tg-option-container

Release 0.4.0

Sep 01, 2017

Contents

TG Option Container - Container for dictionary-like validated data structures

Documentation is available on Read the Docs: http://tg-option-container.readthedocs.io

CHAPTER 1

Contents

API

class tg_option_container.**OptionContainer**(***kwargs*)
Container for dictionary-like validated data structures

Provides a common base logic for building validated dictionaries. Rules are defined by a props attribute on the class. These is mainly used for validating various JSON based configuration data which MUST conform to a specific structure. OptionContainers support single-inheritance based extending (diamond-inheritance does not work). Child classes can also overwrite parent declarations by redefining them in their props.

Examples

```
>>> from tg_option_container import Option, OptionContainer
```

```
>>> class SampleOptions(OptionContainer):
>>>     props = [
>>>         Option.integer('verbosity', default=0, choices=[1, 2, 3]),
>>>     ]
```

```
>>> class ExtendedSampleOptions(SampleOptions):
>>>     props = [
>>>         Option.integer('timeout', default=30),
>>>     ]
```

Note: *ExtendedSampleOptions* accepts both *timeout* and *verbosity* props.

as_dict()
Get a dictionary representation of this OptionContainer :returns: dict

get(*key*)
Get value of *key*

Raises `KeyError` – If key does not exist

set(*key*, *value*)
Set *key* to *value*

Validate the *value* based on props definition for *key*.

Parameters

- **key** (*Union[str, tuple]*) – Key to set for this option container, if the provided value is a tuple, it's expected to be a point to the nested container key.
- **value** – value to set for key

Raises

- `InvalidOption` – If validation fails
- `AssertionError` – If the *key* is not valid for this container
- `NotImplementedError` – If the current option container instance is nested

class `tg_option_container.`**Option**(*name*, *default*, *validators=None*, *clean=None*, ***kwargs*)
Single option definition for option containers

name
str – The option name

default
any – The default value

validators
callable – Callable with signature *fn(value)* -> *bool* used to validate the input value (can also raise InvalidOption)

clean
callable – Callable with signature *fn(value)* -> *any* used to clean the value before running I{validators} (can also be a list of callables).

choices
If provided adds ChoicesValidator to I{validators}

expected_type
If provided adds TypeValidator to I{validators}. This can also be an instance of TypeValidator (or a subclass instance).

min_value
If provided adds MinValueValidator to I{validators}

max_value
If provided adds MaxValueValidator to I{validators}

none_to_default
If provided *None* will be treated as *Undefined* (cleaned to default)

resolve_default
If provided default will be treated as a callable

classmethod boolean(*name*, *default*, *validators=None*, *clean=None*, ***kwargs*)
Option of boolean type

Note: This is a shorthand for: Option(..., expected_type=bool)

Parameters

- **name** – see Option.__init__
- **default** – see Option.__init__
- **validators** – see Option.__init__
- **clean** – see Option.__init__
- ****kwargs** – see Option.__init__

classmethod integer(*name*, *default*, *validators=None*, *clean=None*, ***kwargs*)
Option of integer type

Note: This is a shorthand for: Option(..., expected_type=int)

Parameters

- **name** – see Option.__init__
- **default** – see Option.__init__
- **validators** – see Option.__init__
- **clean** – see Option.__init__
- ****kwargs** – see Option.__init__

classmethod iso8601(*name*, *default*, *validators=None*, *clean=None*, ***kwargs*)
Option of iso8601 type

Note:

This is a shorthand for: Option(..., expected_type=datetime.datetime, expected_type__append=_('Please use ISO_8601.'), clean=clean_datetime)

Accepts the following formats:

- ISO_8601
- ISO_8601 with spaces: 2016-05-09 16:00:00 +02:00

Note: For both variants the timezone part is optional and defaults to UTC

Parameters

- **name** – see Option.__init__
- **default** – see Option.__init__
- **validators** – see Option.__init__
- **clean** – see Option.__init__
- ****kwargs** – see Option.__init__

classmethod **list**(*name*, *default*, *validators=None*, *clean=None*, *inner_type=None*, *allow_empty=True*, ***kwargs*)
Option of list type

Note: This is a shorthand for: Option(..., expected_type=ListValidator(inner_type, autoclean, allow_empty))

Parameters

- **inner_type** (*any*) – Can be used to construct a typed list
- **allow_empty** (*Optional[bool]*) – If False ListValidator will also check that the list is not empty. Defaults to **True**
- **name** – see Option.__init__
- **default** – see Option.__init__
- **validators** – see Option.__init__
- **clean** – see Option.__init__
- ****kwargs** – see Option.__init__

classmethod **nested**(*name*, *container_cls*, *validators=None*, *clean=None*, ***kwargs*)
Option of string type

Note:

This is a shorthand for: Option(..., expected_type=container_cls, clean=clean_option_container(container_cls))

Parameters

- **container_cls** – The option container to nest
- **name** – see Option.__init__
- **validators** – see Option.__init__
- **clean** – see Option.__init__
- ****kwargs** – see Option.__init__

classmethod **string**(*name*, *default*, *validators=None*, *clean=None*, ***kwargs*)
Option of string type

Note: This is a shorthand for: Option(..., expected_type=str)

Parameters

- **name** – see Option.__init__
- **default** – see Option.__init__
- **validators** – see Option.__init__
- **clean** – see Option.__init__
- ****kwargs** – see Option.__init__

validate(*value*)
Clean and validate the provided value against I{validators}

Parameters **value** – Value to validate

CHAPTER 2

Getting started

Install tg-option-container:

```
pip install tg-option-container
```

Then use it in your project:

```
from tg_option_container import Option, OptionContainer


class Character(OptionContainer):
    props = [
        Option.string('name', None),
        Option.string('gender', None, choices=('M', 'N')),
    ]


john = Character(name='John Smith', gender='M')

# This will raise: tg_option_container.types.InvalidOption: Invalid choice x for␣
↪option `gender`, choices are ('M', 'N').
mary = Character(name='Mary Smith', gender='x')
```

CHAPTER 3

Development

You can run the tests by running `tox` in the top-level of the project.

Python Module Index

t

Index