pyzlog Documentation

Release 0.1.3

zeeto.io

January 07, 2016

Contents

1 pyzlog **3**
1.1 Features . . . 3

2 Installation **5**

3 Usage **7**

4 Contributing **9**
4.1 Types of Contributions . . . 9
4.2 Get Started! . . . 10
4.3 Pull Request Guidelines . . . 10
4.4 Tips . . . 11

5 Credits **13**
5.1 Development Lead . . . 13
5.2 Contributors . . . 13

6 History **15**

7 0.1.0 (2015-10-13) **17**

8 0.1.1 (2015-10-30) **19**

9 0.1.2 (2016-01-07) **21**

10 0.1.3 (2016-01-07) **23**

11 Indices and tables **25**

Contents:

CHAPTER 1

pyzlog

Simple python module to standardize logging for applications

- Free software: ISC license
- Documentation: https://pyzlog.readthedocs.org.

1.1 Features

- Standard json logging
- Support for custom fields property
- Log file rotation at specified file size
- Configurable number of backup log files

CHAPTER 2

Installation

At the command line:

```
$ easy_install pyzlog
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pyzlog
$ pip install pyzlog
```

CHAPTER 3

Usage

To use pyzlog in a project:

```
import pyzlog

# initialize the zlogger
pyzlog.init_logs(
    path='/var/log',
    target='foo_app.log',
    level=logging.DEBUG,
    server_hostname='foo.app.io',
    fields={'custom_1': None, 'custom_2': None})

# then, log something
pyzlog.info(extra={'custom_1': 42, 'custom_2': 'foo'})

# log an error
try:
    raise_value_error()
except ValueError:
    pyzlog.error(extra={'custom_1': 'oh noes'})
```

To write tests for an application using pyzlog:

```
import os
import json
import app
import pyzlog
import unittest2


class TestApp(unittest2.TestCase, pyzlog.LogTest):

    def setUp(self):
        self.path = os.path.abspath('.')
        self.target = 'foo.log'
        self.remove_log()
        self.init_logs()

    def tearDown(self):
        self.remove_log()

    def test_log(self):
        app.something_that_logs()
```

```
        events = self.get_log_messages()
        self.assertEqual(1, len(events))
        expected_event = {
            'event_name': 'foo.event',
            # ...
        }
        self.assertEqual(expected_event, json.loads(events[0]))
```

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/zeeto/pyzlog/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" 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

pyzlog could always use more documentation, whether as part of the official pyzlog 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/zeeto/pyzlog/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 *pyzlog* for local development.

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

```
$ git clone git@github.com:your_name_here/pyzlog.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 pyzlog
$ cd pyzlog/
$ pip install -r requirements_dev.txt
```

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:

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

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 2.6, and 2.7. Check https://travis-ci.org/zeeto/pyzlog/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_pyzlog
```

CHAPTER 5

Credits

5.1 Development Lead

- Matt Maybeno <m.maybeno@zeeto.io>
- Herb Johnson <herb@zeeto.io>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

0.1.0 (2015-10-13)

- First release on PyPI.

CHAPTER 8

0.1.1 (2015-10-30)

- Added a class to help make testing easier.

CHAPTER 9

0.1.2 (2016-01-07)

- Properly resolve hostname in log messages.

CHAPTER 10

0.1.3 (2016-01-07)

- Fallback to hostname resolution in init_logs

CHAPTER 11

Indices and tables

- genindex
- modindex
- search