jaraco.services

Release 3.1.dev0+g0e3e497.d20190211

Feb 11, 2019

Contents

CHAPTER 1

History

1.1 3.0

11 Feb 2019

Moved 'envs' functionality to its own package, jaraco.envs.

1.2 2.0

01 Jan 2019

Switch to pkgutil namespace technique for the `jaraco` namespace.

1.3 1.9

09 Nov 2018

Refresh package metadata, including use of declarative config.

Replace deprecated `jaraco.timing` dependency with `tempora`.

1.4 1.8.1

09 Nov 2018

#4: Use `sys.executable` when building tox envs.

1.5 1.8

12 Sep 2018

Added new `envs/ToxEnv` which leverages tox for defining and building environments.

1.6 1.7

06 Sep 2018

#3: Service envs are now created in `.cache/services` by default and no longer consider whether the services are being created in a virtualenv. This allows all services to be created in one flat system. Clients that wish to continue to create services within a virtualenv's root should override the envs.VirtualEnv.root property (on the class or the instance).

1.7 1.6.1

27 Feb 2018

Add missing dependency on virtualenv.

1.8 1.6

22 Feb 2018

Added `jaraco.services.envs` with `VirtualEnv` class.

1.9 1.5.2

29 Mar 2017

#2: Correct scope for `port` reference in HTTPStatus error template.

1.10 1.5.1

27 Jan 2017

#1: Replace use of private `portend._check_port` with call to `portend.free`.

1.11 1.5

11 Jan 2017

In `services.paths`, add `PathFinder.resolve` as a convenience wrapper for including the resolved executable suitable for passing to subprocess.Popen.

1.12 1.4.1

04 Jan 2017

Use `path.Path` for compatibility with path.py 10.

1.13 1.4

17 Aug 2016

Moved project to Github.

1.14 1.3

22 Jul 2015

In HTTPStatus.wait_for_http, allow full timeout for port to be bound.

1.15 1.2

08 Jul 2015

Added `PythonService` class, which will install a Python package into an environment and then launch a process in that environment.

1.16 1.1

06 Jul 2015

Add `HTTPStatus.build_url`. This module provides a Service base class for modeling management of a service, typically launched as a subprocess.

The ServiceManager (deprecated) acts as a collection of interdependent services, can monitor which are running, and will start services on demand. The use case for ServiceManager has been superseded by the more elegant pytest fixtures model.

class jaraco.services.**ServiceManager**(**args*, ***kwargs*)
Bases: `list`

A class that manages services that may be required by some of the unit tests. ServiceManager will start up daemon services as subprocesses or threads and will stop them when requested or when destroyed.

register(*service*)

running

start(*service*)
Start the service, catching and logging exceptions

start_all()
Start all services registered with this manager

start_class(*class_*)
Start all services of a given class. If this manager doesn't already have a service of that class, it constructs one and starts it.

stop(*service*)

stop_all()

stop_class(*class_*)
Stop all services of a given class

class jaraco.services.**Guard**
Bases: object

Prevent execution of a function unless arguments pass self.allowed()

```
>>> class OnlyInts(Guard):
...     def allowed(self, *args, **kwargs):
...         return all(isinstance(arg, int) for arg in args)
>>> @OnlyInts()
... def the_func(val):
...     print(val)
>>> the_func(1)
1
>>> the_func('1')
>>> the_func(1, '1') is None
True
```

allowed(**args*, ***kwargs*)

class jaraco.services.**HTTPStatus**
Bases: object

Mix-in for services that have an HTTP Service for checking the status

build_url(*path*, *host='localhost'*)

proto = 'http'

status_path = '/_status/system'

wait_for_http(*host='localhost'*, *timeout=15*)

class jaraco.services.**Subprocess**
Bases: object

Mix-in to handle common subprocess handling

class PortFree(*port=None*)
Bases: *jaraco.services.Guard*

allowed(*service*, **args*, ***kwargs*)

assert_running()

get_log()

is_external()
A service is external if there's another process already providing this service, typically detected by the port already being occupied.

is_running()

log_root
Much like the property builtin, but only implements __get__, making it a non-data property, and can be subsequently reset.

See http://users.rcn.com/python/download/Descriptor.htm for more information.

```
>>> class X(object):
...   @NonDataProperty
...   def foo(self):
...     return 3
>>> x = X()
>>> x.foo
3
>>> x.foo = 4
>>> x.foo
4
```

stop()

wait_for_pattern(*pattern*, *timeout=5*)

class jaraco.services.**Dependable**(*name*, *bases*, *attribs*)
Bases: type

Metaclass to keep track of services which are depended on by others.

When a class (cls) is created which depends on another (dep), the other gets a reference to cls in its depended_by attribute.

class jaraco.services.**Service**
Bases: object

An abstract base class for services

depends = {}

static find_free_port()

is_running()

static port_free(*port*, *host='localhost'*)

start()

stop()

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

j

Index