lazy-object-proxy

Release 1.9.1.dev11+gaa46e84

Ionel Cristian Mărieș

Dec 15, 2023

CONTENTS

CHAPTER

ONE

OVERVIEW

docs	
tests	
package	

A fast and thorough lazy object proxy.

- Free software: BSD 2-Clause License

Note that this is based on wrapt's ObjectProxy with one big change: it calls a function the first time the proxy object is used, while *wrapt.ObjectProxy* just forwards the method calls to the target object.

In other words, you use *lazy-object-proxy* when you only have the object way later and you use *wrapt.ObjectProxy* when you want to override few methods (by subclassing) and forward everything else to the target object.

Example:

```
import lazy_object_proxy

def expensive_func():
    from time import sleep
    print('starting calculation')
    # just as example for a very slow computation
    sleep(2)
    print('finished calculation')
    # return the result of the calculation
    return 10

obj = lazy_object_proxy.Proxy(expensive_func)
# function is called only when object is actually used
print(obj)  # now expensive_func is called

print(obj)  # the result without calling the expensive_func
```

1.1 Installation

```
pip install lazy-object-proxy
```

1.2 Documentation

https://python-lazy-object-proxy.readthedocs.io/

1.3 Development

To run all the tests run:

```
tox
```

1.4 Acknowledgements

This project is based on some code from wrapt as you can see in the git history.

CHAPTER

TWO

INSTALLATION

At the command line:

```
pip install lazy-object-proxy
```

CHAPTER

THREE

USAGE

To use lazy-object-proxy in a project:

```
import lazy_object_proxy
```

CHAPTER

FOUR

CONTRIBUTING

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

4.1 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.

4.2 Documentation improvements

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

4.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/ionelmc/python-lazy-object-proxy/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 :)

4.4 Development

To set up *python-lazy-object-proxy* for local development:

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

   ```
   git clone git@github.com:YOURGITHUBNAME/python-lazy-object-proxy.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 and docs builder with 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.

4.4.1 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`).
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`.

4.4.2 Tips

To run a subset of tests:

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

To run all the test environments in *parallel*:

```
tox -p auto
```

CHAPTER

FIVE

AUTHORS

- Ionel Cristian Mărieș - https://blog.ionelmc.ro
- Alvin Chow - https://github.com/alvinchow86
- Astrum Kuo - https://github.com/xowenx
- Erik M. Bray - https://github.com/embray
- Ran Benita - https://github.com/bluetech
- “hugovk” - https://github.com/hugovk
- Sandro Tosi - https://github.com/sandrotosi

CHAPTER

SIX

CHANGELOG

6.1 1.10.0 (2023-12-15)

- Added Python 3.12 wheels.
- Dropped support for Python 3.7.
- Applied some reformatting and lint fixes using ruff to the codebase (mostly more Python 2 leftover cleanups).

6.2 1.9.0 (2023-01-04)

- Added support for matrix multiplication operator (`@`).
- Should have all the wheels now (including the manylinux ones).
- Bumped minimum version requirements for setuptools and setuptools-scm.
- Switched the default pure python fallback implementation to the "simple" one (when you `from lazy_object_proxy import Proxy` and the C extension is not available). Previously the "slots" implementation was used but as it turns out it is slower on Python 3.

6.3 1.8.0 (2022-10-26)

- Cleaned up use of cPickle. Contributed by Sandro Tosi in #62.
- Cleaned up more dead Python 2 code.
- Added Python 3.11 wheels.
- Dropped support for Python 3.6.

6.4 1.7.1 (2021-12-15)

- Removed most of the Python 2 support code and fixed `python_requires` to require at least Python 3.6.

  Note that 1.7.0 has been yanked because it could not install on Python 2.7. Installing lazy-object-proxy on Python 2.7 should automatically fall back to the 1.6.0 release now.

6.5 1.7.0 (2021-12-15)

- Switched CI to GitHub Actions, this has a couple consequences:
  - Support for Python 2.7 is dropped. You can still install it there but it's not tested anymore and Python 2 specific handling will be removed at some point.
  - Linux wheels are now provided in *musllinux* and *manylinux2014* variants.
- Fixed `__index__` to fallback to `int` if the wrapped object doesn't have an `__index__` method. This prevents situations where code using a proxy would otherwise likely just call `int` had the object not have an `__index__` method.

6.6 1.6.0 (2021-03-22)

- Added support for async special methods (`__aiter__`, `__anext__`, `__await__`, `__aenter__`, `__aexit__`). These are used in the `async for`, `await` and ``async with` statements.

  Note that `__await__` returns a wrapper that tries to emulate the crazy stuff going on in the ceval loop, so there will be a small performance overhead.
- Added the `__resolved__` property. You can use it to check if the factory has been called.

6.7 1.5.2 (2020-11-26)

- Added Python 3.9 wheels.
- Removed Python 2.7 Windows wheels (not supported on newest image with Python 3.9).

6.8 1.5.1 (2020-07-22)

- Added ARM64 wheels (manylinux2014).

6.9 1.5.0 (2020-06-05)

- Added support for `__fspath__`.
- Dropped support for Python 3.4.

6.10 1.4.3 (2019-10-26)

- Added binary wheels for Python 3.8.
- Fixed license metadata.

6.11 1.4.2 (2019-08-22)

- Included a `pyproject.toml` to allow users install the sdist with old python/setuptools, as the setuptools-scm dep will be fetched by pip instead of setuptools. Fixes #30.

6.12 1.4.1 (2019-05-10)

- Fixed wheels being built with `-coverage` cflags. No more issues about bogus `cext.gcda` files.
- Removed useless C file from wheels.
- Changed `setup.py` to use setuptools-scm.

6.13 1.4.0 (2019-05-05)

- Fixed `__mod__` for the slots backend. Contributed by Ran Benita in #28.
- Dropped support for Python 2.6 and 3.3. Contributed by "hugovk" in #24.

6.14 1.3.1 (2017-05-05)

- Fix broken release (`sdist` had a broken `MANIFEST.in`).

6.15 1.3.0 (2017-05-02)

- Speed up arithmetic operations involving `cext.Proxy` subclasses.

6.16 1.2.2 (2016-04-14)

- Added manylinux wheels.
- Minor cleanup in readme.

6.17 1.2.1 (2015-08-18)

- Fix a memory leak (the wrapped object would get bogus references). Contributed by Astrum Kuo in #10.

6.18 1.2.0 (2015-07-06)

- Don't instantiate the object when __repr__ is called. This aids with debugging (allows one to see exactly in what state the proxy is).

6.19 1.1.0 (2015-07-05)

- Added support for pickling. The pickled value is going to be the wrapped object *without* any Proxy container.
- Fixed a memory management issue in the C extension (reference cycles weren't garbage collected due to improper handling in the C extension). Contributed by Alvin Chow in #8.

6.20 1.0.2 (2015-04-11)

- First release on PyPI.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search