Python

Release

December 05, 2014

1 Overview **3**

2 Changes **5**

3 Basic Usage **7**

4 Advanced Usage **9**

5 Indices and tables **11**

Python Module Index **13**

version 0.2.0

author Luca De Vitis <luca at monkeython.com>

contact http://www.monkeython.com

CHAPTER 1

Overview

(Spelled like multiplug) The purpose of this module is to provide a dead simple plugin handler module. I wanted something:

1. Capable of handling multiple plugins (and that's pretty obvious)
2. Capable of handling multiple implementation of the same plugin
3. Capable of handling multiple `pkg_resources.WorkingSet`-s... by itself
4. Easy to initialize in your pluggable application/framework.

I wanted somthing like:

```
content_types = multipla.power_up('scriba.content_types')

def to_json(object):
    content_type = content_types.get('application/json')
    return content_type.format(ojbect)

def to_user_supplied_type(object, content_type):
    return content_types.get(content_type).format(object)
```

or:

```
from loremipsum import generator
import multipla

samples = multipla.power_up('loremipsum.samples')
vaporware = generator.Generator(samples.get('vaporware'))
```

You can read more on Pythonhosted or Read the Docs. Since this package has en extensive docstring documentation as well as code comments, you can read more browsing the source code or in the python interactive shell.

CHAPTER 2

Changes

0.2.0

- More documentation.

0.1.0

- Added `RatedDict.ratings` method to read item ratings values.

0.0.4

- API refactoring.

0.0.2

- API refactoring.

0.0.1

- Pre-Alpha release.

CHAPTER 3

Basic Usage

power_up(*name*, **args*)

Creates and returns a rated dictionary of plugins.

Parameters

- **name** (*str*) – The multi-plug name (i.e. entry point group).
- **args** – Variable argument list of `pkg_resources.WorkingSet`.

Return type `Multipla`

I meant to have just one `Multipla` instances for each group of entry points. They are powered up by subscribing (as per `pkg_resources.WorkingSet.subscribe()`) to each `pkg_resources.WorkingSet` in the variable argument list. If no extra argument is provided, (default) `pkg_resources.working_set` is used. Subscription causes the `Multipla` instance to register any plugin in the given `pkg_resources.WorkingSet`, and let it be notified of any plugin that will be added in the future. Subscribing a `Multipla` twice (or more) to the same `pkg_resources.WorkingSet` neither add any overhead, nor makes the instance to register a give plugin more than once, so it's safer to use this function as the only module interface.

```
>>> import multipla
>>>
>>> multipla.power_up('plugin_group')
<Multipla 'plugin_group'>
>>>
>>> plugin_group = multipla.power_up('plugin_group')
>>> plugin_group is multipla.power_up('plugin_group')
True
>>> isinstance(plugin_group, multipla.Multipla)
True
```

CHAPTER 4

Advanced Usage

class **Multipla** (*name*)

The power strip to put yout plugs into.

Parameters **name** – The name of the power strip (i.e. the entry point group).

This class represents the plugin group. Since this class inherits from `RatedDict`, it's possible to use it almost as a dictionary, and also rate your plugin names. Each item in an instance of this class is an instance of `MultiPlugAdapter`, actually, so you can use more implementation of a given plugin name. On the average you want to use the higest rated implementation trough the `Multipla.get()` method, but you can also use the dictionary item access syntax to reach for all the implementations a achieve your goal.

get (*name*, *default=None*)

Get the higest rated `plug` for the given plug `name`.

Parameters

- **name** – The plugin name.
- **default** – The default value to return if lookup fails.

Returns The highest rated plugin.

Raises

- **KeyError** – If `name` lookup fails.
- **ValueError** – See `RatedDict.highest_rated`.

switch_on (*name*)

Switch on a socket.

Parameters **name** (*str*) – The socket (entry point) name.

Returns The `MultiPlugAdapter` associated with the `name`.

If the specified `MultiPlugAdapter` already exists, it is returned. If there is no `MultiPlugAdapter` for the specified plugin name, a new one is created and returned.

class **MultiPlugAdapter** (*name*)

The multi-plug adapter that holds all the plugin implementations.

Parameters **name** – The name of the plug adapter (i.e. the name of the entry point).

This class represents all the plugins that implement the give entry point name. Since this class inherit from `RatedDict`, it's possible to rate each implementation. The `pkg_resources` classes allows each distribution to provide their own implementation of a given plugin name: for example, 2 distributions might provide the same `YAML` serialization functions, but each using a different `YAML` library.

`plug_in`(*name*, *plug*)
Try to plug an object in.

Parameters

- **name** (*str*) – The `plug` implementation name.
- **plug** – The object to plug in.

Raises KeyError If another object with the same `name` is already plugghed in.

If you want to explicitly overrid a plug implementation, you must use dictionary item setting syntax.

class **`RatedDict`**
A `dict`-like class that lets you to rate its objects.

This implementation is meant to be thread-safe. Actually, it only supports the following `dict`-like methods:

- `__setitem__`, `__getitem__`, `__delitem__`
- `__contains__`, `__len__`
- `__iter__`, `__reversed__`
- `__str__`
- `update`

`rate`(*updates=()*, ***args*)
Rate the items into the dictionary.

Parameters

- **updates** – A `key: rating` dictionary or an iterable yielding `(key, rating)`.
- **args** (*dict*) – Anyway, variable keyword arguments `args` will be used to update the item ratings.

Raises KeyError If unexpected keys are found.

This method behave like the `dict.update()`, but affects only items ratings. At the end of the update, dictionary keys are sorted by rating, from greater to lower rating value. Rating is supposed to be any kind of number equal or greater than 0. Default item rating is 0.

`top`(*amount=None*)
Returns the top rated items.

Parameters **amount** (*int*) – The number of items to return. Defaults to all items.

Returns A list of `key-value` pairs, sorted by key ratings.

Raises ValueError If `amount` is greater than the available items.

`highest_rated`
The value of the highest rated item.

Raises ValueError If container is empty.

`ratings`
Iterator over `key-rate` pairs, sorted by `rate`.

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*

m

multipla, 7

Index

G

get() (Multipla method), 9

H

highest_rated (RatedDict attribute), 10

M

Multipla (class in multipla), 9
multipla (module), 7
MultiPlugAdapter (class in multipla), 9

P

plug_in() (MultiPlugAdapter method), 9
power_up() (in module multipla), 7

R

rate() (RatedDict method), 10
RatedDict (class in multipla), 10
ratings (RatedDict attribute), 10

S

switch_on() (Multipla method), 9

T

top() (RatedDict method), 10