daemons Documentation

Release 1.2.1

Kevin Conway

November 11, 2014

Contents

Daemons is a resource library for Python developers that want to create daemon processes. The classes in this library provide the basic daemonization, signal handling, and pid management functionality while allowing for any implementation of behaviour and logic.

CHAPTER 1

Pre-Built Base Classes

Inside of daemons is a sub-package called 'prefab' which contains some useful base classes for building new Python daemons.

1.1 RunDaemon

```
from daemons.prefab import run

class MyDaemon(run.RunDaemon):

    def run(self):

        # Code goes here.
```

The RunDaemon requires only that the 'run' method be implemented. This method must contain some form of a loop to keep the process going. The process stops if the run method ever ends.

1.2 StepDaemon

```
from daemons.prefab import step

class MyDaemon(step.StepDaemon):

    def step(self):

        # Code goes here.
```

If you simply have logic that you want run in an infinite loop use the StepDaemon over the RunDaemon. It automatically handles running your method in a loop. Other than that, it is identical to the RunDaemon.

1.3 Gevent/EventletDaemon

```
from daemons.prefab import gevent

class MyDaemon(gevent.GeventDaemon):

    def get_message(self):
```

```
        # Grab a message from some source and return it.

    def handle_message(self, message):

        # Do something with a message.
```

The gevent and eventlet daemons are specialized for working with message handling. Each one uses a green-thread pool to run the handle_message method. The pool size and idle wait time for these daemons are configurable by passing the 'pool_size' and 'idle_time' kwargs to the intializer.

1.4 Common Features

All deamons in the prefab sub-package share a common set of functionality. The following properties are available on all instances of a daemon class.

- pidfile

  Get the absolute path to the pidfile used by the daemon.

- pid

  Get the pid if the processes is running else None.

- start()

  Method to launch the daemon. This backgrounds the process and exits.

- stop()

  Method to stop a running daemon. Will read from the pidfile and send an appropriate signal.

- restart()

  Helper method which stops then starts the daemon.

- handle(signum, handler)

  Add a function to be run when the process receives the signal identified by 'signum'. The handler is passed no arguments.

- send(signum)

  Send a signal to the process.

1.5 Adding Custom Behaviour

The prefab daemons can be extended and specialized if their implementation is insufficient. Each major feature of a daemon has been broken out into a separate mix-in which provides a standardized interface. The 'interfaces' sub-package contains those standardized interfaces. So long as a class implements the given methods with the given behaviours it can plug in with the other features.

For example implementations check the 'simple' module in the sub-package which matches the feature you want to implement.

1.6 Wrapping Existing Code

If you have a function that you want to daemonize without writing a custom class then there is a potential option in the 'daemonizer' module.

```
from daemons import daemonizer

@daemonizer.run(pidfile="/path/to/somewhere")
def some_func():

    while True:

        # Do something.

some_func()
```

This will daemonize the function using a RunDaemon. A StepDaemon may also be used by decorating a function with '@daemonizer.step'.

For convenience, a 'stop()' method is attached to the function to allow for stopping a running daemon.

```
some_func.stop()
```

Signal handlers can also be added using the decorator by passing in a kwarg called 'signals' which contains a dictionary mapping signal numbers to iterables of functions to run.

```
import signal

@daemonizer.run(
    pidfile="/path/to/somewhere",
    signals={
        signal.SIGTERM: [some_cleanup_function],
    }
)
def some_func():

    while True:

        # Do something

some_func()
```

CHAPTER 2

Indices and tables

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