# openmatDB

***Release 0.1***

**Feb 06, 2020**

# Contents:

*openmatDB* (open Modular and exTendable Data Base) is a free (see *Licenses*) data base framework primarily aimed at scientific/engineering data base applications with a focus on easy extendability through user plug-ins (e.g. for specific and complex data analysis).

By using a plug-in concept which relies on RESTful APIs and JSON as an exchange format, *openmatDB* can easily be extended to support plug-ins written in different languages or plug-ins distributed on multiple remote servers.

Since even basic data base functions are provided by plug-ins the code base of the core modules is kept small and simple.

For more details on the *openmatDB* concept and its features see the *Introducion*.

*openmatDB* is currently mainly written in *python* and *JavaScript* (for the web-interface) and its components have a number dependencies on 3rd party libraries - see *Software components* for details.

For details on the licences of the *openmatDB* components see *Licenses*.

---

**Note:** *openmatDB* is currently in its early development stages and not suitable for a production system. However A complete test environment (including a desktop and a browser based GUI) can however be downloaded - see *Test environment*.

---

CHAPTER 1

# Introducion

## 1.1 Framework overview

The *openmatDB* environment basically consist of three components:

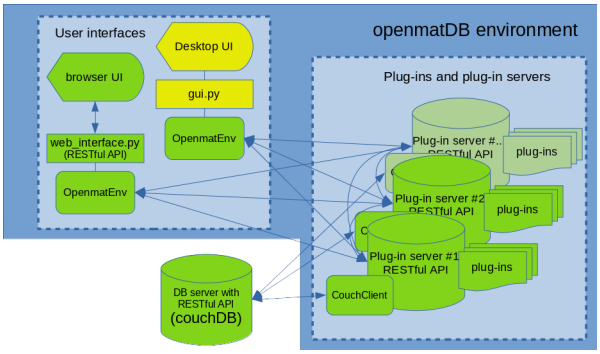
- a data base server exposing a RESTful API (currently couchDB is used)
- one or more plugin-server(s)
- graphical user-interface(s) (desktop and/or browser based)

The plug-in servers form the core of the *openmatDB* environment. They communicate with each other, the UIs and the data base. They host the plug-ins, that carry out all operations. Since the communication is based on RESTful APIs and data is exchanged in form of JSON strings plug-in servers for different programming languages can be created with comparably low effort (as long as JSON handling and HTTP servers are available for that language).

The UIs basically collect "plug-in run-requests" from the users - i.e. name of the requested plug-in and the parameters (e.g. record IDs, record attributes for performing a calculation).

The working principle is explained in more detail in *UI request processing procedure*.

The following illustration provides an overview of the system:




As can be seen in the illustration two classes provide the interfaces of *openmatDB* environment:

- OpenmatEnvironment - provides mainly routing information and easy access to the API calls of the plug-in servers
- CouchClient - provides data base access for plug-ins

The communication principle and the use of the interfaces is illustrated in the following sub-chapters.

## 1.2 Plug-in server initialization procedure

When a plug-in server is launched following steps are performed:

1. The plug-in server will query the *register* function of each plug-in on the server
2. The *register* function will then return information on required to work with the plug-in (e.g. data types for its input parameters and return values, GUI elements to create to capture these input parameters) - for details see the *Plug-ins* chapter.
3. With the return values of all plug-ins the plug-in server create a *plugin_config* dictionary, which it can share with the UIs

## 1.3 UI initialization procedure

This section will illustrate the steps of the initialization process of an *OpenmatEnvironment*, when a user interface is launched.

1. The UI needs to create an *OpenmatEnvironment" object and one or more calls to the *add_plugin_server* method
2. The *add_plugin_server* methods sends some HTTP requests to the plug-in server, which returns information about the plug-ins it is hosting (i.e. it returns the *plugin_config* as explained above)
3. The *add_plugin_server* method then updates the *server_list* and the *plugin_route_table* (which is a look-up table for plug-in names and their corresponding server - its purpose will be explained in more detail in the *UI request processing procedure* chapter.
4. Afterwards the *plugin_route_table* will be send to all plug-in servers in the *server_list* and each server will contain a copy of the *plugin_route_table*.

The animation below illustrates the UI initialization process for an example, where two plug-in servers are added to the environment.

## 1.4 UI request processing procedure

When the user has selected a plug-in, entered the required parameter values into the UI elements (e.g. line edits, spin boxes, combo boxes,...) and launched the plug-in (e.g. by pressing a 'Run plug-in' button) the UI layer sends a call to the UI's *OpenmatEnvironment* object's *run_plugin* function.

The procedure can be sub-divided into following steps:

1. The *run_plugin* method is called with the plug-in name and parameters provided by the UI layer
2. The *run_plugin* method will first perform a look-up in the *plugin_route_table* to find the server, which is hosting the corresponding plug-in

3. It will send an HTTP POST request to the plug-in server with plug-in name and parameter values encoded in the JSON
4. The plug-in server will decode the request and calls the requested plug-in's *run* function passing the parameter values provided by the request as the function arguments
5. The plug-in will then execute its code and eventually encounter a request to run another plug-in.
6. In this case the plug-in server's *run_plugin* method is called.
7. The *run_plugin* method will fetch the server address, which is hosting the requested plug-in and dispatch a plug-in run-request analogous to the request from the UI
8. A plug-in can also request a couple of data base interactions - i.e. the plug-in will call one of the *CouchClient*'s data base request method's
9. The *CouchClient* will then send an HTTP request to the data base and return the (decoded) response to the plug-in
10. The plug-in will eventually do further processing on the data and/or return the information to the */run_plugin* API call, which will a response with the plug-ins results to the sender (i.e. a *run_plugin* method of either a UI or another plug-in server)
11. When a result is finally returned to the UIs *run_plugin* method, it will decode the information and send it to the UI layer, which can render the results accordingly by using the *return_type* definition of the corresponding plug-in stored in the *plugin_config* variable (which might tell the UI to render the result e.g. as a chart or a table)

The animation below illustrates the UI request processing procedure for a common example.

In this example a user request is received to run "pluginC" (hosted by plug-in server #2). The "pluginC" however relies on "pluginA", which is stored on plug-in server #1. While "pluginC" does not request any data base interaction "pluginA" will raise a request for a record stored in the data base.

In a real application "pluginC" could be an implementation of a statistical evaluation, which first calls another plug-in to fetch the required data for the analysis by forwarding some search parameters provided by the user.

CHAPTER 2

# Software components

The source code of the different components of *openmatDB* are currently provided on 5 different repositories on *gitlab*.

**Warning:** *openmatDB* is currently in its early development stage and not suitable for a production environment! However A complete test environment (including a desktop and a browser based GUI) can however be downloaded - see *Test environment* - for testing/development purposes.

For an overview on the licenses on the different software components see the *Licenses* chapter. For more details on licensing and copyright information refer to the **NOTICE** and **LICENSE** files in the corresponding source code repository.

## 2.1 pyopenmatdb

*pyopenmatdb* provides the main components of the system - i.e. the python classes for the interfaces *OpenmatEnvironment* and *CouchClient* and the plug-in server.

*openmat-plugins* is released under Apache License Version 2.0. The source code can be obtained from: https://gitlab.com/thecker/pyopenmatdb

*pyopenmatdb* is written python and depends on following packages:

- python3
- Requests
- Flask
- Flask-CORS

## 2.2 openmat-plugins

*openmat-plugins* is a collection of plug-ins for *openmatDB*'s python plug-in server.

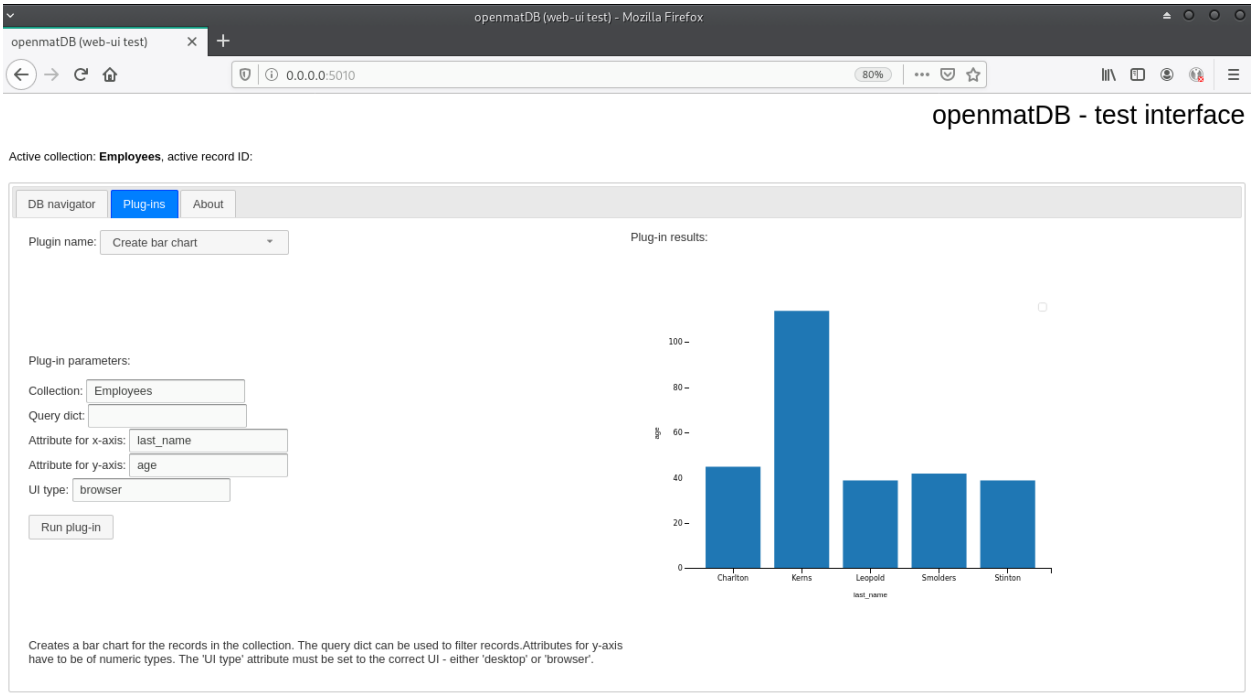
*openmat-plugins* is released under Apache License Version 2.0. The source code can be obtained from https://gitlab.com/thecker/openmat-plugins

The plug-ins depend on following python packages:

- python3
- pandas
- xlrd
- matplotlib
- mpld3
- NumPy
- pyopenmatdb

## 2.3 openmat-webui

The *openmat-webui* repository provides a browser based GUI to the *openmatDB* environment.




openmatDB - test interface

*openmat-webui* is free software released itself under the Apache License Version 2.0. The source code con be obtained from https://gitlab.com/thecker/openmat-webui

It comes bundle with following software components:

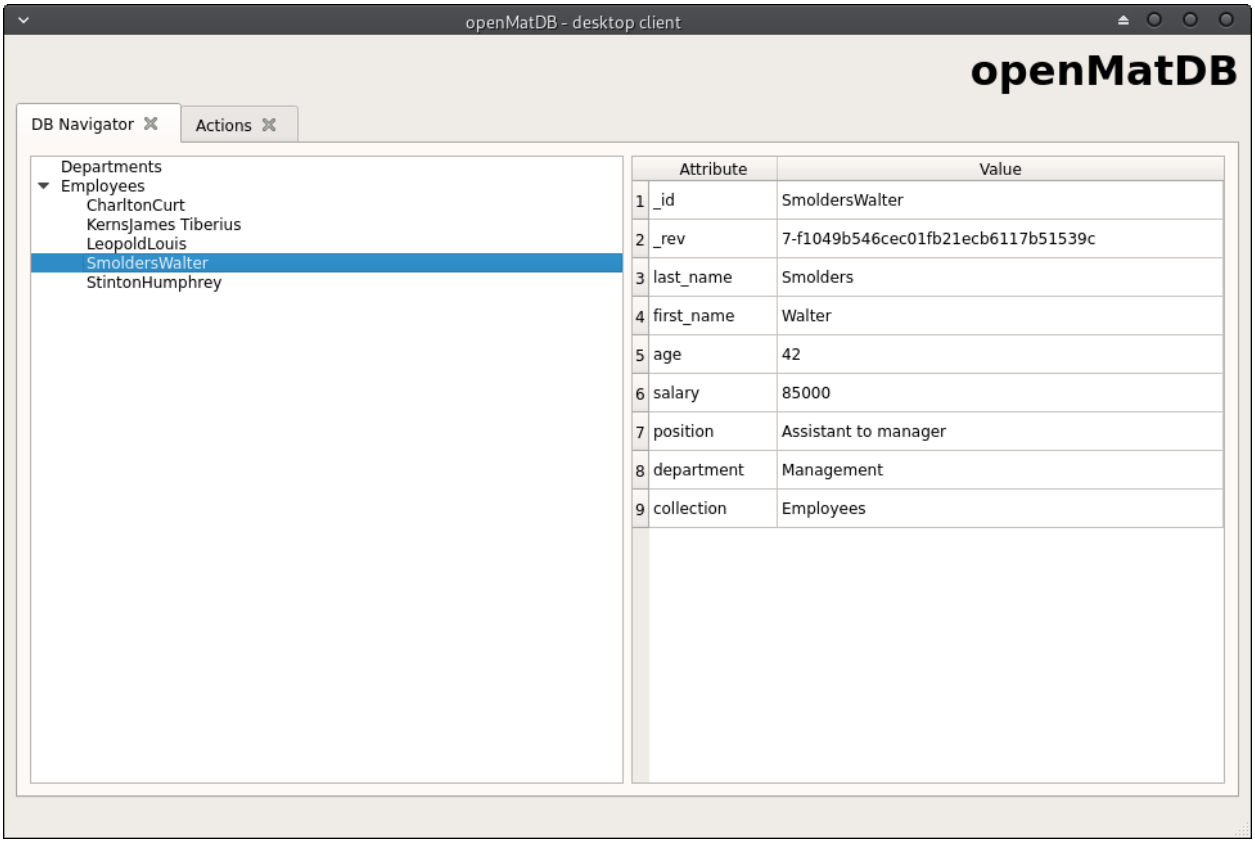
- jQuery JavaScript Library v1.12.4, Copyright jQuery Foundation and other contributors
- jQuery UI v1.12.1, Copyright jQuery Foundation and other contributors
- jQuery UI-themes v1.12.1, Copyright jQuery Foundation and other contributors
- Fancytree, v2.34.0, Copyright 2008-2019 Martin Wendt

*openmat-webui* is written python and JavaScript. The JavaScript dependencies are bundled with the software (see above). For the python part following packages are required:

- python3
- Flask
- Flask-CORS
- pyopenmatdb

## 2.4 openmatGUI

The *openmatGUI* repository provides a GUI, which runs as a local desktop application.




*openmatGUI* is free software released under the GNU GENERAL PUBLIC LICENSE Version 3 (GPLv3).

See NOTICE and LICENSE file for details.

*openmatGUI* is written in python and depends on following python packages:

- Python3
- PyQt5
- matplotlib
- pyopenmatdb

## 2.5 openmat-testenv

The *openmat-testenv* repository contains a collection of configuration files and shell scripts, which automate setting up and launching a test environment for *openmatDB* - for details see the *Test environment* chapter.

*openmat-testenv* is free software released under the Apache License Version 2.0.

In order to run the shell scripts provided by this repository following programs are required (besides a UNIX shell):

- Docker CLI
- cURL

Furthermore if you run the shell scripts following pre-built docker images will be downloaded to your system:

- couchDB:latest, https://hub.docker.com/_/couchdb
- python:3.7, https://hub.docker.com/_/python

The python:3.7 image will be used create two images for the openamtDB environment (openmat-py_plugin:latest and openmat-webui:latest). Note, that the python packages (and their dependencies) required by *pyopenmatdb*, *openmat-plugins* and *openmat-webui* will be downloaded and installed on the images as well (see chapters above).

CHAPTER 3

# Licenses

This chapter lists the licences disclaimers of the individual software components, which are part of the *openmatDB* environment. For details on the license of the individual components see also the LICENSE and NOTICE files inside the corresponding source code repository:

- https://gitlab.com/thecker/pyopenmatdb
- https://gitlab.com/thecker/openmat-plugins
- https://gitlab.com/thecker/openmat-webui
- https://gitlab.com/thecker/openmatGUI
- https://gitlab.com/thecker/openmat-testenv

## 3.1 pyopenmatdb

```
Copyright 2020 Thies Hecker

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

 http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

## 3.2 openmat-plugins

```
Copyright 2020 Thies Hecker

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

 http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

## 3.3 openmat-webui

```
openmat-webui - a browser based GUI for the openmatDB environment

    Copyright 2020 Thies Hecker

    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

     http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.


3rd party Libraries bundled with openmat-webui
==============================================

This project comes bundled with following 3rd party libraries:

* jQuery JavaScript Library v1.12.4
* jQuery UI v1.12.1
* jQuery UI-themes v1.12.1
* Fancytree v2.34.0

The copyright notices and license disclaimers can be found below.

jQuery JavaScript Library v1.12.4
---------------------------------

    jQuery JavaScript Library v1.12.4
    http://jquery.com/

    Includes Sizzle.js
```

(continues on next page)

(continued from previous page)

```
    http://sizzlejs.com/

    Copyright jQuery Foundation and other contributors
    Released under the MIT license
    http://jquery.org/license

In openmat-webui the re-distributed file (including the license information)
of jQuery is located in:
/webpage/static/jquery-ui/external/jquery/jquery.js


jQuery UI v1.12.1
-----------------

    Copyright jQuery Foundation and other contributors, https://jquery.org/

    This software consists of voluntary contributions made by many
    individuals. For exact contribution history, see the revision history
    available at https://github.com/jquery/jquery-ui

    The following license applies to all parts of this software except as
    documented below:

    ====

    Permission is hereby granted, free of charge, to any person obtaining
    a copy of this software and associated documentation files (the
    "Software"), to deal in the Software without restriction, including
    without limitation the rights to use, copy, modify, merge, publish,
    distribute, sublicense, and/or sell copies of the Software, and to
    permit persons to whom the Software is furnished to do so, subject to
    the following conditions:

    The above copyright notice and this permission notice shall be
    included in all copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
    EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
    MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
    LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
    OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
    WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

    ====

    Copyright and related rights for sample code are waived via CC0. Sample
    code is defined as all source code contained within the demos directory.

    CC0: http://creativecommons.org/publicdomain/zero/1.0/

    ====

    All files located in the node_modules and external directories are
    externally maintained libraries used by this software which have their
    own licenses; we recommend you read them, as their terms may differ from
    the terms above.
```

(continues on next page)

(continued from previous page)

```
In the openmat-webui repository the re-distributed files (including the
original license file) of jQuery UI are located under:
/webpage/static/jquery-ui/


jQuery UI-themes v1.12.1
------------------------

    Copyright jQuery Foundation and other contributors, https://jquery.org/

    This software consists of voluntary contributions made by many
    individuals. For exact contribution history, see the revision history
    available at https://github.com/jquery/jquery-ui

    The following license applies to all parts of this software except as
    documented below:

    ====

    Permission is hereby granted, free of charge, to any person obtaining
    a copy of this software and associated documentation files (the
    "Software"), to deal in the Software without restriction, including
    without limitation the rights to use, copy, modify, merge, publish,
    distribute, sublicense, and/or sell copies of the Software, and to
    permit persons to whom the Software is furnished to do so, subject to
    the following conditions:

    The above copyright notice and this permission notice shall be
    included in all copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
    EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
    MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
    LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
    OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
    WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

    ====

    Copyright and related rights for sample code are waived via CC0. Sample
    code is defined as all source code contained within the demos directory.

    CC0: http://creativecommons.org/publicdomain/zero/1.0/

    ====

    All files located in the node_modules and external directories are
    externally maintained libraries used by this software which have their
    own licenses; we recommend you read them, as their terms may differ from
    the terms above.

In the openmat-webui repository the re-distributed files (including the
original license file) of jQuery UI-themes are located under:
/webpage/static/jquery-ui-themes/
```

(continues on next page)

(continued from previous page)

```
Fancytree v2.34.0
-----------------

    Copyright 2008-2019 Martin Wendt,
    https://wwWendt.de/

    Permission is hereby granted, free of charge, to any person obtaining
    a copy of this software and associated documentation files (the
    "Software"), to deal in the Software without restriction, including
    without limitation the rights to use, copy, modify, merge, publish,
    distribute, sublicense, and/or sell copies of the Software, and to
    permit persons to whom the Software is furnished to do so, subject to
    the following conditions:

    The above copyright notice and this permission notice shall be
    included in all copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
    EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
    MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
    NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
    LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
    OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
    WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

In the openmat-webui repository the re-distributed files (including the
original license file) of Fancytree are located under:
/webpage/node_modules/jquery.fancytree/
```

## 3.4 openmatGUI

```
openmatGUI is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

Foobar is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with openmatGUI.  If not, see <https://www.gnu.org/licenses/>.
```

## 3.5 openmat-testenv

```
Copyright 2020 Thies Hecker

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
```

(continues on next page)

(continued from previous page)

```
You may obtain a copy of the License at

 http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

CHAPTER 4

# Plug-ins

## 4.1 General

Plug-ins form the base for *openmatDB's* functionality. Even basic functions like returning the collections in the data base are provided by plug-ins. I.e. although the *openmatDB* user interfaces can run without any plug-in, they are of no real use without them.

In order to allow users to easily extend the data base functionality the plug-in concept is kept as simple as possible.

The concept is based on these principles:

- Each plug-in has to have two mandatory functions - *run* and *register*
- The *run* function is called when a plug-in is launched. It has only one mandatory argument - the *session* argument - which is a representation of the server that runs it (i.e. in case of the python plug-in server a *PythonPluginServer* class object - see API documentation
- Further arguments to the run function could be a collection name, a record ID, an attribute in a record,...
- The *register* function is used to inform a GUI how to handle the plug-in's input and output - e.g.:
- it provides the GUI with UI element definitions and data types required to define each of its parameter's values (e.g. we would like to have a line edit to receive a collection name as a string or a spin box returning an integer to filter records based on a numeric attribute value)
- it also proves information about the return type of the plug-in, so that the GUI knows how to render the results (e.g. as plain text, as a table or a plot chart)
- additional information about the plug-in is provided in order to eventually render a help string, author names, license info, etc.
- plug-ins are able call other plug-ins. This way complex functions can be realized with a small code base in the actual *openmatDB* modules
- plug-ins can be written in different languages. Currently only a python interface is available, but since the plug-ins communicate with other plug-ins by exchanging information as JSON objects new language bindings (e.g. R, C++) can be created with relatively low effort.

- plug-ins shall work without modification on both the desktop UI and the web-interface (which is given because the plug-ins do not directly communicate with the GUI layer. The GUI only needs to support methods to render the different input and return types of the plug-ins)
- plug-ins shall also be usable without any GUI - e.g. allow batch processing

## 4.2 Structure of a plug-in

As mentioned above a plug-in can consist of only two mandatory functions - *run* and *register*. In the example below the code for a simple plug-in, which gets a specific record from the data base by its ID is shown:

```
def run(session, collection=None, recordID=None):
    record = session.db_client.get_record_by_ID(collection=collection, record_
↪ID=recordID)

    return record


def register():
    return {
        'name': 'Get record by ID',
        'UI_elements': [
            {'type': 'TextInput',
             'label': 'Collection',
             'default_value': '',
             'data_type': 'str'},
            {'type': 'TextInput',
             'label': 'Record ID',
             'default_value': '',
             'data_type': 'str'},
        ],
        'help': 'Returns a dictionary representation of record identified by its ID',
        'return_type': 'text'
    }
```

As can be seen the *run* function takes three arguments: the session object, the name of the collection the record belongs to and the record's ID.

In the *run* function the *get_record_by_ID* method of the *session*'s *db_client* is used to communicate with the data base. For data base functions provided by the - have a look at *API for creating plug-ins in python*.

The *register* function does nothing except returning a dictionary with the plug-in config info.

When the plug-in is selected in the GUI's plug-in browser the plug-in's name (defined in 'name'), the help string (defined 'help') and two labelled text input fields (as defined in 'UI_elements') will be created - one for the collection and the other for the record ID. See *Defined UI element types* for details on the usage of the UI element entries.

The help string should clearly state, what the plug-in is doing, so that the users understand its functionality and can eventually use it in combination with their own plug-ins.

The return type (defined by 'return_type') is important for the GUI to correctly render the output of the plug-in. In this case (return type 'text') the output will just be rendered as plain text in a text box. See *Defined return types* for details.

As mentioned before plug-ins can also call other plug-ins. To illustrate this - the *run* function of a more complex plug-in (which returns a tree representation of the data base) is shown:

```
def run(session):
    collections = session.run_plugin('get_collections')

    tree = {}
    for collection in collections:
        records = session.run_plugin('get_records', *[collection, {}])
        record_ids = [record['_id'] for record in records]
        tree[collection] = record_ids

    return tree
```

The you can see this plug-in uses the session's *run_plugin* method to launch other plug-ins and process their return values. In this case both the 'get_collections' and the 'get_records' plug-ins are used.

## 4.3 Defined UI element types

Besides the 'type' key all UI elements support the keys: 'label', 'default_value' and 'data_type'.

Some UI elements may allow/require additional keys. The following UI element types are currently supported:

**TextInput** A line edit

---

**Note:** This list is work in progress and will be extended in future releases.

---

## 4.4 Defined return types

The return type defines how the data shall be rendered in the GUI. Following return types are currently supported:

**text** The returned results will be rendered as a text string a text box

**pyplot** The returned results have to be rendered as a plot chart.

---

**Note:** This list is work in progress and will be extended in future releases.

---

CHAPTER 5

# Test environment

**Warning:** The test environment will run some servers on docker containers on your local machine. In order to allow communication between the desktop-UI and the plug-in servers or to access the web-UI via a browser some ports of the containers are exposed to the host system. Read the Docker security section on the docker website for details on security with regard to docker containers.

The test environment is not intended for production environments - it should only be used for testing and development!

## 5.1 Prerequisites

*openmatDB* is developed and tested under GNU/linux. Since most of the code is based on interpreted languages it should be possible to run on other platforms (e.g. MS Windows) as well, but for this chapter we assume you are running a linux machine.

For building the test environment following programs have to installed on your system:

- docker
- cURL
- python3

Every major linux distribution should have official packages for these tools.

## 5.2 Obtain the source code

In order to setup a full test environment you will first have to download/clone the source code of all *openmatDB* repositories - see *Software components*.

Assuming you want to store the data for the *openmatDB* environment in a folder called “openmatdb” in your home directory, open a terminal and run following commands:

```
cd ~
mkdir openmatdb
cd openmatdb
git clone https://gitlab.com/thecker/pyopenmatdb.git
git clone https://gitlab.com/thecker/openmat-plugins.git
git clone https://gitlab.com/thecker/openmat-webui.git
git clone https://gitlab.com/thecker/openmatGUI.git
git clone https://gitlab.com/thecker/openmat-testenv.git
```

## 5.3 Build docker images

In order to build the docker images required to run the containers change into the *openmat-testenv* folder and start the setup script:

```
cd openmat-testenv
./setup_env.sh
```

This shell script will pull pre-built docker images for *couchDB* and a *python* environment. It build images for the *openmatDB* plug-in server and the web-ui (openmat-py_plguin:latest and openmat-webui:latest) based on the *python* image.

To build the *openmatDB* images the python packages (and their dependencies) required by *pyopenmatdb*, *openmat-plugins* and *openmat-webui* will be downloaded and installed on the images as well.

Check if the docker network 'openmat-net' exists and create it if not (the containers will communicate inside this network and only expose dedicated ports to the host system)

## 5.4 Start docker containers

After you have build the docker images, you can start the containers (make sure the docker daemon is running). A shell script to start the containers with the corresponding configuration is provided.

```
./start_all.sh
```

This script will do following steps:

- start the *couchDB* instance - container name: "couch_test"
- get the IP address of *couchDB* inside the docker network and update plug-in server config files
- start the first plug-in server - container name: "openmat_py_plugins"
- start the second plug-in server - container name: "openmat_py_plugins2"
- get the IP addresses of the plug-in servers in the docker network and update web-interface config files
- start the server for the web-interface (in interactive mode) - container name: "openmat_webui"

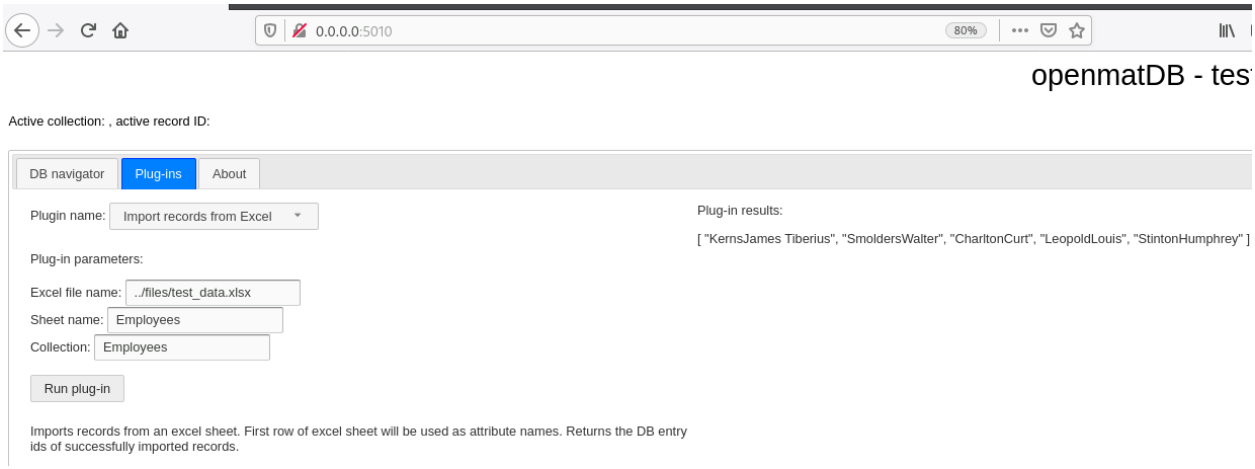
You should now see the log of the web-interface (e.g. incoming HTTP requests) in the terminal.

If you open up a browser window at http://0.0.0.0:5010 you should now get access to the *openmatDB* web-interface. However the data base is still empty.

**Note:** In order to easily modify the programs running on the containers, the folders of the cloned *openmatDB* repositories are mapped to the containers internal file systems. I.e. all changes made to the repository folders will be reflected in the containers.

## 5.5 Importing test data

The easiest way to import test data is to go to the web-interface and in the "plug-ins" tab select the "Import records from Excel" plug-in. The parameters will default to an excel file which is mapped to the filesystem of the plug-in server (you can find it in the "py_plugins1_conf/files" sub directory inside your "openmat-testenv" folder). So you can just click the "Run plug-in" button. You should see a list of record names, which have been imported from the excel file.




## 5.6 Connecting with the desktop-GUI

You can simultaneously connect with the desktop-GUI. Before starting the desktop-GUI you should do 2 steps:

First you should make sure, that the required python packages to run the desktop-GUI are installed.:

- requests
- PyQt5
- matplotlib

You can install these either via your linux distributions packages, *pip* or *conda*,... E.g. via *pip* just open up a new terminal type:

```
pip install requests PyQt5 matplotlib
```

Next you need to add the python modules in the *pyopenmatdb* folder to your PYTHONPATH:

```
cd ~/openmatdb/pyopenmatdb
export PYTHONPATH=$PYTHONPATH:$(pwd)
```

Afterwards you can start the desktop-GUI:

```
cd ../openmatGUI
python gui.py
```

**Note:** Packages for *pip* and *conda* will be provided for future releases, which will take care of the dependency issues and the steps above.

## 5.7 Stopping the test environment

To stop the containers just press CTRL+C in the terminal. You should see a message "All containers stopped successfully" in the end.

## 5.8 Additional remarks

Note that you can also use *Fauxton* (an administration tool that comes with *couchDB*) to access the data base. To do that open your browser at http://127.0.0.1:5984/_utils/

CHAPTER 6

# API reference

This section describes the API of the *pyserver* and *pyopenmatdb* module.

## 6.1 API for creating plug-ins in python

Each plug-in's *run*-function shall have *session* as its first argument. *session* is an instance of *PythonPluginSever*. It has an attribute *db_client* which holds an instance *CouchClient* - i.e. providing direct data base functions to the plug-ins.

### 6.1.1 PythonPluginSever

### 6.1.2 CouchClient

**class** openmatdb.pyopenmatdb.**CouchClient**(*api_url*, *db_name*, *record_id_fields*)

Client for the RESTful API of couchDB

**Parameters**

- **api_url** (*str*) – Url to data base (incl. port number)
- **db_name** (*str*) – Data base name
- **record_id_fields** (*list*) – List of str - record attribute names to join to record Ids.

**api_url**

Url to data base (incl. port number)

**Type** str

**db_name**

Data base name

**Type** str

**record_id_fields**

List of str - record attribute names to join to record Ids.

**Type** list

**get_collections**()
Get collections

**add_collection**(*collection_name*)
Adds a new view/collection

**add_record**(*collection*, *record_data*)
Adds a record to the data base

**add_records**(*collection*, *record_list*)
Add records to a collection

**Parameters**

- **collection** (*str*) – Name of the collection
- **record_list** (*list*) – List of dictionaries with record attributes and values

**Returns** List of valid records

**Return type** list

**get_records**(*collection*, *query_dict=None*)
Returns records as a list

**get_record_by_ID**(*collection*, *record_ID*)
Get record by ID - collection

## 6.2 API for creating UIs in python

Each UI needs to communicate with the plug-in servers. The *OpenmatEnvironment* class provides an interface for this.

### 6.2.1 OpenmatEnvironemnt

**class** openmatdb.pyopenmatdb.**OpenmatEnvironment**
Manager for the openmatDB environment (mainly communication with plug-in servers)

**plugin_servers**
A list of server information. Dict for each server with keys: 'server_name', 'url', 'plug-ins', 'plugin_language'

**Type** list

**active_record_id**
ID of the currently active record (to be set by UI)

**Type** str

**active_collection**
Name of the active collection (to be set by UI)

**Type** str

---

**Note:** The active... attributes can be used to provide default values to plug-ins based on currently selected items in the UI.

---

**get_plugin_server_data**(*url*)
Retrieves information from a plug-in server (i.e. server name, plug-in language, available plug-ins + config) :param url: URL to plug-in server API :type url: str

**Returns** with keys: 'server_name', 'url', 'plug-ins', 'plugin_language'

**Return type** dict

**add_plugin_server**(*url*)
Adds a plug-in server reference and sends an updated plug-in route table to all servers

**Parameters** **url** (*str*) – URL to the plug-in servers API

**plugin_config**
Returns the plug-in configs of all plug-ins in the system

**plugin_route_table**
Returns a dict of plug-in module names and their corresponding server url

**Type** dict

**run_plugin**(*plugin_name*, **params*)
Starts a plug-in launch request

**Parameters**

- **plugin_name** (*str*) – Name of the plug-in to run
- **params** (*list*) – List of parameters to pass to plug-in

## 6.3 RESTful API for building plugin-servers

TBD

CHAPTER 7

# Indices and tables

- genindex
- modindex
- search

# Index