pytest-services Documentation

Release 2.2.1

Anatoly Bubenkov, Paylogic International and others

Oct 30, 2020

Contents

1 Services plugin for pytest testing framework **3**
- 1.1 Install pytest-services . . . 3
- 1.2 Features . . . 3
- 1.3 Fixtures . . . 3
- 1.4 Command-line options . . . 6
- 1.5 Example . . . 6
- 1.6 Contact . . . 6
- 1.7 License . . . 6

2 Internal API **7**

3 Authors **9**

4 Changelog **11**
- 4.1 2.2.1 . . . 11
- 4.2 2.2.0 . . . 11
- 4.3 2.1.0 . . . 11
- 4.4 2.0.1 . . . 11
- 4.5 2.0.0 . . . 11
- 4.6 1.3.1 . . . 12
- 4.7 1.3.0 . . . 12
- 4.8 1.2.1 . . . 12
- 4.9 1.2.0 . . . 12
- 4.10 1.1.15 . . . 12
- 4.11 1.1.14 . . . 12
- 4.12 1.1.12 . . . 12
- 4.13 1.1.11 . . . 12
- 4.14 1.1.7 . . . 12
- 4.15 1.1.3 . . . 13
- 4.16 1.1.2 . . . 13
- 4.17 1.1.0 . . . 13
- 4.18 1.0.10 . . . 13
- 4.19 1.0.8 . . . 13
- 4.20 1.0.2 . . . 13
- 4.21 1.0.1 . . . 13
- 4.22 1.0.0 . . . 13

Contents

- *Welcome to pytest-services's documentation!*
- *Services plugin for pytest testing framework*
  - *Install pytest-services*
  - *Features*
  - *Fixtures*
    - *Infrastructure fixtures*
    - *Service fixtures*
    - *Utility functions*
      - *Django settings*
  - *Command-line options*
  - *Example*
  - *Contact*
  - *License*
- *Internal API*
- *Authors*
- *Changelog*
  - *2.2.1*
  - *2.2.0*
  - *2.1.0*
  - *2.0.1*
  - *2.0.0*
  - *1.3.1*
  - *1.3.0*
  - *1.2.1*
  - *1.2.0*
  - *1.1.15*
  - *1.1.14*
  - *1.1.12*
  - *1.1.11*
  - *1.1.7*
  - *1.1.3*
  - *1.1.2*
  - *1.1.0*
  - *1.0.10*

- *1.0.8*
- *1.0.2*
- *1.0.1*
- *1.0.0*

CHAPTER 1

Services plugin for pytest testing framework

1.1 Install pytest-services

```
pip install pytest-services
```

1.2 Features

The plugin provides a set of fixtures and utility functions to start service processes for your tests with pytest

1.3 Fixtures

- **run_services** Determines whether services should be run or not. False by default if not in distributed environment (without pytest-xdist). Can be manually set to True by overriding this fixture in your test config or just by using *–run-services* command line argument (see below).

1.3.1 Infrastructure fixtures

- **worker_id** Id of the worker if tests are run using pytest-xdist. It is set to *local* if tests are not run using pytest-xdist (with *–dist* command line option set to *load*). Has a deprecated alias `slave_id` which will be removed in a future version.
- **session_id** Test session id. Globally unique, and of course also guaranteed to be different for potentially multiple test sessions running on same test node via pytest-xdist.
- **watcher_getter** Function to instantiate test service watcher (popen object). Includes automatic finalization (exiting) of the service process, and testing the service before returning the watcher from the function. Example of usage for memcached service:

```
@pytest.fixture(scope='session')
def memcached(request, run_services, memcached_socket, watcher_getter):
    """The memcached instance which is ready to be used by the tests."""
    if run_services:
        return watcher_getter(
            name='memcached',
            arguments=['-s', memcached_socket],
            checker=lambda: os.path.exists(memcached_socket),
            # Needed for the correct execution order of finalizers
            request=request,
        )
```

- **services_log** Logger used for debug logging when managing test services.
- **root_dir** Parent directory for test service artifacts (disk based). Set to */tmp* by default.
- **base_dir** Base directory for test service artifacts (disk based), unique subdirectory of *root_dir*. Automatically removed recursively at the end of the test session.
- **temp_dir** *Temporary* directory (disk based), subfolder of the *base_dir*. Used for strictly temporary artifacts (for example - folder where files are uploaded from the user input).
- **memory_root_dir** Parent directory for test service artifacts (memory based). Main idea of having memory base directory is to store performance-critical files there. For example - mysql service will use it to store database file, it speeds up mysql server a lot, especially database management operations. Set to */var/shm* by default, with a fallback to 'root_dir'. Note that if apparmor is running on your system, most likely it will prevent your test service to use it (for example - mysql has it's apparmor profile). You you'll need to disable such profile in apparmor configuration. Example of disabling apparmor for mysqld:

```
sudo ln -s /etc/apparmor.d/usr.sbin.mysqld /etc/apparmor.d/disable/
sudo /etc/init.d/apparmor restart
```

- **memory_base_dir** Base directory for test service artifacts (memory based), unique subdirectory of *memory_root_dir*. Automatically removed recursively at the end of the test session.
- **memory_temp_dir** *Temporary* directory (memory based), subfolder of the *base_dir*.
- **lock_dir** Lock files directory for storing locks created for resource assignment (ports, display, etc). Subfolder of *memory_root_dir*.
- **run_dir** Process id and socket files directory (like system-wide */var/run* but local for test session). Subfolder of *memory_root_dir*.
- **port_getter** Function to get unallocated port. Automatically ensures locking and un-locking of it on application level via flock.
- **display_getter** Function to get unallocated display. Automatically ensures locking and un-locking of it on application level via flock.

- **lock_resource_timeout** Used in function lock_resource. A maximum of total sleep between attempts to lock resource.

1.3.2 Service fixtures

- **memcached** Start memcached instance. Requires *pylibmc* installed or *memcache* indicated as an extra (*pip install 'pytest-services[memcached]'*).
- **memcached_socket** Memcached unix socket file name to be used for connection.
- **memcached_connection** Memcached connection string.
- **do_memcached_clean** Determine if memcached should be cleared before every test run. Equals to *run_services* fixture by default. Requires *pylibmc* installed or *memcache* indicated as an extra (*pip install 'pytest-services[memcached]'*).
- **memcached_client** A pylibmc.Client instance bound to the service. Requires *pylibmc* installed or *memcache* indicated as an extra (*pip install 'pytest-services[memcached]'*).
- **mysql** Start mysql-server instance.
- **mysql_database_name** MySQL database name to be created after initialization of the mysql service *system* database.
- **mysql_database_getter** Function with single parameter - database name. To create additional database(s) for tests. Used in *mysql_database* fixture which is used by *mysql* one.
- **mysql_connection** MySQL connection string.
- **xvfb** Start xvfb instance.
- **xvfb_display** Xvfb display to use for connection.
- **xvfb_resolution** Xvfb display resolution to use. Tuple in form *(1366, 768, 8)*.

1.3.3 Utility functions

Django settings

In some cases, there's a need of switching django settings during test run, because several django projects are tested whithin the single test suite. *pytest_services.django_settings* simplifies switching of django settings to a single function call:

- **setup_django_settings** Override the enviroment variable and call the _setup method of the settings object to reload them.

Example of usage:

conftest.py:

```
from pytest_services import django_settings

django_settings.clean_django_settings()
django_settings.setup_django_settings('your.project.settings')
```

Note that the nice project pytest-django doesn't help with the situation, as it's single django project oriented, as well as standard django testing technique. Single project approach works fine, as long as there are no fixtures to share between them, but when there are fixtures to share, then you can get benefit of joining several django projects tests into a single test run, because all session-scoped fixtures will be instantiated only once for all projects tests. The benefit is only visible if you have big enough test suite and your fixtures are heavy enough.

1.4 Command-line options

- ***–run-services*** Force services to be run even if tests are executed in a non-distributed way (without pytest-xdist).
- ***–xvfb-display*** Skip xvfb service to run and use provided display. Useful when you need to run all services except the xvfb to debug your browser tests, if, for example you use pytest-splinter with or without pytest-bdd.

1.5 Example

test_your_test.py:

```
import MySQLdb


def test_some_mysql_stuff(mysql):
    """Test using mysql server."""
    conn = MySQLdb.connect(user='root')
```

1.6 Contact

If you have questions, bug reports, suggestions, etc. please create an issue on the GitHub project page.

1.7 License

This software is licensed under the MIT license

See License file

CHAPTER 2

Internal API

CHAPTER 3

Authors

Anatoly Bubenkov idea and implementation

These people have contributed to *pytest-services*, in alphabetical order:

- Alessio Bogon
- Dmitrijs Milajevs
- Jason R. Coombs
- Joep van Dijken
- Magnus Staberg
- Michael Shriver
- Oleg Pidsadnyi
- Zac Hatfield-Dodds

CHAPTER 4

Changelog

4.1 2.2.1

- #42: Retry on `zc.lockfile.LockError` in `file_lock`, use existing timeout kwarg (mshriver)

4.2 2.2.0

- #38: Retry to lock resource if zc.lockfile.LockError is raised. Fix needed for pytest-xdist. (StabbarN)

4.3 2.1.0

- #34: Deprecated `slave_id` fixture in favor of `worker_id`, for compatibility with `pytest-xdist` 2.

4.4 2.0.1

- #20: Added workaround for issue with SysLogHandler.

4.5 2.0.0

- #23: Rely on `zc.lockfile` for lockfile behavior.
- #28: Fixtures now supports later versions of mysql and no longer support versions of mysql prior to `mysql --initialize` support.
- #29: Fix issues with later versions of mysql where `mysql_defaults_file` fixture would prevent startup of mysql.

- Fixed issue in test suite where mysql fixture was not tested.
- Removed `pytest_services.locks.lock_file`.

4.6 1.3.1

- Fix race condition causing when using port_getter/display_getter (youtux)

4.7 1.3.0

- Add *request* param to watcher_getter to have proper execution order of finalizers (youtux).

4.8 1.2.1

- Swap kill and terminate in watcher_getter finalization, allowing for a more polite SIGTERM for terminating child procs on Unix. See #15 for details (jaraco)

4.9 1.2.0

- Make pylibmc an optional dependency, available as an extra (jaraco)

4.10 1.1.15

- Fixed hang with updated netcat-openbsd>=1.130.3 (joepvandijken)

4.11 1.1.14

- Use a different strategy to determine whether xvfb supports (youtux)

4.12 1.1.12

- use realpath for mysql base dir (bubenkoff)

4.13 1.1.11

- exclude locked displays for xvfb (bubenkoff)

4.14 1.1.7

- django settings fix (olegpidsadnyi)

4.15 1.1.3

- django 1.8 support (bubenkoff)

4.16 1.1.2

- old django support fix (olegpidsadnyi)

4.17 1.1.0

- django 1.7+ support (bubenkoff)

4.18 1.0.10

- removed auto artifacts cleanup (bubenkoff)

4.19 1.0.8

- fixed popen arguments (bubenkoff)

4.20 1.0.2

- added port and display getters (bubenkoff)

4.21 1.0.1

- Improved documentation (bubenkoff)

4.22 1.0.0

- Initial public release