ultra-config Documentation

Release 0.6.3

Tim Martin

Sep 19, 2018

Contents

Contents:

CHAPTER 1

ultra-config

An extendable configuration that enables you to configure your application via python modules, config files, environment variables and more!

- Free software: MIT license
- Documentation: https://ultra-config.readthedocs.io.

1.1 Installation

```
pip install ultra-config
```

1.2 Features

- Load configuration from a variety of sources including environment variables, json files, ini files, and python objects
- Easily extend with your own configuration mechanisms
- Offers a global configuration object for you application
- Easily inject configuration into functions with the ability to override them for testing
- Ability to fail fast if missing configuration

1.3 Examples

1.3.1 global configuration

```
from ultra_config import GlobalConfig

# Loads all env variables that begin with MY_APP, configuration
# from a json file and a custom override
GlobalConfig.load(env_var_prefix='MY_APP',
                  json_file='/opt/my_app/config.json',
                  overrides={'MY_VAR': 'some_val'})

@GlobalConfig.inject('MY_VAR', value='OTHER_VAR')
def my_func(arg1, value=None):
    print(arg1)
    print(value)

my_func()
# Prints the value of MY_VAR and OTHER_VAR

my_func(value='custom')
# prints the value of MY_VAR and then prints the "custom" since we explicitly passed␣
↪that in
```

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install ultra-config, run this command in your terminal:

```
$ pip install ultra_config
```

This is the preferred method to install ultra-config, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for ultra-config can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/timmmartin19/ultra_config
```

Or download the tarball:

```
$ curl  -OL https://github.com/timmmartin19/ultra_config/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use ultra-config in a project:

```
import ultra_config
```

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/timmmartin19/ultra_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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

ultra-config could always use more documentation, whether as part of the official ultra-config 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/timmmartin19/ultra_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 :)

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/ultra_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 ultra_config
$ cd ultra_config/
$ 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 ultra_config tests
$ python setup.py test or 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/timmmartin19/ultra_config/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_ultra_config
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search