onemillion Documentation

Release 1.0.0

Floyd Hightower

Oct 19, 2022

Contents

Contents:

CHAPTER 1

onemillion

Determine if a domain is in the Alexa or Cisco top one million domain list.

Documentation is available here: https://onemillion.readthedocs.io . There is a UI/API using this package available here: http://onemillion.hightower.space

1.1 Installation

The recommended means of installation is using pip:

```
pip install onemillion
```

Alternatively, you can install onemillion as follows:

```
git clone https://github.com/fhightower/onemillion.git && cd onemillion;
python setup.py install --user;
```

1.2 Usage

When using the default settings, the following steps will be taken when an instance of onemillion is initialized and the `domain_in_million` function called:

1. Check to see if the domain lists have been updated today.

2a. If they have been updated today, look for the given domain in the lists and stop.

2b. If the lists have not been updated today, make a `HEAD` request and check the current etag against the previous etag (stored locally) to see if the lists have been updated.

3a. If the etags are the same (meaning the lists have not been updated), look for the given domain in the lists and stop.

3b. If the etags are different (meaning the lists have been updated), make a request for the lists, unzip them, and save them in the default cache location (`~/.onemillion`).

4. Now that the lists are updated, search for the given domain in the lists.

1.2.1 Default Usage ~ Hello World!

Via Python

The default usage of onemillion is as follows:

```
import onemillion

o = onemillion.OneMillion()
o.domain_in_million("google.com")   # 1
o.domain_in_million("gaagle.com")   # None
```

Using the method described above, the alexa and cisco top one million domain lists as well as a bit of metadata will be stored in the home directory: `~/.onemillion`.

Via Command Line

The default usage of onemillion via command line is as follows:

```
onemillion google.com
```

As as when using onemillion in a python script, the alexa and cisco top one million domain lists as well as a bit of metadata will be stored in the home directory: `~/.onemillion`.

1.2.2 No Caching

Via Python

If you do not want to cache the domain lists, you can tell onemillion not cache anything by setting `cache=False` on initialization as demonstrated below:

```
import onemillion

# do not cache anything
o = onemillion.OneMillion(cache=False)
o.domain_in_million("google.com")   # 1
o.domain_in_million("gaagle.com")   # None
```

The code described above is fine if you are only making one or two calls or if storage space is a concern, but it is suggested that you cache the lists if feasible so as to limit traffic to the domain lists.

NOTE: currently, the 'No caching' configuration will throw an error. This will be updated and handled when issue #12 is fixed.

Via Command Line

Via command line, same principle as above:

```
onemillion google.com --no-cache
```

1.2.3 Custom Cache Location

Via Python

If you are caching the lists but want to cache them somewhere other than your home directory, you can specify a custom cache location by setting the `cache_location` parameter when initializing onemillion as demonstrated below:

```
import onemillion

# cache data to a specific path
o = onemillion.OneMillion(cache_location=<YOUR_PATH_HERE>)
o.domain_in_million("google.com")   # 1
o.domain_in_million("gaagle.com")   # None
```

This will cache the domain lists in the path you provide.

Via Command Line

Via command line, same principle as above:

```
onemillion google.com --cache_location ~/.cache/onemillion/
```

or

```
onemillion google.com -l ~/.cache/onemillion/
```

1.2.4 No Update

Via Python

If you have already run onemillion and have the domain lists cached, but do not want to keep updating them, you can specify `update=False` on initialization as demonstrated below:

```
import onemillion

# do not update cached content
o = onemillion.OneMillion(update=False)
o.domain_in_million("google.com")   # 1
o.domain_in_million("gaagle.com")   # None
```

Be aware that onemillion will, by default, check to see if it has already updated the domain lists today before making any requests. Thus, onemillion handles updating responsibly and intelligently by default and there are few cases in which this configuration (using `update=False`) is necessary. Nevertheless... it's there and you are welcome to use it.

Via Command Line

Via command line, same principle as above:

```
onemillion google.com --no-update
```

1.3 Credits

This package was created with Cookiecutter and the fhightower/python-project-template.

CHAPTER 2

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:

2.1 Types of Contributions

2.1.1 Report Bugs

Report bugs at https://github.com/fhightower/onemillion/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.

2.1.2 Fix Bugs

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

2.1.3 Implement Features

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

2.1.4 Write Documentation

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

2.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/fhightower/onemillion/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 :)

2.2 Get Started!

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

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

```
$ $ git clone git@github.com:<your_github_username_here>/onemillion.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 onemillion
$ cd onemillion/
$ 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 onemillion 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.

2.3 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/fhightower/onemillion/pull_requests and make sure that the tests pass for all supported Python versions.

CHAPTER 3

Credits

3.1 Development Lead

- Floyd Hightower <https://github.com/fhightower/>

3.2 Contributors

None yet. Why not be the first?

CHAPTER 4

History

4.1 0.5.0 (2017-08-12)

- Adding command line capability

4.2 0.4.5 (2017-05-09)

- Returning rank of the given host rather than just a boolean
- Adding (better) codecoverage when testing

4.3 0.4.1 (2017-04-25)

- Improving comments and documentation
- Separating the list updating from the domain check

4.4 0.4.0 (2017-04-12)

- Improved testing

4.5 0.1.0 (2017-04-04)

- First release on PyPI.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search