fm-metadata-service Documentation

Release 1.0.0

Courtney Pacheco

May 19, 2016

Contents

Doc version 2.0.0

Date: 18 May 2016

CHAPTER 1

Overview

This repository contains a metadata-providing service for Fedora Modularization. The service is used by the **modulemd** here: https://pagure.io/modulemd

1.1 Requirements

- python 2.7 - https://www.python.org/download/releases/2.7/
- mongodb - https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/
- httpd - https://httpd.apache.org/docs/current/programs/httpd.html
- bs4 - https://www.crummy.com/software/BeautifulSoup/bs4/doc/#installing-beautiful-soup

To install the required packages

```
>>> sudo dnf install python2 httpd python-beautifulsoup4
```

Then follow the mongodb instructions in the link above.

1.2 Installation

To install this package, make sure you have installed the required packages first. From there, you have two options:

1. Use the Vagrant file here: https://pagure.io/fm-overview.git
2. Install via git:

```
>>> cd /var/www/html && mkdir fm
>>> git clone https://pagure.io/fm-metadata-service.git /var/www/html/fm
```

1.3 Files & Folders in this Repository:

- **Scripts:**
  - **databaseUtilities.py** - Database utilities for importing & retrieving module metadata
  - **demo.py** - A demo script for running the metadata service
  - **fm.py** - Web server script for obtaining metadata

- **Other files:**
  - **Makefile** - For building Python Nose tests & the rpm builder
- **Folders:**
  - **package** - Contains an rpm builder for developing your own test rpm packages
  - **test** - Python Nose unit tests

CHAPTER 2

Capabilities

fm-metadata-service allows users to search for modules by:

- Name
- License
- Version
- URL
- References (community, documentation, tracker)
- Relese
- Koschei Monitor
- Monitor
- Review URL
- Upstream URL
- Dependencies information
- xmd
- Status (Approved/Rejected)
- ACLS
- modulemd data

CHAPTER 3

databaseUtilities Examples

The following examples illustrate how to properly use **databaseUtilities.py**. These examples include querying the mongodb database and clearing all data from the database.

Most examples are taken from **demo.py**, so if you would like to run these examples and see their outputs, you can simply run **demo.py**.

3.1 1. Querying data from the Modularity Database

Below are examples of how to query by name, license, version, URL, etc.. Each query returns a list of dictionary objects. If no results are found, the output dictionary is empty.

Note: Queries are formed from dictionary objects and they follow the mongodb query format.

Simple Query: Query by name

```
myDb = ModularityDb()
q = {'name':'testmodule'}

print "result:\n", myDb.query(q)
```

Output:

```
>>> result:
>>> [{u'_id': ObjectId('573df289ed35590b340cfb61'),
     u'acls': [],
     u'creation_date': 1460472886.0,
     u'description': u'This is a test module for the modularity working group stuff.',
     u'koschei_monitor': True,
     u'modulemd': {u'data': {u'components': {u'rpms': {u'dependencies': True,
         u'fulltree': True,
         u'packages': {u'bar': {u'cache': u'https://example.com/cache',
           u'commit': u'26ca0c0',
           u'repository': u'https://pagure.io/bar.git'},
          u'baz': None,
          u'xxx': {u'arches': [u'i686', u'x86_64'], u'multilib': [u'x86_64']},
          u'xyz': None}}},
       u'dependencies': {u'buildrequires': {u'c-build': 6.0, u'core': 23},
        u'requires': {u'core': 23}},
       u'description': u'A module for the demonstration of the metadata format. Also, the obligatory
       u'license': {u'content': [], u'module': [u'MIT']},
       u'name': u'testmodule',
       u'references': {u'community': u'http://www.example.com/',
```

```
        u'documentation': u'http://www.example.com/',
        u'tracker': u'http://www.example.com/'},
       u'release': 1,
       u'summary': u'A test module',
       u'version': 1.0,
       u'xmd': None},
      u'document': u'modulemd',
      u'version': 0},
     u'monitor': False,
     u'name': u'testmodule',
     u'namespace': u'modules',
     u'review_url': u'https://bugzilla.redhat.com/12345',
     u'status': u'Approved',
     u'summary': u'A test module for modularity in stg',
     u'upstream_url': u''}]
```

More complex query: Query by references

```
myDb = ModularityDb()
q = {'modulemd.data.references' : {'documentation' : 'http://www.example.com/', 'community' : 'http:/

print "result:\n", myDb.query(q)
```

Output:

```
>>> result:
>>> [{u'_id': ObjectId('573df289ed35590b340cfb61'),
  u'acls': [],
  u'creation_date': 1460472886.0,
  u'description': u'This is a test module for the modularity working group stuff.',
  u'koschei_monitor': True,
  u'modulemd': {u'data': {u'components': {u'rpms': {u'dependencies': True,
      u'fulltree': True,
      u'packages': {u'bar': {u'cache': u'https://example.com/cache',
        u'commit': u'26ca0c0',
        u'repository': u'https://pagure.io/bar.git'},
       u'baz': None,
       u'xxx': {u'arches': [u'i686', u'x86_64'], u'multilib': [u'x86_64']},
       u'xyz': None}}},
    u'dependencies': {u'buildrequires': {u'c-build': 6.0, u'core': 23},
     u'requires': {u'core': 23}},
    u'description': u'A module for the demonstration of the metadata format. Also, the obligatory lor
    u'license': {u'content': [], u'module': [u'MIT']},
    u'name': u'testmodule',
    u'references': {u'community': u'http://www.example.com/',
     u'documentation': u'http://www.example.com/',
     u'tracker': u'http://www.example.com/'},
    u'release': 1,
    u'summary': u'A test module',
    u'version': 1.0,
    u'xmd': None},
   u'document': u'modulemd',
   u'version': 0},
  u'monitor': False,
  u'name': u'testmodule',
  u'namespace': u'modules',
  u'review_url': u'https://bugzilla.redhat.com/12345',
  u'status': u'Approved',
  u'summary': u'A test module for modularity in stg',
```

```
  u'upstream_url': u''}]
```

3.2 2. Clearing the Modularity Database

Use the `cleanup()` function to clear the database:

```
myDb = ModularityDb()
myDb.cleanup()
```

CHAPTER 4

Metadata Server Examples

Assuming you have installed this package properly, you should be able to use the metadata server automatically.

In this example, we have a server set up at 209.132.178.225.

4.1 Return Entire Database

To list all the contents of the database, we enter the following URL: http://209.132.178.225:8008/fm/list

This returns all the contents as a JSON string

4.2 Search Database by Name

To search the database for a module by name: http://209.132.178.225:8008/fm/info/**<name>**, where **<name>** is the name of a module.

For example, to search for *yakuake* (which doesn't exist), http://209.132.178.225:8008/fm/info/yakuake

To search for *testmodule2*, http://209.132.178.225:8008/fm/info/testmodule2