Thingy

Apr 18, 2023

Contents

Dictionaries as objects, that can have different dictionary views!

CHAPTER 1

Install

```
$ pip install thingy
```

CHAPTER 2

Examples

2.1 Dictionaries as objects...

```
>>> class MyThingy(Thingy)
...     @property
...     def foobaz(self):
...         return self.foo + self.baz

>>> thingy = MyThingy({"foo": "bar", "baz": "qux"})
>>> thingy.foo
"bar"
>>> thingy.foobaz
"barqux"

>>> thingy.foo = "BARRRR"
>>> thingy.view()
{"foo": "BARRRR", "baz": "qux"}
```

2.2 ...that can have different dictionary views!

```
>>> MyThingy.add_view(name="fooz", include=["foo", "foobaz"])
>>> MyThingy.add_view(name="no_foo", defaults=True, exclude="foo")

>>> thingy = MyThingy({"foo": "bar", "baz": "qux"})
>>> thingy.view("fooz")
{"foo": "bar", "foobaz": "barqux"}
>>> thingy.view("no_foo")
{"baz": "qux"}
```

CHAPTER 3

Why Thingy?

Because it's much more enjoyable to write `foo.bar` than `foo["bar"]`.

Thingy is mainly meant to be used inside other libraries to provide abstractions over dictionaries, which can be useful for writing ORMs or similar utilities.

Thingy's views system is also particularly useful as-is when you intensively manipulate dictionaries and often restrict those dictionaries to a few redundant items.

CHAPTER 4

Tests

To run Thingy tests:

- install developers requirements with `pip install -r requirements.txt`;
- run `pytest`.

CHAPTER 5

Sponsors

CHAPTER 6

License

MIT

CHAPTER 7

Documentation

7.1 API reference

class thingy.**View**(*defaults=False*, *include=None*, *exclude=None*, *ordered=False*)
Transform an `object` into a dict

Parameters

- **defaults** (*bool*) – Include attributes of object
- **include** (*list*) – A list of properties to include
- **exclude** (*list*) – A list of attributes to exclude
- **ordered** (*bool*) – Use an `OrderedDict` instead

class thingy.**Thingy**(**args*, ***kwargs*)
Allows you to use object notation instead of dict notation

classmethod add_view(*name*, **args*, ***kwargs*)

update(**args*, ***kwargs*)

view(*name='defaults'*)

class thingy.**DatabaseThingy**(**args*, ***kwargs*)

database

database_name = 'database'

classmethod get_database()

classmethod get_database_name()

classmethod get_table()

classmethod get_table_name()

table

```
    table_name = 'thingy'
```

Python Module Index

t

Index