PyMaybe Documentation

Release 0.2.0

Eran Kampf

Oct 30, 2017

Contents

Contents:

CHAPTER 1

PyMaybe

A Python implementation of the Maybe pattern.

Installation

```
pip install pymaybe
```

Getting Started

```
from pymaybe import maybe
first_name = maybe(deep_hash)['account']['user_profile']['first_name'].or_else("
↪<unknown>")
```

Documentation

Maybe monad is a programming pattern that allows to treat None values that same way as non-none values. This is done by wrapping the value, which may or may not be None to, a wrapper class.

The implementation includes two classes: *Maybe* and *Something*. *Something* represents a value while *Nothing* represents a None value. There's also a method *maybe* which wraps a regular value and and returns *Something* or *Nothing* instance.

```
>>> maybe("I'm a value")
"I'm a value"

>>> maybe(None);
None
```

Both *Something* and *Nothing* implement 4 methods allowing you to test their real value: *is_some*, *is_none*, *get* and *or_else*

```
>>> maybe("I'm a value").is_some()
True

>>> maybe("I'm a value").is_none()
False

>>> maybe(None).is_some()
False

>>> maybe(None).is_none()
True

>>> maybe("I'm a value").get()
"I'm a value"

>>> maybe("I'm a value").or_else(lambda: "No value")
"I'm a value"

>>> maybe(None).get()
Traceback (most recent call last):
...
Exception: No such element

>>> maybe(None).or_else(lambda: "value")
'value'

>>> maybe(None).or_else("value")
'value'
```

In addition, *Something* and *Nothing* implement the Python magic methods allowing you to treat them as dictionaries:

```
>>> nested_dict = maybe(nested_dict)
>>> nested_dict['store']['name']
'MyStore'

>>> nested_dict['store']['address']
None

>>> nested_dict['store']['address']['street'].or_else('No Address Specified')
'No Address Specified'
```

All other method calls on *Something* are forwarded to its real *value*:

```
>>> maybe('VALUE').lower()
'value'

>>> maybe(None).invalid().method().or_else('unknwon')
'unknwon'
```

Examples & Use Cases

The Maybe pattern helps you avoid nasty try..except blocks. Consider the following code:

```
try:
    url = rss.load_feeds()[0].url.domain
except (TypeError, IndexError, KeyError, AttributeError):
    url = "planetpython.org"
```

With Maybe you could simply do:

```
url = maybe(rss).load_feeds()[0]['url'].domain.or_else("planetpython.org")
```

Getting the current logged in user's name. Without maybe:

```
def get_user_zipcode():
    address = getattr(request.user, 'address', None)
    if address:
        return getattr(address, 'zipcode', '')

    return ''
```

With maybe:

```
def get_user_zipcode():
    return maybe(request.user).address.zipcode.or_else('')
```

Further Reading

- Option (Scala)
- Maybe (Java)
- Maybe pattern (Python recipe)
- Data.Maybe (Haskell)
- Maybe (Ruby)

Copyright and License

Copyright 2015 - Eran Kampf

- Free software: BSD license
- Documentation: https://pymaybe.readthedocs.org.
- Code is hosted on GitHub

CHAPTER 2

Installation

At the command line:

```
$ easy_install pymaybe
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pymaybe
$ pip install pymaybe
```

CHAPTER 3

Usage

To use PyMaybe in a project:

```
import pymaybe
```

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/ekampf/pymaybe/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.

Fix Bugs

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

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

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

Submit Feedback

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

Get Started!

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

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

   ```
   $ git clone git@github.com:your_name_here/pymaybe.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 pymaybe
   $ cd pymaybe/
   $ 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 pymaybe tests
   $ python setup.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.

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, and 3.4, and for PyPy. Check https://travis-ci.org/ekampf/pymaybe/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

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

CHAPTER 5

Credits

- Eran Kampf - http://www.developerzen.com

Contributors

None yet. Why not be the first?

CHAPTER 6

History

CHAPTER 7

0.1.0 (2015-01-11)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search