Extremely Ease Event Emitter Documentation

Release 0.1.1

Paweł Zadrożny

Mar 19, 2018

Contents:

CHAPTER 1

Extremely Easy Event Emitter

Info Extremely Easy Event Emitter.

Author Paweł Zadrożny @pawelzny <pawel.zny@gmail.com>

1.1 Features

- Asynchronous Event emitter based on asyncio
- Subscribe any callable handler
- Filter events by Publisher
- Easy enable-disable events on runtime
- Subscribe handlers using decorator

1.2 Installation

```
pip install eeee
```

Package: https://pypi.org/project/eeee/

1.3 Documentation

Read full documentation at http://eeee.readthedocs.io/en/stable/

1.4 Quick Example

```
from eeee import Event, Publisher

my_event = Event('MyEvent')

# Subscribe takes publisher instance or name as optional argument.
# If publisher is defined handler will be triggered only when that
# particular publisher send a message.
# Leave empty to listen to all publishers within this event.
@my_event.subscribe()
async def custom_handler(message, publisher, event):
    print(message, publisher, event)

result = await my_event.publish('New message arrived!', Publisher('global'))
```

CHAPTER 2

Public API

2.1 Publish/Subscribe

2.1.1 Event

2.1.2 Publisher

2.1.3 Subscriber

2.2 Exceptions

2.3 Utilities

2.3.1 Subscribe decorator

2.3.2 Asyncio context manager

Provided by Context Loop can be imported from *eeee* as well. Context loop allow to execute asynchronous code ad-hoc.

```
import eeee import Loop

>>> my_event = Event('MyEvent')
>>> with Loop(my_event.publish({'message': 'secret'})) as loop:
...   result = loop.run_until_complete()
...
```

CHAPTER 3

Credits

3.1 Development

- Paweł Zadrożny @pawelzny <pawel.zny@gmail.com>

3.2 Contributors

None yet. Why not be the first?

Read more how to contribute on *Contributing*.

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/pawelzny/eeee/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

authentication could always use more documentation, whether as part of the official authentication 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/pawelzny/eeee/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 *eeee* for local development.

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

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

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

```
$ cd eeee/
$ make install-dev
```

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:

```
$ make test-all
```

   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.
3. The pull request should work for Python 3.4, 3.5, and 3.6, and for PyPy3. Check https://circleci.com/gh/pawelzny/eeee and make sure that the tests pass for all supported Python versions.

CHAPTER 5

LICENSE

ISC License

Copyright (c) 2017, Paweł Zadrożny @pawelzny <pawel.zny@gmail.com>

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Python Module Index

e

Index

E