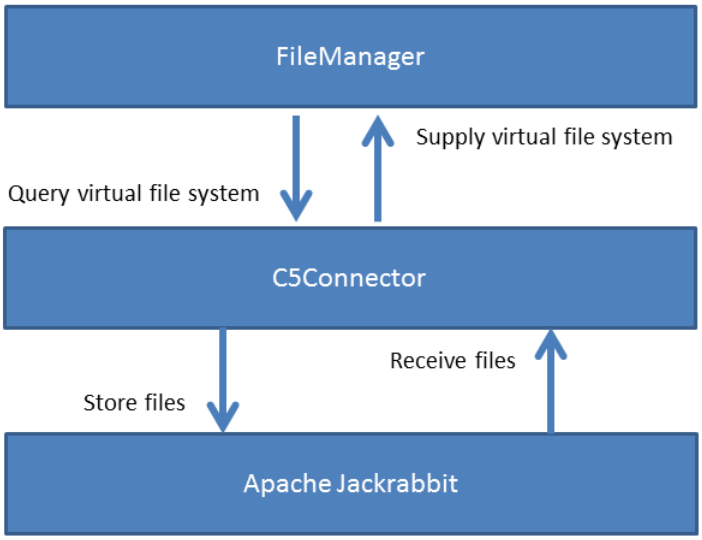

Figure 3.1: Architecture

More information about the Apache Jackrabbit project can be found at: http://jackrabbit.apache.org/jcr/index.html. It has been chosen as file backend, as it contains a lot useful features and is very flexible. It stores files and all kinds of other content on flexible schema that is aware of the notions "Nodes" and "Edges". Furthermore it features functionality like a revision history. At this stage this more advanced features of Jackrabbit are not used, and the "Nodes" and "Edges" are used to create a file explorer like experience.

Figure 3.2 illustrates how the Repository is integrated into the OMiLAB Portal, as it is not implemented like other Microservices. Solely the PSM interfaces regarding the service lifecycle, consisting in creation and destruction (of the projects folders) are implemented. Additionally the user information from the "OMiLAB Generic Request" format is not used, the Repository rather connects directly to the CAS, if it needs login

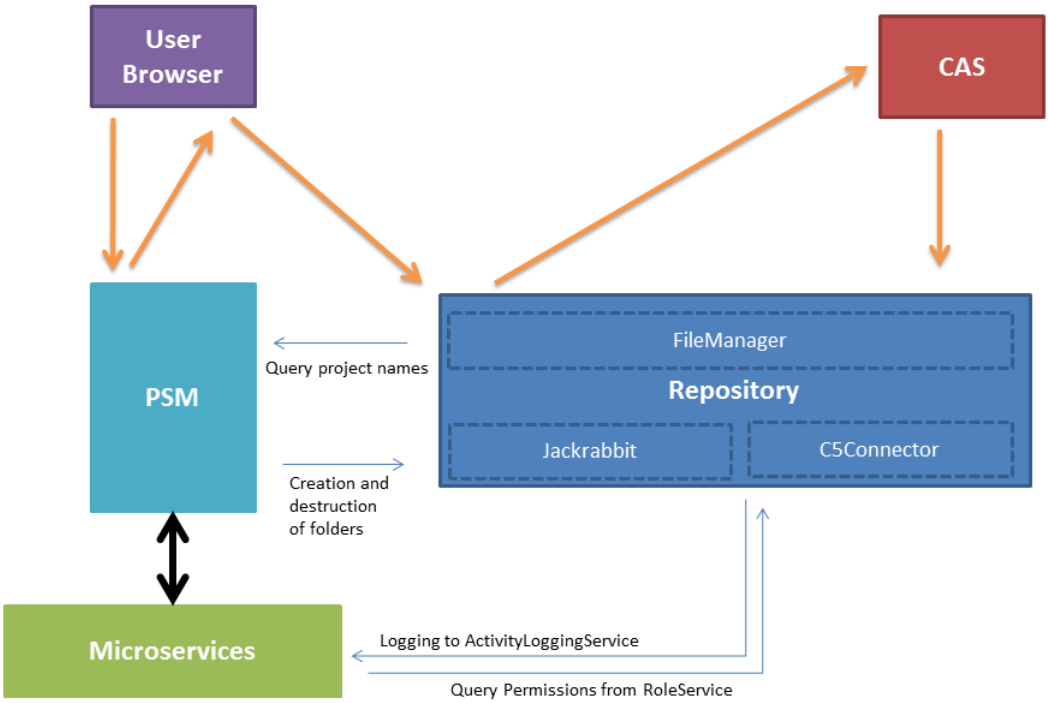

Figure 3.2: Infrastructure integration

information. In addition to the lifecycle interfaces of the PSM , the Repository uses a hidden interface of the PSM, to resolve the project names. Each project in the PSM an unique id assigned (UUIDv4). This is internally used as the folder name. Whenever somebody access the Repository, it will query the PSM for each of this IDs and fill in the abbreviation of a project instead of the ID. On the one hand this helps to propagate changes of the project abbreviation to the Repository. As this abbreviation is also used in the downloadlink, this implies on the other hand, that a change of the project abbreviation will invalidate all downloadlinks in use. Furthermore the Repository is integrated with the RoleService, that facilitates the display of folders that only the current user has access to. This integration is also made possible by the according unique ids.

Calls to Repository are done through JavaScript in the browser of the user, as discussed in the following subsection.

Integration between other services and Repository

```
1 <script>
2  var repourl = "http://vienna.omilab.org/repo/filemanager"
3 </script>
4 <input class="form-control" id="data" value="URL COMES HERE"
     type="text">
5 <span class="input-group-btn">
6     <button class="btn btn-default" type="button" id="browse
       " onclick="queryRepository('data');">BROWSE</button>
7 </span>
```

It is easy to integrate the Repository into your own Service. In order to do so, you will need a HTML “input” tag, similar to the own in line 4 with an arbitrary id, in this case “data”. Furthermore you will need a button like the one in line 6, that has the according “onclick” event. Make sure that the argument of the “queryReopsitory” function is the id of the “input” tag, where the output from the Repository should be stored. Furthermore it is necessary to supply the path to the Repository somewhere through the “repourl” variable.

3.3 Logging Events

This section should take a closer look, on what events are sent to the LoggingService by the Repository. All CRUD actions regarding files and folders are sent to the LoggingService.

28	delete	2015-09-21 15:41:05	nt:file	/omilab/631ae026-00de-4919-b935-e74885c78961/test3.png	dgoetzinger	4
49	create	2015-09-23 11:51:34	nt:folder	/omilab/3760fcec-92f0-443e-ba76-575ca8903121/My_folder	dgoetzinger	4
50	upload	2015-09-23 12:03:02	nt:file	/omilab/3760fcec-92f0-443e-ba76-575ca8903121/ADOxx15.zip	dgoetzinger	4

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