Skinfer Documentation

Release 0.2.0

Scrapinghub

February 26, 2016

Contents

Contents:

CHAPTER 1

Skinfer - tool for working with JSON schemas

Simple tool to infer and/or merge JSON schemas

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

1.1 Features

- Generating schema in **JSON Schema draft 4** format
- Inferring schema from multiple samples
- Merging schemas - nice for generating schema in Map-Reduce fashion or updating an old schema with new data

Example of using *skinfer* to generate a schema from a list of samples:

```
$ cat samples.jsonl
{"name": "Claudio", "age": 29}
{"name": "Roberto", "surname": "Gomez", "age": 72}
$ skinfer --jsonlines samples.jsonl
{
    "$schema": "http://json-schema.org/draft-04/schema",
    "required": [
        "age",
        "name"
    ],
    "type": "object",
    "properties": {
        "age": {
            "type": "number"
        },
        "surname": {
            "type": "string"
        },
        "name": {
            "type": "string"
        }
    }
}
```

Install with:

```
$ pip install skinfer
```

Or, if you don't have `pip`, you can still install it with:

```
$ easy_install skinfer
```

CHAPTER 2

Usage

2.1 Infering schemas from multiple samples

Use the *skinfer* script to generate a schema from a list of samples:

```
$ skinfer --help
usage: skinfer [-h] [--jsonlines] SAMPLE [SAMPLE ...]

Generates a JSON schema based on samples

positional arguments:
  SAMPLE       JSON data sample files

optional arguments:
  -h, --help   show this help message and exit
  --jsonlines  Assume samples are in JSON lines format
```

You can also infer the schema programatically:

```
>>> import skinfer, json
>>> sample1 = {'name': 'Claudio'}
>>> sample2 = {'name': 'Roberto', 'surname': 'Salazar'}
>>> schema = skinfer.infer_schema([sample1, sample2])
>>> import pprint
>>> pprint.pprint(schema)
{'$schema': u'http://json-schema.org/draft-04/schema',
 u'properties': {'name': {'type': 'string'}, 'surname': {'type': 'string'}},
 u'required': ['name'],
 u'type': u'object'}
```

Using the API, you can also generate a schema for only one sample:

```
>>> skinfer.generate_schema({"name": "Claudio", "surname": "Salazar"})
{'$schema': u'http://json-schema.org/draft-04/schema',
 'properties': {'name': {'type': 'string'}, 'surname': {'type': 'string'}},
 'required': ['surname', 'name'],
 'type': 'object'}
```

2.2 Merging existing JSON Schemas

Use *schema_merger* to merge a list of JSON schemas into one JSON schema that represents the common properties:

```
$ schema_merger --help
usage: schema_merger [-h] [-o OUTPUT] schemas [schemas ...]

Merges given JSON Schemas, inferring the required properties

positional arguments:
  schemas     List of JSON schema files to merge

  optional arguments:
    -h, --help  show this help message and exit
    -o OUTPUT   Write JSON schema to this file
```

You can also merge the schema programatically:

```
>>> import skinfer
>>> any_object = {'type': 'object'}
>>> requires_name = {'type': 'object', 'required': ['name'], 'properties': {'name': {'type': 'string
>>> merged_schema = skinfer.merge_schema(any_object, requires_name)
>>> import pprint
>>> pprint.pprint(merged_schema)
{u'properties': {'name': {'type': 'string'}}, u'type': u'object'}
```

CHAPTER 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:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/scrapinghub/skinfer/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.

3.1.2 Fix Bugs

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

3.1.3 Implement Features

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

3.1.4 Write Documentation

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

3.1.5 Submit Feedback

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

3.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/skinfer.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 skinfer
$ cd skinfer/
$ 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 skinfer 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 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 2.7 and for PyPy. Check https://travis-ci.org/scrapinghub/skinfer/pull_requests and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_draft4_generator
```

CHAPTER 4

Credits

4.1 Development Lead

- Scrapinghub <info@scrapinghub.com>

4.2 Contributors

None yet. Why not be the first?

CHAPTER 5

History

CHAPTER 6

0.2.0 (2015-08-10)

- Renamed entry-point script schema_inferer -> skinfer
- Added json_validator
- Dropped Python 2.6 support
- Added more tests

CHAPTER 7

0.1.2 (2015-08-04)

- Bugfix: removed buggy -o argument
- Automated PyPI release via Travis

CHAPTER 8

0.1.1 (2015-05-01)

- Support more complex string-type schemas
- Attempt to infer JSON lines format instead of just failing
- API cleanup: no need for long imports anymore
- Updated documentation, added docstrings
- Fixed merging schema for arrays with tuple vs list validation
- Fixed compatibility issues with Python 2.6
- Improved test coverage, added end-to-end tests

CHAPTER 9

0.1.0 (2015-03-03)

- First release on PyPI.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search