Swagger Parser Documentation

Release 1.0.0

Cyprien Guillemot

Jun 29, 2018

Contents

Contents:

CHAPTER 1

swagger-parser

Swagger-parser is a python module giving you access to some interesting data about your swagger file. Like getting a dictionary example from a definition name, get the definition of a dictionary, and more.

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-stub: A stub you can use in your client's unit tests. All the HTTP calls to your swagger API are mocked by default. You can also add your own mocked_calls in your test functions.
- https://github.com/Trax-air/swagger-aggregator: Aggregate several swagger specs into one. Useful for your API gateways!

1.2 Example Usage

```
from swagger_parser import SwaggerParser

parser = SwaggerParser(swagger_path='swagger_path')  # Init with file
parser = SwaggerParser(swagger_dict={})  # Init with dictionary

# Get an example of dict for the definition Foo
parser.definitions_example.get('Foo')

# Get the definition of a dictionary
test = {
  'foo': 'bar'
}
```

(continues on next page)

(continued from previous page)

```
parser.get_dict_definition(test)

# Validate the definition of a dict
parser.validate_definition('Foo', test)

# Validate that the given data match a path specification
parser.validate_request('/foo', 'post', body=test, query={'foo': 'bar'})

# Get the possible return value of a path
# It will return a dictionary with keys as status_code
# and value as example of return value.
parser.get_request_data('/foo', 'post', body=test)

# Get an example of a correct body for a path
parser.get_send_request_correct_body('/foo', 'post')
```

1.3 Documentation

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

1.4 Setup

make install or *pip install swagger-parser*

1.5 License

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

CHAPTER 2

Installation

At the command line:

```
$ easy_install swagger_parser
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv swagger_parser
$ pip install swagger_parser
```

CHAPTER 3

Usage

To use Swagger Parser in a project:

```
from swagger_parser import SwaggerParser

parser = SwaggerParser(swagger_path='swagger_path')  # Init with file
parser = SwaggerParser(swagger_dict={})  # Init with dictionary

# Get an example of dict for the definition Foo
parser.definitions_example.get('Foo')

# Get the definition of a dictionary
test = {
  'foo': 'bar'
}
parser.get_dict_definition(test)

# Validate the definition of a dict
parser.validate_definition('Foo', test)

# Validate that the given data match a path specification
parser.validate_request('/foo', 'post', body=test, query={'foo': 'bar'})

# Get the possible return value of a path
# It will return a dictionary with keys as status_code
# and value as example of return value.
parser.get_request_data('/foo', 'post', body=test)

# Get an example of a correct body for a path
parser.get_send_request_correct_body('/foo', 'post')
```

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_parser/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 Parser could always use more documentation, whether as part of the official Swagger Parser 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_parser/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_parser* for local development.

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

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

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 1.0.0 (2017-6-11)

- Drop support for python 2.6, add support for python 3.5, python 3.6 and pypy
- Fix issue #35
- Add file parser tests and fixes for #40, #41, #42, #43, #44, #45, thanks to @mtherieau
- Use isinstance for simple type checking, thanks to @pankaj28843
- Fixes for #31, #32, #33, thanks to @crudo10 and @beanqueen for the review
- Bug fix when dictionary only contains 1 element, thanks to @TenOs
- Add tests for "official" petstore json and yaml, thanks to @beanqueen

6.2 0.1.11 (2016-9-25)

- Support additionalProperties.

6.3 0.1.10 (2016-8-25)

- Don't choke if there are no definitions
- Generate operations without operationId
- Generate example from properties

6.4 0.1.9 (2016-7-28)

- Support array definitions.

6.5 0.1.8 (2016-5-11)

- Support type field to be an array.
- Use base path to validate request.

6.6 0.1.7 (2016-4-1)

- Support UTF-8 in swagger.yaml.

6.7 0.1.6 (2016-3-16)

- Add support for path-level parameters.

6.8 0.1.5 (2016-2-17)

- Add support for parameters references in path specs.

6.9 0.1.4 (2016-2-10)

- Handle string as status_code.

6.10 0.1.3 (2016-2-3)

- Fix a bug in get_response_example with schema only containing a type field.

6.11 0.1.2 (2016-2-3)

- Support schema with only a type field.

6.12 0.1.1 (2016-1-31)

- Change license to MIT.

6.13 0.1 (2016-1-28)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search