requests-staticmock Documentation

Release 1.4.0

Anthony Shaw

Jun 04, 2020

Contents

A static HTTP mock interface for testing classes that leverage Python *requests* with **no** monkey patching!

- Free software: Apache 2 License
- Documentation: https://requests-staticmock.readthedocs.org.

CHAPTER 1

Usage

1.1 As a context manager for requests Session instances

requests_staticmock.context.**mock_session_with_fixtures**(**args*, ***kwds*)

Context Manager

Mock the responses with a particular session to any files found within a static path

Parameters

- **session** (requests.Session) – The requests session object
- **path** (str) – The path to the fixtures
- **url** (str or list) – The base URL to mock, e.g. http://mock.com, http:// supports a single URL or a list

1.1.1 Example

```
import requests
import requests_staticmock

session = requests.Session()
with requests_staticmock.mock_session_with_fixtures(session, 'tests/fixtures', 'http:/
↪/test_context.com'):
    # will return a response object with the contents of tests/fixtures/test.json
    response = new_session.request('get', 'http://test_context.com/test.json')
```

1.2 As an adapter

You can inject the *requests_staticmock* adapter into an existing (or new) requests session to mock out a particular URL or domain, e.g.

```
import requests
from requests_staticmock import Adapter

session = requests.Session()
special_adapter = Adapter('fixtures')
session.mount('http://specialwebsite.com', special_adapter)
session.request('http://normal.com/api/example') # works as normal
session.request('http://specialwebsite.com') # returns static mocks
```

1.3 Class adapter

Instead of using a static asset adapter, you can use an adapter that expects an internal method to respond with a string, e.g.

GET */test/example.xml* will call method *_test_example_xml(self, request)*

GET */test/example.xml?query=param* will call method *_test_example_xml(self, request)*

This can be used via *requests_staticmock.ClassAdapter* or the context manager

requests_staticmock.context.**mock_session_with_class**(**args*, ***kwds*)

Context Manager

Mock the responses with a particular session to any private methods for the URLs

Parameters

- **session** (requests.Session) – The requests session object
- **cls** (object) – The class instance with private methods for URLs
- **url** (str or list) – The base URL to mock, e.g. http://mock.com, http:// supports a single URL or a list

1.3.1 Example

```
import requests
import requests_staticmock


class MyTestClass(requests_staticmock.BaseMockClass):
    def _api_v1_idea(self, request):
        return "woop woop"

session = requests.Session()
with requests_staticmock.mock_session_with_class(session, MyTestClass, 'http://test_
↪context.com'):
    # will return a response object with the contents 'woop woop'
    response = new_session.request('get', 'http://test_context.com/api/v1/idea')
```

1.4 Class adapter with unpacked requests

The class adapter supports unpacking of the following components, just add these keyword arguments to your callback methods and the class adapter will match them to the arguments.

- *method* - The HTTP verb, e.g. GET
- *url* - The full URL
- *params* - The dict with the request parameters
- *headers* - The request headers
- *body* - The request body text

```
import requests
import requests_staticmock

class_session = Session()
class TestMockClass(BaseMockClass):
    def _api_v1_idea(self, method, params, headers):
        if params['special'] == 'value':
            return 'yes'
    def _api_v1_brillo(self, url, body):
        if json.loads(body)['special'] == 'value':
            return 'yes'

a = ClassAdapter(TestMockClass)

session = requests.Session()
with requests_staticmock.mock_session_with_class(session, MyTestClass, 'http://test_
↪context.com'):
    response = new_session.request('get', 'http://test_context.com/api/v1/idea')
```

See StaticResponseFactory for simple ways of returning good and bad responses.

class requests_staticmock.responses.**StaticResponseFactory**

Static factory for producing internal instances of *requests* Response objects

static BadResponse(*body*, *request*, *status_code=None*, *headers=None*)

Construct a Bad HTTP response (defined in DEFAULT_BAD_RESPONSE_CODE)

Parameters

- **body** (str) – The body of the response
- **request** (requests.Request) – The HTTP request
- **status_code** (int) – The return status code, defaults to DEFAULT_GOOD_STATUS_CODE if not specified
- **headers** (dict) – Response headers, defaults to DEFAULT_RESPONSE_HEADERS if not specified

Return type requests.Response

Returns a Response object

static GoodResponse(*body*, *request*, *status_code=None*, *headers=None*)

Construct a Good HTTP response (defined in DEFAULT_GOOD_RESPONSE_CODE)

Parameters

- **body** (str) – The body of the response
- **request** (requests.Request) – The HTTP request
- **status_code** (int) – The return status code, defaults to DEFAULT_GOOD_STATUS_CODE if not specified

- **headers** (dict) – Response headers, defaults to DEFAULT_RESPONSE_HEADERS if not specified

Return type requests.Response

Returns a Response object

CHAPTER 2

Features

- Allow mocking of HTTP responses via a directory of static fixtures
- Support for sub-directories matching URL paths

CHAPTER 3

Credits

This project takes inspiration and ideas from the *requests_mock* package, maintained by the OpenStack foundation. I redesigned this based on the abstractions within the requests project instead of using the patching pattern used in *requests_mock*. I find the responses more native, easier to work with and also the ability to load static files much easier.

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

Contents:

3.1 API Reference

3.1.1 Adapter Module

class requests_staticmock.adapter.**Adapter**(*base_path=None*)

A replacement session adapter that responds with the content of static files matching the path of the requested URL

close()

Cleans up adapter specific items.

match_url(*request*)

Match the request against a file in the adapter directory

Parameters **request** (requests.Request) – The request

Returns Path to the file

Return type str

register_path(*path*)

Register a new search path

Parameters **path** (str) – The new search path

response_from_fixture(*request*, *fixture_path*)

send(*request*, ***kwargs*)
Sends PreparedRequest object. Returns Response object.

Parameters

- **request** – The `PreparedRequest` being sent.
- **stream** – (optional) Whether to stream the request content.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a (connect timeout, read timeout) tuple.
- **verify** – (optional) Either a boolean, in which case it controls whether we verify the server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use
- **cert** – (optional) Any user-provided SSL certificate to be trusted.
- **proxies** – (optional) The proxies dictionary to apply to the request.

class requests_staticmock.adapter.**ClassAdapter**(*cls*)
A requests Adapter for a class that has methods matching the URLS, e.g. *def _api_v1_test()* would be called for session.get('api/v1/test')

send(*request*, ***kwargs*)
Sends PreparedRequest object. Returns Response object.

Parameters

- **request** – The `PreparedRequest` being sent.
- **stream** – (optional) Whether to stream the request content.
- **timeout** (*float or tuple*) – (optional) How long to wait for the server to send data before giving up, as a float, or a (connect timeout, read timeout) tuple.
- **verify** – (optional) Either a boolean, in which case it controls whether we verify the server's TLS certificate, or a string, in which case it must be a path to a CA bundle to use
- **cert** – (optional) Any user-provided SSL certificate to be trusted.
- **proxies** – (optional) The proxies dictionary to apply to the request.

3.1.2 Context Managers

requests_staticmock.context.**mock_session_with_class**(**args*, ***kwds*)
Context Manager

Mock the responses with a particular session to any private methods for the URLs

Parameters

- **session** (`requests.Session`) – The requests session object
- **cls** (`object`) – The class instance with private methods for URLs
- **url** (`str` or `list`) – The base URL to mock, e.g. http://mock.com, http:// supports a single URL or a list

requests_staticmock.context.**mock_session_with_fixtures**(**args*, ***kwds*)
Context Manager

Mock the responses with a particular session to any files found within a static path

Parameters

- **session** (`requests.Session`) – The requests session object

- **path** (str) – The path to the fixtures
- **url** (str or list) – The base URL to mock, e.g. http://mock.com, http:// supports a single URL or a list

3.1.3 Response Factory

class requests_staticmock.responses.**StaticResponseFactory**

Static factory for producing internal instances of *requests* Response objects

static BadResponse(*body*, *request*, *status_code=None*, *headers=None*)

Construct a Bad HTTP response (defined in DEFAULT_BAD_RESPONSE_CODE)

Parameters

- **body** (str) – The body of the response
- **request** (requests.Request) – The HTTP request
- **status_code** (int) – The return status code, defaults to DEFAULT_GOOD_STATUS_CODE if not specified
- **headers** (dict) – Response headers, defaults to DEFAULT_RESPONSE_HEADERS if not specified

Return type requests.Response

Returns a Response object

static GoodResponse(*body*, *request*, *status_code=None*, *headers=None*)

Construct a Good HTTP response (defined in DEFAULT_GOOD_RESPONSE_CODE)

Parameters

- **body** (str) – The body of the response
- **request** (requests.Request) – The HTTP request
- **status_code** (int) – The return status code, defaults to DEFAULT_GOOD_STATUS_CODE if not specified
- **headers** (dict) – Response headers, defaults to DEFAULT_RESPONSE_HEADERS if not specified

Return type requests.Response

Returns a Response object

3.2 Usage

3.2.1 As a context manager for requests Session instances

The *requests_staticmock*

```
import requests
import requests_staticmock

session = requests.Session()
with requests_staticmock.mock_session_with_fixtures(session, 'tests/fixtures', 'http:/
↪/test_context.com'):
```

(continues on next page)

(continued from previous page)

```
    # will return a response object with the contents of tests/fixtures/test.json
    response = new_session.request('get', 'http://test_context.com/test.json')
```

3.2.2 As an adapter

You can inject the *requests_staticmock* adapter into an existing (or new) requests session to mock out a particular URL or domain, e.g.

```
import requests
from requests_staticmock import Adapter

session = requests.Session()
special_adapter = Adapter('fixtures')
session.mount('http://specialwebsite.com', special_adapter)
session.request('http://normal.com/api/example') # works as normal
session.request('http://specialwebsite.com') # returns static mocks
```

3.2.3 Class adapter

Instead of using a static asset adapter, you can use an adapter that expects an internal method to respond with a string, e.g.

GET */test/example.xml* will call method *_test_example_xml(self, request)*

GET */test/example.xml?query=param* will call method *_test_example_xml(self, request)*

This can be used via *requests_staticmock.ClassAdapter* or the context manager

```
import requests
import requests_staticmock


class MyTestClass(requests_staticmock.BaseMockClass):
    def _api_v1_idea(self, request):
        return "woop woop"

session = requests.Session()
with requests_staticmock.mock_session_with_class(session, MyTestClass, 'http://test_
↪context.com'):
    # will return a response object with the contents 'woop woop'
    response = new_session.request('get', 'http://test_context.com/api/v1/idea')
```

3.2.4 Class adapter with unpacked requests

The class adapter supports unpacking of the following components, just add these keyword arguments to your callback methods and the class adapter will match them to the arguments.

- *method* - The HTTP verb, e.g. GET
- *url* - The full URL
- *params* - The dict with the request parameters
- *headers* - The request headers

- *body* - The request body text

```
import requests
import requests_staticmock

class_session = Session()
class TestMockClass(BaseMockClass):
    def _api_v1_idea(self, method, params, headers):
        if params['special'] == 'value':
            return 'yes'
    def _api_v1_brillo(self, url, body):
        if json.loads(body)['special'] == 'value':
            return 'yes'

a = ClassAdapter(TestMockClass)

session = requests.Session()
with requests_staticmock.mock_session_with_class(session, MyTestClass, 'http://test_
↪context.com'):
    response = new_session.request('get', 'http://test_context.com/api/v1/idea')
```

3.3 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:

3.3.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/tonybaloney/requests-staticmock/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

requests-staticmock could always use more documentation, whether as part of the official requests-staticmock 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/tonybaloney/requests-staticmock/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 :)

3.3.2 Get Started!

Ready to contribute? Here's how to set up *requests-staticmock* for local development.

1. Fork the *requests-staticmock* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/requests-staticmock.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 requests-staticmock
$ cd requests-staticmock/
$ 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 requests-staticmock 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.

3.3.3 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/tonybaloney/requests-staticmock/pull_requests and make sure that the tests pass for all supported Python versions.

3.3.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_requests-staticmock
```

3.4 Credits

3.4.1 Development Lead

- Anthony Shaw <anthonyshaw@apache.org>

3.4.2 Contributors

None yet. Why not be the first?

3.5 History

3.5.1 1.4.0 (2017-09-01)

- Class adapter correctly maps - character to _ as - is invalid method name in Python

3.5.2 1.3.0 (2017-09-01)

- Add a property in MockClass for the adapter instance, helps when you want to respond with static fixture data

3.5.3 1.2.0 (2017-05-10)

- Add support for case-insensitive file matching

3.5.4 1.1.0 (2017-05-10)

- Add support for query params being part of the file path

3.5.5 0.8.0 (2017-02-02)

- Add support for streaming requests and iter_content/iter_lines

3.5.6 0.7.0 (2017-01-29)

- Add support version unpacking, class adapters now support a range of keyword arguments, provided in no particular order.

3.5.7 0.6.0 (2017-01-29)

- Add support for the class adapter methods to return either a string or a response object
- Moved to Py.Test

3.5.8 0.3.0 (2017-01-29)

- Added a class adapter

3.5.9 0.2.0 (2017-01-28)

- Added a context manager for the static mocks

3.5.10 0.1.0 (2017-01-01)

- First release on PyPI.
- genindex
- modindex
- search

Python Module Index

r

Index