pytest-object-getter

Release 0.0.1

Konstantinos Lampridis

Jul 31, 2022

CONTENTS:

1 PYTEST OBJECT GETTER **1**

2 Highlights **3**

3 Quickstart **5**
- 3.1 Prerequisites . . . 5
- 3.2 Installing . . . 5
- 3.3 A Use Case . . . 5
- 3.4 License . . . 6

4 Development **7**
- 4.1 Introduction . . . 7
- 4.2 Why this Package? . . . 8
- 4.3 Usage . . . 8
- 4.4 pytest_object_getter . . . 9

5 Indices and tables **11**

Python Module Index **13**

Index **15**

CHAPTER

ONE

PYTEST OBJECT GETTER

A Pytest Plugin providing the *get_object* fixture.

Code: https://github.com/boromir674/pytest-object-getter
Docs: https://pytest-object-getter.readthedocs.io/en/master/
PyPI: https://pypi.org/project/pytest-object-getter/
CI: https://github.com/boromir674/pytest-object-getter/actions/

CHAPTER

TWO

HIGHLIGHTS

1. **pytest_object_getter** *python package*, hosted on pypi.org
    - Installable with *pip*
    - *get_object* fixture available to your tests
        1. Dynamically import an object from a module
        2. Optionally mock any object that is present in the module's namespace
        3. Construct the mock object at runtime
        4. Alter the bahaviour of an object at runtime
2. Tested against multiple *platforms* and *python* versions
    - platforms: Ubuntu, MacOS
    - python: 3.6, 3.7, 3.8, 3.9, 3.10

    For more, see the CI Pipeline and the *Test* workflow, defined in test.yaml.

You can read more on pytest and fixtures in pytest latest documentation.

CHAPTER

THREE

QUICKSTART

3.1 Prerequisites

You need to have *Python* installed.

3.2 Installing

Using *pip* is the approved way for installing *pytest_object_getter*.

```
python3 -m pip install pytest_object_getter
```

After installation the *get_object* pytest fixture should be available in your tests.

3.3 A Use Case

Let's see how to write a test and use the 'get_object' fixture to mock the *requests.get* method to avoid actual network communication:

```
python3 -m pip install ask-pypi
```

```
import pytest

@pytest.fixture
def mock_response():
    def init(self, package_name: str):
        self.status_code = 200 if package_name == 'existing-package' else 404
    return type('MockResponse', (), {
        '__init__': init
    })

@pytest.fixture
def create_mock_requests(mock_response):
    def _create_mock_requests():
        def mock_get(*args, **kwargs):
            package_name = args[0].split('/')[-1]
            return mock_response(package_name)
        return type('MockRequests', (), {
```

(continues on next page)

(continued from previous page)

```
            'get': mock_get,
        })
    return _create_mock_requests

def test_fixture(get_object, create_mock_requests):

    from ask_pypi import is_pypi_project

    assert is_pypi_project('numpy') == True
    assert is_pypi_project('pandas') == True
    assert is_pypi_project('existing-package') == False

    get_object('is_project', 'ask_pypi.pypi_project',
        overrides={'requests': lambda: create_mock_requests()})

    assert is_pypi_project('existing-package') == True

    assert is_pypi_project('numpy') == False
    assert is_pypi_project('pandas') == False
    assert is_pypi_project('so-magic') == False
```

3.4 License

Free software:

- GNU Affero General Public License v3.0

CHAPTER

FOUR

DEVELOPMENT

Here are some useful notes related to doing development on this project.

1. **Test Suite**, using pytest, located in *tests* dir
2. **Parallel Execution** of Unit Tests, on multiple cpu's
3. **Documentation Pages**, hosted on *readthedocs* server, located in *docs* dir
4. **Automation**, using tox, driven by single *tox.ini* file
    a. **Code Coverage** measuring
    b. **Build Command**, using the build python package
    c. **Pypi Deploy Command**, supporting upload to both pypi.org and test.pypi.org servers
    d. **Type Check Command**, using mypy
    e. **Lint** *Check* and *Apply* commands, using isort and black
5. **CI Pipeline**, running on Github Actions, defined in *.github/*
    a. **Job Matrix**, spanning different *platform*'s and *python version*'s
        1. Platforms: *ubuntu-latest*, *macos-latest*
        2. Python Interpreters: *3.6*, *3.7*, *3.8*, *3.9*, *3.10*
    b. **Parallel Job** execution, generated from the *matrix*, that runs the *Test Suite*

4.1 Introduction

This is **Pytest Object Getter**, a *Python Package* desinged to ...

Goal of this project is to facilitate easier mocking when writing test cases.

It provides with infrastructure (as pytest fixtures) to simplify writing a test case that requires one or more (python) objects to be mocked

This documentation aims to help people understand what are the package's features and to demonstrate how to leverage them for their use cases.

4.2 Why this Package?

So, why would one opt for this Package?

It is **easy** to *install* (using pip) and intuitive to *use*.

Pytest Object Getter features the 'get_object' pytest fixture that arguably simplifies the process of mocking an object present in a namespace.

Well-tested against multiple Python Interpreter versions (3.6 - 3.10), tested on both *Linux* (Ubuntu) and *Darwin* (Macos) platforms.

Tests trigger automatically on **CI**. The package's releases follow **Semantic Versioning**.

4.3 Usage

4.3.1 Installation

pytest_object_getter is available on PyPI hence you can use *pip* to install it.

It is recommended to perform the installation in an isolated *python virtual environment* (env). You can create and activate an *env* using any tool of your preference (ie *virtualenv*, *venv*, *pyenv*).

Assuming you have 'activated' a *python virtual environment*:

```
python -m pip install pytest-object-getter
```

4.3.2 Simple Use Case

Common Use Case for the pytest_object_getter is to use the 'get_obejct' fixture
to mock an object in your python test case (using pytest).

Let's see a test that mocks the *requests.get* method to avoid actual network communication:

Install Python dependencies:

```
python3 -m pip install ask-pypi
```

Test case:

```
import pytest

@pytest.fixture
def mock_response():
    def init(self, package_name: str):
        self.status_code = 200 if package_name == 'existing-package' else 404
    return type('MockResponse', (), {
        '__init__': init
    })

@pytest.fixture
```

(continues on next page)

(continued from previous page)

```
def create_mock_requests(mock_response):
    def _create_mock_requests():
        def mock_get(*args, **kwargs):
            package_name = args[0].split('/')[-1]
            return mock_response(package_name)
        return type('MockRequests', (), {
            'get': mock_get,
        })
    return _create_mock_requests

def test_fixture(get_object, create_mock_requests):

    from ask_pypi import is_pypi_project

    assert is_pypi_project('numpy') == True
    assert is_pypi_project('pandas') == True
    assert is_pypi_project('existing-package') == False

    get_object('is_project', 'ask_pypi.pypi_project',
        overrides={'requests': lambda: create_mock_requests()})

    assert is_pypi_project('existing-package') == True

    assert is_pypi_project('numpy') == False
    assert is_pypi_project('pandas') == False
    assert is_pypi_project('so-magic') == False
```

4.4 pytest_object_getter

4.4.1 pytest_object_getter package

Module contents

pytest_object_getter.**attribute_getter**()

pytest_object_getter.**generic_object_getter_class**(*attribute_getter*, *monkeypatch*)

Class instances can extract a requested object from within a module and optionally patch any object in the module's namespace at runtime.

pytest_object_getter.**get_object**(*object_getter_class*)

Import an object from a module and optionally mock any object in its namespace.

A callable that can import an object, given a reference (str), from a module , given its "path" (string represented as 'dotted' modules: same way python code imports modules), and provide the capability to monkeypatch/mock any object found in the module's namespace at runtime.

The client code must supply the first 2 arguments at runtime, correspoding to the object's symbol name (str) and module "path" (str).

The client code can optionally use the 'overrides' kwarg to supply a python dictionary to specify what runtime objects to mock and how.

Each dictionary entry should model your intention to monkeypatch one of the module namespace' objects with a custom 'mock' value.

Each dictionary key should be a string corresponding to an object's reference name (present in the module's namespace) and each value should be a callable that can construct the 'mock' value. The callable should take no arguments and acts as a "factory", that when called should provide the 'mock' value.

Example

def mocked_request_get() business_method = get_object(

"business_method", "business_package.methods", overrides={"production": lambda: 'mocked'}

)

Parameters

- **symbol** (*str*) – the object's reference name
- **module** (*str*) – the module 'path' represented as module names "joined" by "." (dots)
- **overrides** (*dict, optional*) – declare what to monkeypatch and with what "mocks". Defaults to None.

Returns the object imported from the module with its namespace potentially mocked

Return type Any

pytest_object_getter.**object_getter_class**(*generic_object_getter_class*)

Do a dynamic import of a module and get an object from its namespace.

This fixture returns a Python Class that can do a dynamic import of a module and get an object from its namespace.

Instances of this class are callable's (they implement the __call__ protocol) and uppon calling the return a reference to the object "fetched" from the namespace.

Callable instances arguments: * 1st: object with the 'symbol_namel': str and 'object_module_string': str

attributes expected "on it"

Returns Class that can do a dynamic import and get an object

Return type ObjectGetter

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p
pytest_object_getter, 9

INDEX

A

attribute_getter() (*in module pytest_object_getter*), 9

G

generic_object_getter_class() (*in module pytest_object_getter*), 9
get_object() (*in module pytest_object_getter*), 9

M

module
    pytest_object_getter, 9

O

object_getter_class() (*in module pytest_object_getter*), 10

P

pytest_object_getter
    module, 9