gitEasyChangelog Documentation

Release 0.2.0

Eric Horvat

Aug 26, 2020

Contents:

CHAPTER 1

Readme file

CHAPTER 2

Git Easy Changelog

This project helps at managing yours REALEASE/CHANGELOG `.md` files, avoiding git conflicts as it use one file per change reported in the final release file.

2.1 Usage

2.1.1 Basic Usage

1. Install by `pip install giteasychangelog`.
2. Create a `CHANGELOG` folder.
3. Create a folder for the next version.
4. Add a new `.md` single line file for each change that should be reported.
5. Run `giteasychangelog`
6. Go back to step 3.

2.1.2 Advanced options

- You can change previous version files, up to error, reference of newest changes or anything you want.
- You can add a `date.md` file in any version file, which will be add as release date reference.

Example

You can test the project in `example` branch, by simple running `giteasychangelog`

Credits

This package was created with Cookiecutter* and the `audreyr/cookiecutter-pypackage`* project template.

Cookiecutter audreyr/cookiecutter-pypackage

CHAPTER 3

Installation

3.1 Stable release

To install gitEasyChangelog, run this command in your terminal:

```
$ pip install giteasychangelog
```

This is the preferred method to install gitEasyChangelog, 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.

3.2 From sources

The sources for gitEasyChangelog can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/EricHorvat/giteasychangelog
```

Or download the tarball:

```
$ curl  -OL https://github.com/EricHorvat/giteasychangelog/tarball/master
```

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

```
$ python setup.py install
```

CHAPTER 4

Usage

To use gitEasyChangelog in a project:

```
import giteasychangelog
```

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/EricHorvat/giteasychangelog/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

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

5.2 Get Started!

Ready to contribute? Here's how to set up `giteasychangelog` for local development.

#. Fork the `giteasychangelog` repo on GitHub. #.

   Clone your fork locally:

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

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

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

2. Create a branch for local development:

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

   Now you can make your changes locally.

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

```
$ flake8 giteasychangelog tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

4. 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
```

5. Submit a pull request through the GitHub website.

5.3 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.md.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/EricHorvat/giteasychangelog/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_giteasychangelog
```

5.5 Deploying

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

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

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Eric Horvat eric.nahuel.horvat@gmail.com

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

New features in the latest update

7.1 0.2.0 [Aug 26th, 2020]:

- Internal improves of testing/release
- Ignore last n from each changelog file

7.2 0.1.0 [Feb 18th, 2019]:

- First release of the tool!
- Usable as a cli command!
- Function as describe in README!

CHAPTER 8

Indices and tables

- genindex
- modindex
- search