Test Utils Documentation

Release 0...1...0

Tomaz Muraus

September 23, 2014

Contents

1 Installation **3**

2 API Documentation **5**

3 Process Runner classes **7**

3.1 TCPProcessRunner class 7

Python test utils is a collection of different functions and classes which make writing integration tests easier.

CHAPTER 1

Installation

Latest stable version can be installed from PyPi using pip:

```
pip install test-utils
```

CHAPTER 2

API Documentation

For API documentation, please see the `API Documentation page`.

CHAPTER 3

Process Runner classes

ProcessRunner allows you to manage a long running process which needs to run during your test process execution.

Process runner does this in three steps:

1. Spawn a process before running the tests
2. Wait for the process to come online
3. Run the tests
4. Stop the managed process

This long running process can be an API server, database, Twisted service or any other long running process.

3.1 TCPProcessRunner class

TCPProcessRunner allows you to manage a long running process which exposes a TCP interface. It detects if a process is running by connecting to the specified IP and port.

3.1.1 Example usage

```
# Licensed to Tomaz Muraus under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# Tomaz muraus licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
import os
import sys

from glob import glob
from os.path import splitext, basename, join as pjoin
from distutils.core import Command
```

```
from unittest import TextTestRunner, TestLoader

TEST_PATHS = ['tests']


class TestCommand(Command):
    description = 'run test suite'
    user_options = []

    def initialize_options(self):
        self._dir = os.getcwd()

    def finalize_options(self):
        pass

    def run(self):
        self._run_mock_api_server()
        status = self._run_tests()
        sys.exit(status)

    def _run_tests(self):
        testfiles = []
        for test_path in TEST_PATHS:
            test_files = glob(pjoin(self._dir, test_path, 'test_*.py'))
            for t in test_files:
                module_path = '.'.join([test_path.replace('/', '.'),
                                        splitext(basename(t))[0]])
                testfiles.append(module_path)

        tests = TestLoader().loadTestsFromNames(testfiles)

        t = TextTestRunner(verbosity=2)
        res = t.run(tests)
        return not res.wasSuccessful()

    def _run_mock_api_server(self):
        from test_utils.process_runners import TCPProcessRunner

        script = pjoin(os.path.dirname(__file__), 'tests/mock_http_server.py')

        for port in [8881, 8882, 8883]:
            args = [script, '--port=%s' % (port)]
            log_path = 'mock_api_server_%s.log' % (port)
            wait_for_address = ('127.0.0.1', port)
            server = TCPProcessRunner(args=args,
                                      wait_for_address=wait_for_address,
                                      log_path=log_path)
            server.setUp()
```

This example shows how `TCPProcessRunner` can be used in the `TestCommand` in your `setup.py` file. It is used to start a mock API server which runs for the whole duration of your test suite run.

Keep in mind that you need to call `test_utils.process_runners.TCPProcessRunner.setUp()` function. This function is responsible for starting the managed process and waiting for it to come online.

For a real-life example you can have a look at python-yubico-client setup.py file. In this case, ProcessRunner is used to spawn multiple mock API servers.