ConfigObject Documentation

Release 0.4

Gael Pasgrimaud

September 23, 2016

Contents

ConfigObject is a wrapper to the python ConfigParser to allow to access sections/options with attribute names:

```
>>> from ConfigObject import ConfigObject
>>> config = ConfigObject()

>>> config.section.value = 1
```

Values are stored as string:

```
>>> config.section.value
'1'
```

Values are returned as *ConfigValue* to convert result to other types:

```
>>> config.section.value.as_int()
1
```

Here is how list are stored:

```
>>> config.section.value1 = range(2)
>>> print config.section.value1
0
1
>>> config.section.value1.as_list()
['0', '1']
```

You can use keys:

```
>>> config['set']['value'] = 1
>>> config['set']['value'].as_int()
1
```

You can set a section as dict:

```
>>> config.dict = dict(value=1, value2=['0', '1'])
>>> config.dict.value.as_int()
1
>>> config.dict.value2.as_list()
['0', '1']
```

See what your section look like:

```
>>> config['set']
{'value': '1'}
```

Delete options:

```
>>> del config['set'].value
>>> config['set']
{}
```

CHAPTER 1

Module Contents

class ConfigObject.**ConfigObject**(**args*, ***kwargs*)

ConfigParser wrapper

class ConfigObject.**ConfigValue**

as_bool(*true=True*, *false=False*)

convert value to bool:

```
>>> config = ConfigObject()
>>> config.bools = dict(true=True, false=False)
```

```
>>> config.bools.true
'true'
>>> config.bools.true.as_bool()
True
>>> config.bools.true.as_bool('on', 'off')
'on'
```

```
>>> config.bools.false
'false'
>>> config.bools.false.as_bool()
False
>>> config.bools.false.as_bool('on', 'off')
'off'
```

```
>>> config.bools.none.as_bool()
False
```

as_float()

convert value to float

as_int()

convert value to int

as_list(*sep=None*)

convert value to list:: >>> config = ConfigObject() >>> config.listes = dict(liste=[0, 1], string='1;2;3')

```
>>> print config.listes.liste
0
1
>>> config.listes.liste.as_list()
['0', '1']
```

```
>>> config.listes.string.as_list(sep=';')
['1', '2', '3']
```

`ConfigObject.`**`config_module`**(*name*, *file*, **filenames*, ***defaults*)

Allow to set a *ConfigObject* as module. You have to add this to `yourproject/config.py`:

```
# -*- coding: utf-8 -*-
from ConfigObject import config_module
import os
filename = os.path.join(os.path.dirname(__file__), 'config.ini')
config = config_module(__name__, __file__, filename)
```

Then you are able to use `from yourproject import config` where `config` is the *ConfigObject* itself.

Python Module Index

C

A

C