pluggage Documentation

Release stable

September 10, 2015

Contents

A Utility library for using plugins with python There are two ways to use this library to dynamically load objects, a lightweight load-on-demand via a dictionary interface and a heavier factory pattern using a base class to register objects with a factory.

CHAPTER 1

Lightweight Plugins

The lightweight plugin system uses a dictionary API to load dot-delimited names and either return the object or optionally call it.

```
from pluggage.plugins import Plugins
loader = Plugins()

# load references to objects
func_ref = loader['some_module.some_submodule.some_function']
cls_ref = loader['some_module.some_submodule.SomeClass']

# call the objects and get the results
func_result = loader('some_module.some_submodule.some_function', *func_args, **func_kwargs)
someclass_instance = loader('some_module.some_submodule.SomeClass', *ctor_args, **ctor_kwargs)
```

CHAPTER 2

Heavyweight Plugins

The Heavyweight Plugins are implemented via a metaclass registry that allows classes to be registered as plugins on import, and then accessed via a named factory. Classes inheriting from the base plugin class specify which factory they will be registered with and by default will register under that factory with their class name, or that can also be overridden by inheritance. The plugins can be accessed using a get_factory call provided in the pluggage.registry module.

```
factory = get_factory('my_factory')
some_class_instance = factory('SomeClass', *ctor_args, **ctor_kwargs)
```

To register a class with a factory, inherit from the PluggagePlugin class and set the PLUGGAGE_FACTORY_NAME class attribute:

```
from pluggage.factory_plugin import PluggagePlugin

class SomeClass(PluggagePlugin):
    """
    sample plugin
    """
    PLUGGAGE_FACTORY_NAME = 'my_factory'
    def __init__(self, value):
        self.value = value
    def __call__(self):
        print(value)

factory = get_factory('my_factory')
some_class_instance = factory('SomeClass', 'abc')
some_class_instance()  # prints 'abc'
```