$\mathbf{json}_{c}onfigDocumentation$

Release 2.0.1

Manu Phatak

January 01, 2016

Contents

1 Contents: **1**

1.1 json_config 1
1.2 Installation 2
1.3 Usage 2
1.4 Contributing 2
1.5 Credits 4
1.6 History 5
1.7 json_config package 6

2 Feedback **9**

3 Indices and tables **11**

Python Module Index **13**

CHAPTER 1

Contents:

1.1 json_config

A convenience utility for working with JSON config files.

1.1.1 Features

- Documentation: https://json_config.readthedocs.org
- Open Source: https://github.com/bionikspoon/json_config
- MIT license
- Automatically syncs file on changes.
- Automatically handles complicated nested data structures.
- Designed to be easily extended. Use different serializer libraries to easily switch to yaml, ini, etc.
- Lightweight (<5KB) and Fast.
- Takes advantage of Python's native dictionary syntax.
- Tested against python 2.6, 2.7, 3.3, 3.4, 3.5, and PYPY!
- Unit Tested with high coverage.
- Idiomatic, self-descriptive code & api

```
>>> import json_config
>>> config = json_config.connect('categories.json')
>>> config
Connect({})
>>> config['comics']['dc']['batman']['antagonists'] = ['Scarecrow', 'The Joker', 'Bane']
>>> config['comics']['marvel']['ironman']['antagonists'] = 'Ultron'
>>> print(config.serialize())
{
  "comics": {
    "dc": {
      "batman": {
        "antagonists": [
          "Scarecrow",
          "The Joker",
          "Bane"
        ]
```

```
      }
    },
    "marvel": {
      "ironman": {
        "antagonists": "Ultron"
      }
    }
  }
}
```

1.1.2 Credits

Tools used in rendering this package:

- Cookiecutter
- bionikspoon/cookiecutter-pypackage forked from audreyr/cookiecutter-pypackage

1.2 Installation

At the command line either via easy_install or pip:

```
$ pip install json_config
```

```
$ easy_install json_config
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv json_config
$ pip install json_config
```

Uninstall:

```
$ pip uninstall json_config
```

1.3 Usage

To use json_config in a project:

```
import json_config

config = json_config.connect('config.json')
config['root'] = '/var/www/html/'

print(config['root'])
#OUT: '/var/www/html/'
config
#OUT:  Connect({'root': '/var/www/html/'})
```

1.4 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:

1.4.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/bionikspoon/json_config/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

json_config could always use more documentation, whether as part of the official json_config 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/bionikspoon/json_config/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 :)

1.4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/json_config.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 json_config
$ cd json_config/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b feature/name-of-your-feature
$ git checkout -b hotfix/name-of-your-bugfix
```

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 json_config 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.

1.4.3 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, 3.5, and PyPy. Check https://travis-ci.org/bionikspoon/json_config/pull_requests and make sure that the tests pass for all supported Python versions.

1.4.4 Tips

To run a subset of tests:

```
$ py.test tests/test_json_config.py
```

1.5 Credits

1.5.1 Development Lead

- Manu Phatak <bionikspoon@gmail.com>

1.5.2 Contributors

None yet. Why not be the first?

1.6 History

1.6.1 Next Release

- Stay tuned

1.6.2 2.0.0 (2016-01-01)

- BREAKING: (Internal API) `connect.block` removed
- BREAKING: (Internal API) `connect.write_file` renamed to `connect.save`
- Feature: Rewrote the entire library to encapsulate logic
- Feature: Extendable serializer contract, to allow any config format.
- Feature: Upgrade to stable.
- Feature: Removed threading in favor of a smarter locking mechanism
- Feature: Add support for py26 and py35
- Feature: Pin dependencies
- Feature: Reorganized package and tests
- Fix: Updated doc builds
- Fix: Readme badge links
- 2.0.1: Fix: Failed deploy (travis requirements)

1.6.3 1.2.0 (2015-05-18)

- Feature: Improved compatibility to py27, py32, py33, py34, and pypy
- Feature: Supports multiple config files.
- Feature: Writes less, smarter logic on deciding if a write is necessary.
- Feature: Delegates writes to a background process.
- Testing: Renamed tests to be more descriptive of expectations.
- Testing: Added a bunch of tests describing different scenarios.
- Massive Refactoring

1.6.4 1.1.0 (2015-04-15)

- Massive improvement to documentation and presentation.

1.6.5 1.0.0 (2015-04-13)

- First working version.

1.6.6 0.1.0 (2015-04-11)

- First release on PyPI.

1.7 json_config package

1.7.1 JSON Config

A convenience utility for working with JSON config files.

```
>>> import json_config, os
>>> config = json_config.connect('config.json')
>>> config['root'] = '/var/www/html'
>>> print(config['root'])
/var/www/html
>>> config
Connect({'root': '/var/www/html'})
```

```
>>> os.remove('config.json')
```

`json_config.`**`connect`**(*config_file*, *file_type=None*, ***kwargs*)

1.7.2 Submodules

1.7.3 json_config._compat module

Python 2to3 compatibility handling.

class `json_config._compat.`**`NullHandler`**(*level=0*)

Bases: `logging.Handler`

This handler does nothing. It's intended to be used to avoid the "No handlers could be found for logger XXX" one-off warning. This is important for library code, which may contain code to log events. If a user of the library does not configure logging, the one-off warning might be produced; to avoid this, the library developer simply needs to instantiate a NullHandler and add it to the top-level logger of the library module or package.

`createLock`()

`emit`(*record*)

`handle`(*record*)

`json_config._compat.`**`FileNotFoundError`**

alias of `IOError`

1.7.4 json_config.contracts module

class `json_config.contracts.`**`AbstractSaveFile`**

Bases: `object`

`config_file` = **NotImplemented**

`save`()

class `json_config.contracts.`**`AbstractSerializer`**

Bases: `object`

deserialize(*string*)

serialize(***options*)

serializer_ext = **NotImplemented**

class `json_config.contracts.`**`AbstractTraceRoot`**
Bases: `object`

1.7.5 json_config.main module

class `json_config.main.`**`AutoConfigBase`**(*config_file=None*, ***kwargs*)
Bases: *`json_config.main.AutoSyncMixin`*, *`json_config.main.AutoDict`*

class `json_config.main.`**`AutoDict`**(*obj=None*, *_root=None*, *_parent=None*, *_key=None*)
Bases: *`json_config.main.TraceRootMixin`*, `collections.defaultdict`

save()

update(*E*, ***F*) → None. Update D from E and F: for k in E: D[k]
= E[k] (if E has keys else: for (k, v) in E: D[k] = v) then: for k in

F: D[k] = F[k]

class `json_config.main.`**`AutoSyncMixin`**(***kwargs*)
Bases: *`json_config.contracts.AbstractSaveFile`*, *`json_config.contracts.AbstractTraceRoot`*, *`json_config.contracts.AbstractSerializer`*

config_file = **None**

save()

class `json_config.main.`**`PrettyJSONMixin`**
Bases: *`json_config.contracts.AbstractSerializer`*

deserialize(*string*)

serialize(***options*)

serializer_ext = **'json'**

serializer_indent = **2**

serializer_sort_keys = **True**

class `json_config.main.`**`TraceRootMixin`**
Bases: *`json_config.contracts.AbstractTraceRoot`*

lock(**args*, ***kwds*)

`json_config.main.`**`connect`**(*config_file*, *file_type=None*, ***kwargs*)

CHAPTER 2

Feedback

If you have any suggestions or questions about **json_config** feel free to email me at bionikspoon@gmail.com.

If you encounter any errors or problems with **json_config**, please let me know! Open an Issue at the GitHub https://github.com/bionikspoon/json_config main repository.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

j

`json_config`, 6
`json_config._compat`, 6
`json_config.contracts`, 6
`json_config.main`, 7

A

AbstractSaveFile (class in json_config.contracts), 6
AbstractSerializer (class in json_config.contracts), 6
AbstractTraceRoot (class in json_config.contracts), 7
AutoConfigBase (class in json_config.main), 7
AutoDict (class in json_config.main), 7
AutoSyncMixin (class in json_config.main), 7

C

config_file (json_config.contracts.AbstractSaveFile attribute), 6
config_file (json_config.main.AutoSyncMixin attribute), 7
connect() (in module json_config), 6
connect() (in module json_config.main), 7
createLock() (json_config._compat.NullHandler method), 6

D

deserialize() (json_config.contracts.AbstractSerializer method), 6
deserialize() (json_config.main.PrettyJSONMixin method), 7

E

emit() (json_config._compat.NullHandler method), 6

F

FileNotFoundError (in module json_config._compat), 6

H

handle() (json_config._compat.NullHandler method), 6

J

json_config (module), 6
json_config._compat (module), 6
json_config.contracts (module), 6
json_config.main (module), 7

L

lock() (json_config.main.TraceRootMixin method), 7

N

NullHandler (class in json_config._compat), 6

P

PrettyJSONMixin (class in json_config.main), 7

S

save() (json_config.contracts.AbstractSaveFile method), 6
save() (json_config.main.AutoDict method), 7
save() (json_config.main.AutoSyncMixin method), 7
serialize() (json_config.contracts.AbstractSerializer method), 7
serialize() (json_config.main.PrettyJSONMixin method), 7
serializer_ext (json_config.contracts.AbstractSerializer attribute), 7
serializer_ext (json_config.main.PrettyJSONMixin attribute), 7
serializer_indent (json_config.main.PrettyJSONMixin attribute), 7
serializer_sort_keys (json_config.main.PrettyJSONMixin attribute), 7

T

TraceRootMixin (class in json_config.main), 7

U

update() (json_config.main.AutoDict method), 7