pyoneering

Release 0.1.0

Oct 15, 2018

Contents

CHAPTER 1

Overview

docs	
tests	
package	

Decorators for deprecating and refactoring

- Free software: Apache Software License 2.0

1.1 Installation

```
pip install pyoneering
```

1.2 Documentation

https://python-pyoneering.readthedocs.io/

1.3 Development

To run the all tests run:

```
tox
```

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER 2

Installation

You can install pyoneering with `pip` or `pipenv`.

- `pip`

```
$ pip install pyoneering
```

- `pipenv`

```
$ pip install pyoneering
```

CHAPTER 3

Getting started

3.1 Configuration

In order to provide a module-wide configuration for the decorators `deprecated()` and `refactored()` include the following code-snippet in your module.

```
from tests.example import __version__

from pyoneering import DevUtils

_module = DevUtils(__version__)
deprecated, refactored = _module.deprecated, _module.refactored
```

Then these functions are accessible from anywhere in your module.

```
```

3.2 Examples

The examples are generated with `__version__='1.0'`.

3.2.1 deprecated class

```
@deprecated('0.6', '1.8')
class DeprecatedClass:
    """Example of a deprecated class."""
    pass
```

3.2.2 deprecated method

```
@deprecated('0.2', '0.8', details='Use new method instead')
def deprecated_method():
    """Example of a deprecated method."""
    pass
```

3.2.3 renamed parameter

```
@refactored('0.4', '2.0', parameter_map={'old_kwarg2': 'new_kwarg1'})
def renamed_parameter(kwarg1=5, new_kwarg1=False):
    """Example of a method with changed signature."""
    pass
```

3.2.4 merged parameter

```
def _merged_parameters(old_kwarg2=True, old_kwarg3=False):
    if not old_kwarg2 and not old_kwarg3:
        new_kwarg1 = 'cat-1'
    elif old_kwarg2 and not old_kwarg3:
        new_kwarg1 = 'cat-2'
    else:
        new_kwarg1 = 'error'

    return dict(new_kwarg1=new_kwarg1)
```

```
@refactored('0.4', parameter_map=_merged_parameters)
def merged_parameter(kwarg1=5, new_kwarg1='error'):
    """Example of a method with changed signature."""
    pass
```

CHAPTER 4

Reference

4.1 pyoneering

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When 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.

5.2 Documentation improvements

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

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/FHaase/python-pyoneering/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 code contributions are welcome :)

5.4 Development

To set up *python-pyoneering* for local development:

1. Fork python-pyoneering (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/python-pyoneering.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- pytest -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 6

Authors

- Fabian Haase - https://blog.ionelmc.ro

CHAPTER 7

Changelog

7.1 0.1.0 (2018-10-12)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search