
cascade-config

Release 0.3.0

RalfG

Jul 03, 2023

CONTENTS

- 1 Installation 3**
- 2 Quickstart 5**
- 3 Contributing 7**
- 4 Changelog 9**
 - 4.1 Usage 9
 - 4.2 API 9
 - 4.3 Changelog 11
 - 4.3.1 [0.3.0] - 15/11/2021 11
 - 4.3.2 [0.2.0] - 28/09/2021 11
 - 4.3.3 [0.1.0-a0] - 03/08/2020 12
 - 4.4 Contributing 12
 - 4.4.1 Before you begin 12
 - 4.4.2 How to contribute 12
 - 4.4.3 Development setup 12
 - 4.4.4 Development workflow 12
- Python Module Index 13**
- Index 15**

Cascading Python configuration from the CLI and multiple config files.

cascade-config simplifies handling multiple configuration sources, such as config files, command line arguments, or even simple dictionaries. Configuration sources can be added one-by-one and will be parsed in hierarchical order, with each new source updating the existing configuration.

INSTALLATION

Install with pip

```
pip install cascade-config
```


QUICKSTART

Multiple configuration sources can be added to a `cascade_config.CascadeConfig` object. When parsed, each configuration will be added in hierarchical order and update the existing configuration. The result is a single dictionary containing the cascaded configuration.

```
from cascade_config import CascadeConfig

# Setup CascadeConfig instance with JSON schema for validation
cascade_conf = CascadeConfig(validation_schema="config_schema.json")

# Add default and user configurations in cascading order
cascade_conf.add_json("config_default.json")
cascade_conf.add_json("config_user.json")

# Parse the configuration files into a dictionary
config = cascade_conf.parse()
```

See [Usage](#) for more information and examples.

CONTRIBUTING

Bugs, questions or suggestions? Feel free to post an issue in the [issue tracker](#) or to make a pull request! See [Contributing](#) for more info.

CHANGELOG

See [Changelog](#).

4.1 Usage

To do.

4.2 API

Cascading configuration from the CLI and config files.

class `cascade_config.CascadeConfig`(*validation_schema=None, none_overrides_value=False*)
Cascading configuration.

Parameters

- **validation_schema** (*str, path-like, dict, or cascade_config.ValidationSchema, optional*) – JSON Schema to validate fully cascaded configuration
- **none_overrides_value** (*bool*) – If True, a None value overrides a not-None value from the previous configuration. If False, None values will never override not-None values.

Examples

```
>>> cascade_conf = CascadeConfig(validation_schema="config_schema.json")
>>> cascade_conf.add_json("config_default.json")
>>> cascade_conf.add_json("config_user.json")
>>> config = cascade_conf.parse()
```

property validation_schema

JSON Schema to validate fully cascaded configuration.

add_dict (*args, **kwargs)

Add dictionary configuration source to source list. *args and **kwargs are passed to `cascade_config.DictConfigSource()`.

add_argumentparser (*args, **kwargs)

Add argumentparser configuration source to source list. *args and **kwargs are passed to `cascade_config.ArgumentParserConfigSource()`.

add_namespace (*args, **kwargs)

Add argparse Namespace configuration source to source list. *args and **kwargs are passed to `cascade_config.NamespaceConfigSource()`.

add_json (*args, **kwargs)

Add JSON configuration source to source list. *args and **kwargs are passed to `cascade_config.JSONConfigSource()`.

parse() → Dict

Parse all sources, cascade, validate, and return cascaded configuration.

class `cascade_config.DictConfigSource` (source, validation_schema=None, subkey=None)

Initialize a single configuration source.

Parameters

- **source** (str, path-like, dict, argparse.ArgumentParser) – source for the configuration, either a dictionary, path to a file, or argument parser.
- **validation_schema** (str, path-like, dict, or `cascade_config.ValidationSchema`, optional) – JSON Schema to validate single configuration
- **subkey** (str) – adds the configuration to a subkey of the final cascaded configuration; e.g. specifying a subkey “user” for a configuration source, would add it under the key “user” in the cascaded configuration, instead of updating the root of the existing configuration

load()

load the configuration from the source and return it as a dictionary

class `cascade_config.JSONConfigSource` (source, validation_schema=None, subkey=None)

Initialize a single configuration source.

Parameters

- **source** (str, path-like, dict, argparse.ArgumentParser) – source for the configuration, either a dictionary, path to a file, or argument parser.
- **validation_schema** (str, path-like, dict, or `cascade_config.ValidationSchema`, optional) – JSON Schema to validate single configuration
- **subkey** (str) – adds the configuration to a subkey of the final cascaded configuration; e.g. specifying a subkey “user” for a configuration source, would add it under the key “user” in the cascaded configuration, instead of updating the root of the existing configuration

load()

load the configuration from the source and return it as a dictionary

class `cascade_config.ArgumentParserConfigSource` (source, validation_schema=None, subkey=None)

Initialize a single configuration source.

Parameters

- **source** (str, path-like, dict, argparse.ArgumentParser) – source for the configuration, either a dictionary, path to a file, or argument parser.
- **validation_schema** (str, path-like, dict, or `cascade_config.ValidationSchema`, optional) – JSON Schema to validate single configuration
- **subkey** (str) – adds the configuration to a subkey of the final cascaded configuration; e.g. specifying a subkey “user” for a configuration source, would add it under the key “user” in the cascaded configuration, instead of updating the root of the existing configuration

load()
load the configuration from the source and return it as a dictionary

class `cascade_config.NamespaceConfigSource` (*source*, *validation_schema=None*, *subkey=None*)

Initialize a single configuration source.

Parameters

- **source** (*str*, *path-like*, *dict*, *argparse.ArgumentParser*) – source for the configuration, either a dictionary, path to a file, or argument parser.
- **validation_schema** (*str*, *path-like*, *dict*, or `cascade_config.ValidationSchema`, *optional*) – JSON Schema to validate single configuration
- **subkey** (*str*) – adds the configuration to a subkey of the final cascaded configuration; e.g. specifying a subkey “user” for a configuration source, would add it under the key “user” in the cascaded configuration, instead of updating the root of the existing configuration

load()
load the configuration from the source and return it as a dictionary

class `cascade_config.ValidationSchema` (*source*)
ValidationSchema.

classmethod from_object (*obj*)
Return ValidationSchema from str, path-like, dict, or ValidationSchema.

load() → Dict
Load validation schema.

4.3 Changelog

All notable changes to this project will be documented in this file.

The format is based on [Keep a Changelog](#), and this project adheres to [Semantic Versioning](#).

4.3.1 [0.3.0] - 15/11/2021

- Fix parsing of empty dictionaries and `False` configuration values. These would previously be overridden by the downstream configuration.

4.3.2 [0.2.0] - 28/09/2021

- Add `none_overrides_value` option. Before this change, `None` values would unexpectedly override previously configured values. Now, the previous value will be retained if newer values are `None`. The old behavior can be re-enabled with by setting the `none_overrides_value` argument of `CascadeConfig` to `True`.

4.3.3 [0.1.0-a0] - 03/08/2020

- Initial release

4.4 Contributing

This document briefly describes how to contribute to [cascade-config](#).

4.4.1 Before you begin

If you have an idea for a feature, use case to add or an approach for a bugfix, you are welcome to communicate it with the community by creating an issue in [GitHub issues](#).

4.4.2 How to contribute

- Fork [cascade-config](#) on GitHub to make your changes.
- Commit and push your changes to your [fork](#).
- Open a [pull request](#) with these changes. Your pull request message ideally should include:
 - A description of why the changes should be made.
 - A description of the implementation of the changes.
 - A description of how to test the changes.
- The pull request should pass all the continuous integration tests which are automatically run by [GitHub Actions](#).

4.4.3 Development setup

1. Setup Python 3 and [Flit](#)
2. Clone the [cascade-config repository](#) and run `flit install` to setup an editable version of `cascade-config`.

4.4.4 Development workflow

- When a new version is ready to be published:
 1. Change the `__version__` in `cascade_config.py` following [semantic versioning](#).
 2. Update the documentation (`README.md` and `docs/source/usage.rst`) if required.
 3. Update the changelog (if not already done) in `CHANGELOG.md` according to [Keep a Changelog](#).
 4. Commit all final changes to the `master` branch.
 5. On `master`, set a new tag with the version number, e.g. `git tag v0.1.5`.
 6. Push to GitHub, with the tag: `git push; git push --tags`.
- When a new tag is pushed to (or made on) GitHub that matches `v*`, the following GitHub Actions are triggered:
 1. The Python package is build and published to PyPI.
 2. Using the [Git Release](#) action, a new GitHub release is made with the changes that are listed in `CHANGELOG.md`.

PYTHON MODULE INDEX

C

`cascade_config`, [9](#)

INDEX

A

`add_argumentparser()` (*cascade_config.CascadeConfig* method), 9
`add_dict()` (*cascade_config.CascadeConfig* method), 9
`add_json()` (*cascade_config.CascadeConfig* method), 10
`add_namespace()` (*cascade_config.CascadeConfig* method), 9
`ArgumentParserConfigSource` (class in *cascade_config*), 10

C

`cascade_config`
 module, 9
`CascadeConfig` (class in *cascade_config*), 9

D

`DictConfigSource` (class in *cascade_config*), 10

F

`from_object()` (*cascade_config.ValidationSchema* class method), 11

J

`JSONConfigSource` (class in *cascade_config*), 10

L

`load()` (*cascade_config.ArgumentParserConfigSource* method), 10
`load()` (*cascade_config.DictConfigSource* method), 10
`load()` (*cascade_config.JSONConfigSource* method), 10
`load()` (*cascade_config.NamespaceConfigSource* method), 11
`load()` (*cascade_config.ValidationSchema* method), 11

M

module
 cascade_config, 9

N

`NamespaceConfigSource` (class in *cascade_config*), 11

P

`parse()` (*cascade_config.CascadeConfig* method), 10

V

`validation_schema()` (*cascade_config.CascadeConfig* property), 9
`ValidationSchema` (class in *cascade_config*), 11