pyf.aggregator Documentation

Release 0.1.0

Jens Klein, Veit Schiele, Lukas Guziel

Feb 14, 2023

Contents:

CHAPTER 1

Python Package Filter Aggregator

The Python Package Filter Aggregator (`pyf.aggregator`) aggregates the meta information of all Python packages in the PyPI and enhances it with data from GitHub.

1.1 Requirements

We need a running Typesense search engine.

1.2 Install

Git clone the package somewhere and change into the pyf.aggregator repository directory. Create a virtualenv and install the package

```
$ python -m venv venv
$ ./venv/bin/pip install -e .
```

For the connection to Typesense we need to define some environment variables in a .env file.

```
TYPESENSE_HOST=localhost
TYPESENSE_PORT=8108
TYPESENSE_PROTOCOL=http
TYPESENSE_API_KEY=<your_secret_typesense_apikey>
TYPESENSE_TIMEOUT=120
GITHUB_COOLOFFTIME=2
GITHUB_TOKEN=<your_secret_github_apikey>
```

1.3 Quickstart

To aggregate Plone content from PyPi run the following command:

```
$ ./venv/bin/pyfaggregator -ft "Framework :: Plone" -i -t packages1
```

The target Typesense collection is given here with `-t`.

To enrich the data with data from GitHub, run the following command:

```
$ ./venv/bin/pyfgithub -t packages1
```

Note: We need a typesense alias called "packages" pointing to the most recent packages collection: 'packages1'.

1.3.1 Add a typesense alias

```
$ ./venv/bin/pyfupdater --add-alias -s packages -t packages1
```

To list existing aliases, use the following command:

```
$ ./venv/bin/pyfupdater -lsa
{'aliases': [{'collection_name': 'packages1', 'name': 'packages'}]}
```

1.3.2 Create a search only api key

A search only api key is used by the client to search for Plone add-on's. It should be limited to a collection, in our case the collection alias "packages".

```
$ ./venv/bin/pyfupdater --add-search-only-apikey -t packages
res_key: {'actions': ['*'], 'collections': [''], 'description': 'Search-only key.',
↪'expires_at': 64723363199, 'id': 4, 'value': 'sHlV6xOtgsg0AaegA62eniyU5aALn1Os'}
```

or if you want to define the key your self, you can provide one:

```
$ ./venv/bin/pyfupdater --add-search-only-apikey -t packages -key␣
↪sHlV6xOtgsg0AaegA62eniyU5aAsn1Os
```

The pyfupdater command can also be used to migrate a Typesense collection to another, when you make bigger changes to the schema.

1.4 License

The code is open-source and licensed under the Apache License 2.0.

1.5 Credits

- @jensens
- @veit
- @guziel
- @pgrunewald

- @MrTango
- @pypa

CHAPTER 2

Installation

2.1 Stable release

To install pyf.aggregator, run this command in your terminal:

```
$ pip install pyf.aggregator
```

This is the preferred method to install pyf.aggregator, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for pyf.aggregator can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git@github.com:collective/pyf.aggregator.git
```

Or download the tarball:

```
$ curl  -OL https://github.com/collective/pyf.aggregator/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ pip install -e .[dev]
```

CHAPTER 3

Usage

`pyf.aggregator` is meant to be used with the following command-line tool

```
$ pyfaggregator --filter-name some.package --limit 10
```

This command triggers the aggregation of up to `10` result items according to the given filter query `some.package`. These items will be added to ElasticSearch service (running on default configuration, i.e. `localhost:9200`).

For more details on the pyfaggregator command, please refer to the `--help` option:

```
$ pyfaggregator --help
usage: pyfaggregator [-h] [-f] [-i] [-s [SINCEFILE]] [-l [LIMIT]] [-n [FILTER_NAME]]␣
↪[-t FILTER_TROOVE] [--github-token [GITHUB_TOKEN]]

Fetch information about pinned versions and its overrides in simple and complex/
↪cascaded buildouts.

optional arguments:
-h, --help            show this help message and exit
-f, --first           First fetch from PyPI
-i, --incremental     Incremental fetch from PyPI
-s [SINCEFILE], --sincefile [SINCEFILE]
                        File with timestamp of last run
-l [LIMIT], --limit [LIMIT]
-n [FILTER_NAME], --filter-name [FILTER_NAME]
-t FILTER_TROOVE, --filter-troove FILTER_TROOVE
--github-token [GITHUB_TOKEN]
                        Github OAuth token
```

3.1 Using GitHub API

For every package with an associated repository on GitHub, there will be made the attempt to retrieve interesting metadata (e.g. number of stars, number of watchers, last update etc.). Due to very strict rate limits for accessing the

GitHub API (at the time of writing: 60 requests / hour), there is the option to add your GitHub OAuth token to exploit the higher rate limit for registered users (at the time of writing: 5000 requests / hour).

To do so, create your token and pass it with the `--github-token` option.

When the rate limit is being hit, the aggregator will wait until the Github API can be accessed again.

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/collective/pyf.aggregator/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.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

pyf.aggregator could always use more documentation, whether as part of the official pyf.aggregator 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/collective/pyf.aggregator/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 *pyf.aggregator* for local development.

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

   ```
   $ git clone git@github.com:your_name_here/pyf.aggregator.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 pyf.aggregator
   $ cd pyf.aggregator/
   $ 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 pyf.aggregator tests
   $ python setup.py test or py.test
   $ tox
   ```

   To get flake8 and tox, just pip install them 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.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_pyf.aggregator
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

CHAPTER 5

Credits

5.1 Contributors

- @jensens
- @veit
- @guziel
- @tangoman78

CHAPTER 6

History

6.1 v1.0.0.dev0 yyyy-mm-dd

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search