mockingjay Documentation

Release 0.2.0

Kevin J. Qiu

May 28, 2015

Contents

1 mockingjay **3**
- 1.1 Features . . . 3
- 1.2 Usage . . . 3

2 Installation **5**

3 Usage **7**
- 3.1 Create a mock service . . . 7
- 3.2 Mock response body . . . 7
- 3.3 Assert request . . . 8

4 Contributing **9**
- 4.1 Types of Contributions . . . 9
- 4.2 Get Started! . . . 10
- 4.3 Pull Request Guidelines . . . 10
- 4.4 Tips . . . 11

5 Credits **13**
- 5.1 Development Lead . . . 13
- 5.2 Contributors . . . 13

6 History **15**

7 0.2.0 (2015-05-27) **17**

8 0.1.1 (2015-05-25) **19**

9 0.1.0 (2015-05-23) **21**

10 Indices and tables **23**

Contents:

CHAPTER 1

mockingjay

A simple library to build mock http services based on HTTPretty

- Free software: BSD license
- Documentation: https://mockingjay.readthedocs.org.

1.1 Features

- Provides a fluent interface for building service mocks using HTTPretty
- Allows mocking of response body using templated fixtures
- Automatic assertion of request attributes (headers, body, etc)

1.2 Usage

See usage

CHAPTER 2

Installation

At the command line:

```
$ pip install mockingjay
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv mockingjay
$ pip install mockingjay
```

CHAPTER 3

Usage

To use mockingjay in a project:

```
import mockingjay
```

3.1 Create a mock service

To create a mock service:

```
service = mockingjay.MockService('https://api.stripe.com')
```

Then you can mock the endpoints using the builder:

```
service.endpoint('GET /v1/tokens/some_token') \  # this creates a mock builder
    .should_return(200, {}, "{}")             \  # specify what it should return in one call
    .register()                                  # register the mock
```

After this is setup, you are able to make a request to the endpoint using any Python HTTP libraries:

```
import requests

response = requests.get('https://api.stripe.com/v1/tokens/some_token')
print(response.text)    # should print out '{}'
```

You can also build up the mock in multiple steps using the fluent interface:

```
service.endpoint('GET /v1/tokens/some_token') \
    .should_return_code(200)  \
    .should_return_header('content-type', 'application/json') \
    .should_return_body('{}') \
    .register()
```

3.2 Mock response body

You have a few options to specify the return body.

3.2.1 Plain text

The simplest way is to use plain text:

```
service.endpoint('GET /v1/tokens/some_token') \
    .should_return_body('{}') \
    .register()
```

3.2.2 JSON

Use `.should_return_json` method if you want the response to be the json representation of a Python dictionary:

```
service.endpoint('GET /v1/tokens/some_token') \
    .should_return_json({"id": "tok_166TGW2jyvokPAPl2u7xKy0v"}) \
    .register()
```

3.2.3 Fixture Template

The most flexible way to mock response is to use a template. To use it, you first need to tell the mock service where to locate the response templates:

```
service = mockingjay.MockService('https://api.stripe.com', fixture_root='/path/to/fixtures')
```

Then create your template. Mockingjay supports Jinja2 templates out of the box, but it's just as easy to hook up with your favourite templating library.

A sample template:

```
{
    "id": "{{ id }}"
}
```

Mock the response using the template:

```
service.endpoint('GET /v1/tokens/some_token') \
    .should_return_body_from_fixture('token.json', id="tok_166TGW2jyvokPAPl2u7xKy0v") \
    .register()
```

3.3 Assert request

Mockingjay also has the ability to automatically assert certain aspects of the request. For example, if your application is talking to the Stripe API, and you want to mock out the charge endpoint. As shown above, it's easy to do so with the `should_return_*` methods to build up the response, but you may also want to assert that the request is made with certain authentication headers or the request body, so that you know your code is calling the external API with the right parameters and so on.

To use the automatic request assertion, you can specify what the request should look like during the endpoint building stage:

```
service.endpoint('POST /v1/charge') \
    .expect_request_user('sk_test_BQokikJOvBiI2HlWgH4olfQ2') \
    .expect_request_body('amount=400&currency=usd') \
    .register()
service.assert_requests_matched()
```

`assert_requests_matched` call will look at the requests made while httpretty is activated, find the matching requests based on the URI and check if the body matches the expected.

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/kevinjqiu/mockingjay/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

mockingjay could always use more documentation, whether as part of the official mockingjay 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/kevinjqiu/mockingjay/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 *mockingjay* for local development.

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

```
$ git clone git@github.com:your_name_here/mockingjay.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 mockingjay
$ cd mockingjay/
$ 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 mockingjay 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, and 3.4, and for PyPy. Check https://travis-ci.org/kevinjqiu/mockingjay/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_mockingjay
```

CHAPTER 5

Credits

5.1 Development Lead

- Kevin J. Qiu <kevin@idempotent.ca>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

0.2.0 (2015-05-27)

- Added the request assertion feature

0.1.1 (2015-05-25)

- Added `should_return_json` builder method
- Mock response status code

CHAPTER 9

0.1.0 (2015-05-23)

- First release on PyPI.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search