osf$_{a}pi_{v}2_{c}lient Documentation$

Release 0.1.0

Center for Open Science

September 09, 2015

Contents

NOTE: These docs are a work in progress. Feel free to contribute by creating an issue or submitting a pull request!

CHAPTER 1

osf_api_v2_client

A client for accessing the OSF v2 API

- Free software: Apache license
- Documentation: https://osf-api-v2-client.readthedocs.org.

1.1 Features

- Simplifies access of JSON dictionary (dict) data:
  - Instead of `user[u'attributes'][u'fullname']`, `user.attributes.fullname` can be used.
  - However, you are not locked into the simplified "dot" access. Both regular dict access and "dot" access will work, and they can be switched up:

```
user[u'attributes'].fullname
user.attributes[u'fullname']
```

  - This works recursively for dicts within dicts (as seen above, given

    `user = {u'attributes': {u'fullname': u'John Cleese'}}`),

    and for dicts inside a list of dicts (see below).
  - Given:

```
dict_with_list = {
    u'mylist': [
        {u'fullname': u'John Cleese'},
        {u'fullname': u'Terry Jones'},
        {u'fullname': u'Eric Idle'}
    ]
}
```

  - Calls such as the following can be used:

```
dict_with_list.mylist[0].fullname
dict_with_list[u'mylist'][2][u'fullname']
```

- Simplifies access of multiple items:

- Though items are returned from the API with only a certain number of items per page (often ten), this library takes care of pagination in the background, providing a generator to return all desired items one at a time within a loop.
- The following example prints the gravatar urls of the first 30 users in the OSF:

```
from osf_api_v2_client.session import Session

session = Session()
for user in session.get_user_generator(num_requested=30):
    print(user.attributes.gravatar_url)
```

CHAPTER 2

Installation

At this time the client library is not yet available on PyPI.

To use the library, clone the GitHub repo into the desired directory on your computer:

```
git clone https://github.com/jamiehand/osf_api_v2_client.git your_directory_here
```

The source code that makes up the client is found in the `osf_api_v2_client` directory within the main library directory (that is, in `osf_api_v2_client/osf_api_v2_client`).

CHAPTER 3

Usage

To use osf_api_v2_client in a project:

- In the command line:

```
cp osf_api_v2_client/settings/local-dist.py osf_api_v2_client/settings/local.py
```

- Modify `settings/local.py` to work for your purposes (see docstring at top of local-dist.py for how to do this.)
- In your module:

```
import osf_api_v2_client
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/jamiehand/osf_api_v2_client/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.
- Expected behavior and behavior you are experiencing.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

4.1.4 Write Documentation

osf_api_v2_client could always use more documentation, whether as part of the official osf_api_v2_client docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/jamiehand/osf_api_v2_client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *osf_api_v2_client* for local development.

1. Fork the *osf_api_v2_client* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/osf_api_v2_client.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv osf_api_v2_client
$ cd osf_api_v2_client/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 osf_api_v2_client tests
$ python setup.py test
```

   To get flake8, just pip install it into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes"
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.

3. The pull request should work for Python 2.7, 3.3, and 3.4. Check https://travis-ci.org/jamiehand/osf_api_v2_client/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

Use the command line to run a subset of the tests. You can run tests from modules, classes, or even individual test methods. For example:

```
$ python -m unittest test_nodes test_session
$ python -m unittest test_nodes.TestGetNodes
$ python -m unittest test_nodes.TestGetNodes.test_get_public_node_auth_non_contrib
```

CHAPTER 5

Credits

5.1 Development Leads

- jamiehand
- reinaH

5.2 Contributors

None yet. Why not be the first? See: CONTRIBUTING.rst

CHAPTER 6

History