DotObject Documentation

Release 1.0

Sep Ehr (Seperman)

March 15, 2016

Contents

DotObject: Dot lets you define objects in dot notation format with save and load methods.

Compatibility: Python 2.7, 3.3, 3.4, 3.5

CHAPTER 1

Installation

Install from PyPi:

```
pip install dotobject
```

CHAPTER 2

DotObject 1.0.0

Subclass Dot to use this module

class dot.**Dot**(*root_name=None*, *int_starts_with='i'*)

Dot Notation Object.

Dot lets you define objects in dot notation format and load/save them to external resource when needed.

Background

Dot Notation object was originally designed to be the base library for a Redis client for Python. Thus the names 'load' and 'save' come from. The idea was to have python object that simply by writing obj.item="value", it sets the redis key "obj.item" with "value" value. And as soon as it detects you are retrieving the value, it gets the latest version from Redis. But in the mean time, it gives you a lazy object till it actually needs the value from Redis.

So the Dot notation object is basically a lazy object that once its "load" and "save" methods are defined, it will run those methods when the object is saved or retrieved.

Parameters

root_name [String, Optional.] It is used to overwrite the Dot object root name.

int_starts_with: String, Optional. Default: i It is used to idefntify integer parts since Python does not let integers as attributes.

Returns

A lazy object that will be evaluated when it is actually used.

Examples

Defining your own Dot

```
>>> from dot import Dot
>>> class This(Dot):
...     def __init__(self, *args, **kwargs):
...         super(This, self).__init__(*args, **kwargs)
...         self.items = {}
...     def load(self, paths):
...         return {i: self.items[i] if i in self.items else "value %s" % i for i in paths}
...     def save(self, path, value):
...         self.items[path] = value
...
```

Creating a Dot object

```
>>> this = This()
>>> aa = this.part1.part2.part3.part4
>>> aa
<Lazy object: this.part1.part2.part3.part4>
>>> print(aa)
value this.part1.part2.part3.part4
>>> aa
value this.part1.part2.part3.part4
```

Dot objects get evaluated in a batch

```
>>> this = This()
>>> aa = this.part1
>>> aa
<Lazy object: this.part1>
>>> bb = this.part2
>>> bb
<Lazy object: this.part2>
>>> print(aa)
value this.part1
>>> aa
value this.part1
>>> bb
value this.part2
```

Dealing with paths that have integers as a part

```
>>> bb = this.part1.part2.i120
>>> bb
<Lazy object: this.part1.part2.120>
>>> print bb
value this.part1.part2.120
```

Dealing with Dots like dictionary keys

```
>>> cc = this['part1.part2.part4']
>>> cc
<Lazy object: this.part1.part2.part4>
>>> dd = this['part1.%s.part4' % 100]
>>> dd
<Lazy object: this.part1.100.part4>
```

Saving Dots

```
>>> this.part1.part2.part3.part4 = "new value"
>>> zz = this.part1.part2.part3.part4
>>> zz
new value
```

Changing Root name

```
>>> class That(This):
...     pass
>>> that = That()
>>> aa = that.something
>>> print(aa)
value that.something
>>> bb = this.something
>>> bb
<Lazy object: this.something>
```

Flushing cache

```
>>> aa = this.part1
>>> print aa
value this.part1
>>> bb = this.part1 # reads from the cache
>>> this.flush()
>>> bb = this.part1 # Will evaluate this.part1 again
```

flush()
Emptys the Dot cache

load(*paths*)
Must be implemented by subclasses. All Dot objects are lazy loaded. Once the object needs to actually be used, it gets evaluated. The load method here is what is used to evaluate a batch of object paths.

Parameters

paths [List of paths.] The load method should get a list of paths to be loaded and return a dictionary of {path: value}

save(*path*, *value*)
Must be implemented by subclasses.

The save method is called to save a value for a path.

Parameters

path : String.

value: Can be any Python object.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

CHAPTER 4

Changelog

CHAPTER 5

Author

Seperman

- Github
- Linkedin

d

D

F

L

S