Swagger Stub Documentation

Release 0.2.1

Cyprien Guillemot

Jun 28, 2018

Contents

Contents:

CHAPTER 1

swagger-stub

Swagger-stub create automatically a stub of your swagger REST API. This stub can be used anywhere you want like in a pytest fixture for your unit test.

In addition of mocking your API, you can mock some call, and check every call that have been made to the API.

1.1 Related Libraries

You may find related libraries to this one:

- https://github.com/Trax-air/swagger-tester: Auto-test your swagger API in your unit tests. All test calls are generated by your swagger file.
- https://github.com/Trax-air/swagger-aggregator: Aggregate several swagger specs into one. Useful for your API gateways!
- https://github.com/Trax-air/swagger-parser: A helper that parses swagger specs. You can access the HTTP actions / paths and some example data

1.2 Example Usage

```
import pytest
import requests

from swagger_stub import swagger_stub

# This is the fixture of your stub
# You only need to specify the path of the swagger file and the address
# where you want to bind your stub.
@pytest.fixture
def test_stub():
    return swagger_stub([('swagger.yaml', 'http://foo.com')]).next()
```

(continues on next page)

(continued from previous page)

```
# Then you can use this fixture anywhere you want like your API is really running.
def test_swagger_stub(test_stub):
    # Get a definition example
    test_stub.definitions['Foo']

    # Check a simple call
    response = requests.get('http://foo.com/v1/bar/')
    assert response.status_code == 200
    assert response.json() == {
      'foo': 'bar'
    }

    # Check that an invalid body cause an error
    response = requests.post('http://foo.com/v1/bar/', data='invalid data')
    assert response.status_code == 400

    # Mock a call
    test_stub.add_mock_call('get', '/test', {'mock': 'call'})
    response = requests.get('http://foo.com/v1/test')
    assert response.json() == {'mock': 'call'}

    # Set some side_effect like in the mock library
    test_stub.add_mock_side_effect('get', '/iter', [{'test': '1'}, {'test': '2'}, {
↪'test': '3'}])
    response = requests.get('http://foo.com/v1/iter')
    assert response.json() == {'test': '1'}
    response = requests.get('http://foo.com/v1/iter')
    assert response.json() == {'test': '2'}
    response = requests.get('http://foo.com/v1/iter')
    assert response.json() == {'test': '3'}

    # This side effect will raise a custom error
    test_stub.add_mock_side_effect('get', '/error', Exception)

    with pytest.raises(Exception):
        response = requests.get('http://foo.com/v1/error')
```

1.3 Documentation

More documentation is available at https://swagger-stub.readthedocs.org/en/latest/.

1.4 Setup

make install or *pip install swagger-stub*

1.5 License

swagger-stub is licensed under http://opensource.org/licenses/MIT.

CHAPTER 2

Installation

At the command line:

```
$ easy_install swagger_stub
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv swagger_stub
$ pip install swagger_stub
```

CHAPTER 3

Usage

To use Swagger Stub in a project:

```
import pytest
import requests

from swagger_stub import swagger_stub

# This is the fixture of your stub
# You only need to specify the path of the swagger file and the address
# where you want to bind your stub.
@pytest.fixture
def test_stub():
    return swagger_stub([('swagger.yaml', 'http://foo.com')]).next()

# Then you can use this fixture anywhere you want like your API is really running.
def test_swagger_stub(test_stub):
    # Check a simple call
    response = requests.get('http://foo.com/v1/bar/')
    assert response.status_code == 200
    assert response.json() == {
      'foo': 'bar'
    }

    # Check that an invalid body cause an error
    response = requests.post('http://foo.com/v1/bar/', data='invalid data')
    assert response.status_code == 400

    # Mock a call
    test_stub.add_mock_call('get', '/test', {'mock': 'call'})
    response = requests.get('http://foo.com/v1/test')
    assert response.json() == {'mock': 'call'}

    # Set some side_effect like in the mock library
    test_stub.add_mock_side_effect('get', '/iter', [{'test': '1'}, {'test': '2'}, {
↪'test': '3'}])
```

(continues on next page)

(continued from previous page)

```
    response = requests.get('http://foo.com/v1/iter')
    assert response.json() == {'test': '1'}
    response = requests.get('http://foo.com/v1/iter')
    assert response.json() == {'test': '2'}
    response = requests.get('http://foo.com/v1/iter')
    assert response.json() == {'test': '3'}

    # This side effect will raise a custom error
    test_stub.add_mock_side_effect('get', '/error', Exception)

    with pytest.raises(Exception):
        response = requests.get('http://foo.com/v1/error')
```

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/cyprieng/swagger_stub/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

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

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/swagger_stub.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 swagger_stub
$ cd swagger_stub/
$ 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 swagger_stub 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.

4.3 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/cyprieng/swagger_stub/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Cyprien Guillemot <cyprien.guillemot@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.2.1 (2016-6-6)

- Fix repeated base path.

6.2 0.2.0 (2016-5-2)

- Add access to definition example.

6.3 0.1.1 (2016-1-31)

- Change license to MIT.

6.4 0.1 (2016-1-29)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search