Pytest plugin to simplify running shell commands against the system

Release 1.8.0

Pedro Algarvio

Jul 02, 2023

CONTENTS

1 pytestshellutils package **1**
- 1.1 Subpackages . . . 1
  - 1.1.1 pytestshellutils.utils package . . . 1
- 1.2 Submodules . . . 5
- 1.3 pytestshellutils.customtypes module . . . 5
- 1.4 pytestshellutils.exceptions module . . . 6
- 1.5 pytestshellutils.plugin module . . . 7
- 1.6 pytestshellutils.shell module . . . 7
- 1.7 pytestshellutils.version module . . . 16

2 Changelog **17**
- 2.1 shell-utilities [UNRELEASED DRAFT] (2023-07-02) . . . 17
- 2.2 shell-utilities 1.8.0 (2023-07-02) . . . 17
  - 2.2.1 Breaking Changes . . . 17
  - 2.2.2 Improvements . . . 17
  - 2.2.3 Bug Fixes . . . 17
  - 2.2.4 Trivial/Internal Changes . . . 17
- 2.3 shell-utilities 1.7.0 (2022-09-23) . . . 18
  - 2.3.1 Bug Fixes . . . 18
  - 2.3.2 Trivial/Internal Changes . . . 18
- 2.4 shell-utilities 1.6.0 (2022-07-28) . . . 18
  - 2.4.1 Improvements . . . 18
- 2.5 shell-utilities 1.5.0 (2022-06-02) . . . 18
  - 2.5.1 Improvements . . . 18
  - 2.5.2 Improved Documentation . . . 18
- 2.6 shell-utilities 1.4.0 (2022-05-26) . . . 18
  - 2.6.1 Improvements . . . 18
- 2.7 shell-utilities 1.3.0 (2022-05-26) . . . 18
  - 2.7.1 Improvements . . . 18
- 2.8 shell-utilities 1.2.1 (2022-05-23) . . . 19
  - 2.8.1 Bug Fixes . . . 19
- 2.9 shell-utilities 1.2.0 (2022-05-20) . . . 19
  - 2.9.1 Improvements . . . 19
  - 2.9.2 Trivial/Internal Changes . . . 19
- 2.10 shell-utilities 1.1.0 (2022-05-16) . . . 19
  - 2.10.1 Improvements . . . 19
  - 2.10.2 Trivial/Internal Changes . . . 19
- 2.11 shell-utilities 1.0.5 (2022-02-21) . . . 19
  - 2.11.1 Bug Fixes . . . 19
- 2.12 shell-utilities 1.0.4 (2022-02-17) . . . 20

2.12.1 Improvements . . . 20
2.12.2 Bug Fixes . . . 20
2.13 shell-utilities 1.0.3 (2022-02-16) . . . 20
2.13.1 Bug Fixes . . . 20
2.14 shell-utilities 1.0.2 (2022-02-05) . . . 20
2.14.1 Bug Fixes . . . 20
2.15 shell-utilities 1.0.1 (2022-01-25) . . . 20
2.15.1 Bug Fixes . . . 20
2.16 shell-utilities 1.0.0 (2022-01-25) . . . 20
2.17 shell-utilities 1.0.0rc7 (2022-01-25) . . . 20
2.17.1 Trivial/Internal Changes . . . 20
2.18 shell-utilities 1.0.0rc6 (2022-01-24) . . . 21
2.19 shell-utilities 1.0.0rc5 (2022-01-24) . . . 21
2.19.1 Trivial/Internal Changes . . . 21
2.20 shell-utilities 1.0.0rc4 (2022-01-23) . . . 21
2.21 shell-utilities 1.0.0rc3 (2022-01-21) . . . 21
2.22 shell-utilities 1.0.0rc2 (2022-01-21) . . . 21
2.23 shell-utilities 1.0.0rc1 (2022-01-21) . . . 21

Python Module Index **23**

Index **25**

CHAPTER

ONE

PYTESTSHELLUTILS PACKAGE

1.1 Subpackages

1.1.1 pytestshellutils.utils package

pytestshellutils.utils.**resolved_pathlib_path**(*path: str | Path*) → Path

Return a resolved `pathlib.Path`.

pytestshellutils.utils.**format_callback_to_string**(*callback: str | Callable[[...], Any]*, *args: Tuple[Any, ...] | None = None*, *kwargs: Dict[str, Any] | None = None*) → str

Convert a callback, its arguments and keyword arguments to a string suitable for logging purposes.

Parameters

callback – The callback function

Keyword Arguments

- **args** – The callback arguments
- **kwargs** – The callback keyword arguments

Returns

The formatted callback string

Return type

str

pytestshellutils.utils.**warn_until**(*version: str*, *message: str*, *category: ~typing.Type[Warning] = <class 'DeprecationWarning'>*, *stacklevel: int | None = None*, *_dont_call_warnings: bool = False*, *_pkg_version_: str | None = None*) → None

Show a deprecation warning.

Helper function to raise a warning, by default, a `DeprecationWarning`, until the provided `version`, after which, a `RuntimeError` will be raised to remind the developers to remove the warning because the target version has been reached.

Parameters

- **version** – The version string after which the warning becomes a `RuntimeError`. For example `2.1`.
- **message** – The warning message to be displayed.

Keyword Arguments

- **category** – The warning class to be thrown, by default `DeprecationWarning`
- **stacklevel** – There should be no need to set the value of `stacklevel`.
- **_dont_call_warnings** – This parameter is used just to get the functionality until the actual error is to be issued. When we're only after the version checks to raise a `RuntimeError`.

Returns
Nothing.

Submodules

pytestshellutils.utils.ports module

Ports related utility functions.

`pytestshellutils.utils.ports.`**get_unused_localhost_port**(*use_cache: bool = False*) → int

Return a random unused port on localhost.

Keyword Arguments
use_cache – If `use_cache` is `True`, consecutive calls to this function will never return the cached port.

`pytestshellutils.utils.ports.`**get_connectable_ports**(*ports: Iterable[int]*) → Set[int]

Given a list of ports, returns those that we can connect to.

Parameters
ports – An iterable of ports to try and connect to

Returns
Returns a set of the ports where connection was successful

Return type
set

pytestshellutils.utils.processes module

Process related utilities.

class `pytestshellutils.utils.processes.`**MatchString**

Bases: `str`

Simple subclass around `str` which provides a `.matcher` property.

This `.matcher` property is an instance of `LineMatcher`

property matcher: `LineMatcher`

Return an instance of `LineMatcher`.

`pytestshellutils.utils.processes.`**convert_string_to_match_string**(*value: str | None*) → *MatchString* | None

Convert strings into `MatchString` instances.

class `pytestshellutils.utils.processes.`**ProcessResult**(*, *returncode: int*, *stdout: str | None*, *stderr: str | None*, *cmdline: List[str] | None = None*, *data_key: str | None = None*, *data: Dict[Any, Any] | None = _Nothing.NOTHING*)

Bases: object

Wrapper class around a subprocess result.

This class serves the purpose of having a common result class which will hold the resulting data from a subprocess command.

Keyword Arguments

- **returncode** – The returncode returned by the process
- **stdout** – The `stdout` returned by the process
- **stderr** – The `stderr` returned by the process
- **cmdline** – The command line used to start the process
- **data** – The data returned by parsing `stdout`, when possible.
- **data_key** – When `stdout` can be parsed as JSON, sometimes there's a top level key which is not that interesting. By using `data_key`, we define that we're actually only interested on the data structure which is keyed by `data_key`.

Note: Cast *ProcessResult* to a string to pretty-print it.

---

returncode: int

stdout: *MatchString*

stderr: *MatchString*

cmdline: List[str] | None

data_key: str | None

data: Dict[Any, Any] | None

property exitcode: int

Return the process returncode.

This property is deprecated and should not be used. It only exists to support projects that are migrating from pytest-salt-factories versions. Use `.returncode` instead.

property json: Dict[Any, Any] | None

Return the process output parsed as JSON, if possible.

This property is deprecated and should not be used. It only exists to support projects that are migrating from pytest-salt-factories versions. Use `.data` instead.

__str__() → str

String representation of the class.

pytestshellutils.utils.processes.**collect_child_processes**(*pid: int*) → List[Process]

Try to collect any started child processes of the provided pid.

Parameters
pid – The PID of the process

Returns
List of child processes

pytestshellutils.utils.processes.**terminate_process_list**(*process_list: List[Process]*, *kill: bool = False*, *slow_stop: bool = False*) → None

Terminate a list of processes.

Parameters
process_list – An iterable of `psutil.Process` instances to terminate

Keyword Arguments

- **kill** – Kill the process instead of terminating it.
- **slow_stop** – First try to terminate each process in the list, and if termination was not successful, kill it.

Returns
Nothing.

pytestshellutils.utils.processes.**terminate_process**(*pid: int | None = None*, *process: Process | None = None*, *children: List[Process] | None = None*, *kill_children: bool | None = None*, *slow_stop: bool = False*) → None

Try to terminate/kill the started process.

Keyword Arguments

- **pid** – The PID of the process
- **process** – An instance of `psutil.Process`
- **children** – An iterable of `psutil.Process` instances, children to the process being terminated
- **kill_children** – Also try to terminate/kill child processes
- **slow_stop** – First try to terminate each process in the list, and if termination was not successful, kill it.

pytestshellutils.utils.socket module

Namespace the standard library `socket` module.

This module's sole purpose is to have the standard library `socket` module functions under a different namespace to be used in pytest-shell-utilities so that projects using it, which need to mock `socket` functions, don't influence the pytest-shell-utilities run time behavior.

pytestshellutils.utils.time module

Namespace the standard library `time` module.

This module's sole purpose is to have the standard library `time` module functions under a different namespace to be used in pytest-shell-utilities so that projects using it, which need to mock `time` functions, don't influence the pytest-shell-utilities run time behavior.

1.2 Submodules

1.3 pytestshellutils.customtypes module

Custom Types.

class pytestshellutils.customtypes.**EnvironDict**

Bases: Dict[str, str]

Environ dictionary type.

__str__() → str

String representation of the class.

class pytestshellutils.customtypes.**GenericCallback**(**args*, ***kwargs*)

Bases: Protocol

Generic callback function.

__call__(**args: Any*, ***kwargs: Any*) → None

Call the generic callback.

class pytestshellutils.customtypes.**DaemonCallback**(**args*, ***kwargs*)

Bases: Protocol

Daemon callback function.

__call__(*daemon:* Daemon) → None

Call the daemon callback.

class pytestshellutils.customtypes.**Callback**(**, func: Callable[[...], Any], args: Tuple[Any, ...] | None = None, kwargs: Dict[str, Any] | None = None*)

Bases: object

Class which "stores" information of a callback.

func: Callable[[...], Any]

args: Tuple[Any, ...]

kwargs: Dict[str, Any]

__str__() → str

String representation of the class.

__call__(**args: Any*, ***kwargs: Any*) → Any

Call the callback.

1.4 pytestshellutils.exceptions module

Pytest Shell Utilities related exceptions.

exception pytestshellutils.exceptions.**ShellUtilsException**

Bases: Exception

Base pytest shell utilities exception.

exception pytestshellutils.exceptions.**CallbackException**

Bases: *ShellUtilsException*

Exception raised during a before/after start/stop daemon callback.

exception pytestshellutils.exceptions.**ProcessFailed**(*message: str*, *process_result:* ProcessResult | *None* = *None*)

Bases: *ShellUtilsException*

Exception raised when a sub-process fails.

Parameters
message – The exception message

Keyword Arguments
process_result – The `ProcessResult` instance when the exception occurred

__str__() → str

Return a printable representation of the exception.

exception pytestshellutils.exceptions.**FactoryFailure**(*message: str*, *process_result:* ProcessResult | *None* = *None*)

Bases: *ProcessFailed*

Exception raised when a sub-process fails on one of the factories.

exception pytestshellutils.exceptions.**FactoryNotStarted**(*message: str*, *process_result:* ProcessResult | *None* = *None*)

Bases: *FactoryFailure*

Exception raised when a factory failed to start.

Please look at *FactoryFailure* for the supported keyword arguments documentation.

exception pytestshellutils.exceptions.**FactoryNotRunning**(*message: str*, *process_result:* ProcessResult | *None* = *None*)

Bases: *FactoryFailure*

Exception raised when trying to use a factory's *.stopped* context manager and the factory is not running.

Please look at *FactoryFailure* for the supported keyword arguments documentation.

exception pytestshellutils.exceptions.**ProcessNotStarted**(*message: str*, *process_result:* ProcessResult | *None* = *None*)

Bases: *FactoryFailure*

Exception raised when a process failed to start.

Please look at *FactoryFailure* for the supported keywords. arguments documentation.

exception pytestshellutils.exceptions.**FactoryTimeout**(*message: str*, *process_result:* ProcessResult | *None* = *None*)

Bases: *FactoryNotStarted*

Exception raised when a process timed-out.

Please look at *FactoryFailure* for the supported keywords. arguments documentation.

1.5 pytestshellutils.plugin module

Pytest shell utilities plugin.

pytestshellutils.plugin.**shell**() → *Subprocess*

Shell fixture.

Example

```
def test_assert_good_exitcode(shell):

    ret = shell.run("exit", "0")
    assert ret.returncode == 0


def test_assert_bad_exitcode(shell):

    ret = shell.run("exit", "1")
    assert ret.returncode == 1
```

1.6 pytestshellutils.shell module

Shelling class implementations.

class pytestshellutils.shell.**BaseFactory**(*, *cwd: str* | *Path* = *_Nothing.NOTHING*, *environ:* EnvironDict = *_Nothing.NOTHING*)

Bases: object

Base factory class.

Keyword Arguments

- **cwd** – The path to the desired working directory
- **environ** – A dictionary of `key`, `value` pairs to add to the environment.

cwd: Path

environ: *EnvironDict*

class pytestshellutils.shell.**SubprocessImpl**(*, *factory:* Factory | Subprocess | ScriptSubprocess)

Bases: object

Subprocess interaction implementation.

Parameters
factory – The factory instance, either *Subprocess* or a sub-class of it.

factory: *Factory* | *Subprocess* | *ScriptSubprocess*

cmdline(**args: str*, ***kwargs: Any*) → List[str]

Construct a list of arguments to use when starting the subprocess.

Parameters
args – Additional arguments to use when starting the subprocess

By default, this method will just call it's factory's `cmdline()` method, but can be overridden.

init_terminal(*cmdline: List[str]*, *shell: bool = False*, *env:* EnvironDict | *None = None*, *cwd: str | Path | None = None*) → Popen[Any]

Instantiate a terminal with the passed command line(`cmdline`) and return it.

Additionally, it sets a reference to it in `self._terminal` and also collects an initial listing of child processes which will be used when terminating the terminal

Parameters
cmdline – List of strings to pass as `args` to Popen

Keyword Arguments

- **shell** – Pass the value of `shell` to Popen
- **env** – A dictionary of `key`, `value` pairs to add to the `pytestshellutils.shell.Factory.environ`.
- **cwd** – A path for the CWD when running the process.

Returns
A Popen instance.

is_running() → bool

Returns true if the sub-process is alive.

Returns
Returns true if the sub-process is alive

terminate() → *ProcessResult*

Terminate the started subprocess.

property pid: int | None

The pid of the running process. None if not running.

run(**args: str*, *shell: bool = False*, *env:* EnvironDict | *None = None*, *cwd: str | Path | None = None*, ***kwargs: Any*) → Popen[Any]

Run the given command synchronously.

Parameters
args – The command to run.

Keyword Arguments

- **shell** – Pass the value of *shell* to `pytestshellutils.shell.Factory.init_terminal()`
- **env** – A dictionary of `key`, `value` pairs to add to the `pytestshellutils.shell.Factory.environ`.
- **cwd** – A path for the CWD when running the process.

Returns
A Popen instance.

class pytestshellutils.shell.**Factory**(*, *cwd: str | Path = _Nothing.NOTHING*, *environ: EnvironDict = _Nothing.NOTHING*, *slow_stop: bool = True*, *system_encoding: str = _Nothing.NOTHING*, *timeout: int | float = _Nothing.NOTHING*)

Bases: *BaseFactory*

Base shell factory class.

Keyword Arguments

- **slow_stop** – Whether to terminate the processes by sending a `SIGTERM` signal or by calling `terminate()` on the sub-process. When code coverage is enabled, one will want *slow_stop* set to *True* so that coverage data can be written down to disk.
- **system_encoding** – The system encoding to use when decoding the subprocess output. Defaults to "utf-8".
- **timeout** – The default maximum amount of seconds that a script should run. This value can be overridden when calling `run()` through the `_timeout` keyword argument, and, in that case, the timeout value applied would be that of `_timeout` instead of `self.timeout`.

slow_stop: bool

system_encoding: str

timeout: int | float

impl: *SubprocessImpl*

__attrs_post_init__() → None

Post `attrs` class initialization routines.

cmdline(**args: str*) → List[str]

Method to construct a command line.

get_display_name() → str

Returns a human readable name for the factory.

is_running() → bool

Returns true if the sub-process is alive.

terminate() → *ProcessResult*

Terminate the started subprocess.

property pid: int | None

The pid of the running process. None if not running.

__ne__(*other*)

Method generated by attrs for class Factory.

class pytestshellutils.shell.**Subprocess**(*, *cwd: str | Path = _Nothing.NOTHING*, *environ: EnvironDict = _Nothing.NOTHING*, *slow_stop: bool = True*, *system_encoding: str = _Nothing.NOTHING*, *timeout: int | float = _Nothing.NOTHING*)

Bases: *Factory*

Base shell factory class.

run(**args: str*, *env:* EnvironDict | *None* = *None*, *_timeout: int | float | None* = *None*, ***kwargs: Any*) → *ProcessResult*

Run the given command synchronously.

Keyword Arguments

- **args** – The list of arguments to pass to `cmdline()` to construct the command to run
- **env** – Pass a dictionary of environment key, value pairs to inject into the subprocess.
- **_timeout** – The timeout value for this particular `run()` call. If this value is not `None`, it will be used instead of `timeout`, the default timeout.

process_output(*stdout: str*, *stderr: str*, *cmdline: List[str] | None* = *None*) → Tuple[str, str, Dict[Any, Any] | None]

Process the output. When possible JSON is loaded from the output.

Returns
Returns a tuple in the form of `(stdout, stderr, loaded_json)`

class pytestshellutils.shell.**ScriptSubprocess**(**, *cwd: str | Path* = *_Nothing.NOTHING*, *environ:* EnvironDict = *_Nothing.NOTHING*, *slow_stop: bool* = *True*, *system_encoding: str* = *_Nothing.NOTHING*, *timeout: int | float* = *_Nothing.NOTHING*, *script_name: str*, *base_script_args: List[str]* = *_Nothing.NOTHING*)

Bases: *Subprocess*

Base CLI script/binary class.

Keyword Arguments

- **script_name** – This is the string containing the name of the binary to call on the subprocess, either the full path to it, or the basename. In case of the basename, the directory containing the basename must be in your `$PATH` variable.
- **base_script_args** – An list or tuple iterable of the base arguments to use when building the command line to launch the process

Please look at *Factory* for the additional supported keyword arguments documentation.

script_name: `str`

base_script_args: `List[str]`

get_display_name() → str

Returns a human readable name for the factory.

get_script_path() → str

Returns the path to the script to run.

get_base_script_args() → List[str]

Returns any additional arguments to pass to the CLI script.

get_script_args() → List[str]

Returns any additional arguments to pass to the CLI script.

cmdline(**args: str*) → List[str]

Construct a list of arguments to use when starting the subprocess.

Parameters
args – Additional arguments to use when starting the subprocess

class pytestshellutils.shell.**StartDaemonCallArguments**(*, *args: Tuple[str, ...]*, *kwargs: Dict[str, Any]*)

Bases: object

This class holds the arguments and keyword arguments used to start a daemon.

It's used when restarting the daemon so that the same call is used.

Keyword Arguments

- **args** – List of arguments
- **kwargs** – Dictionary of keyword arguments

args: Tuple[str, ...]

kwargs: Dict[str, Any]

class pytestshellutils.shell.**DaemonImpl**(*, *factory: Daemon*, *before_start_callbacks: List[Callback] = _Nothing.NOTHING*, *after_start_callbacks: List[Callback] = _Nothing.NOTHING*, *before_terminate_callbacks: List[Callback] = _Nothing.NOTHING*, *after_terminate_callbacks: List[Callback] = _Nothing.NOTHING*)

Bases: *SubprocessImpl*

Daemon subprocess interaction implementation.

Please look at *SubprocessImpl* for the additional supported keyword arguments documentation.

factory: *Daemon*

before_start(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run before the daemon starts.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

after_start(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run after the daemon starts.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

before_terminate(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run before the daemon terminates.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

after_terminate(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run after the daemon terminates.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

start(**extra_cli_arguments: str*, *max_start_attempts: int | None = None*, *start_timeout: int | float | None = None*) → bool

Start the daemon.

Keyword Arguments

- **extra_cli_arguments** – Extra arguments to pass to the CLI that starts the daemon
- **max_start_attempts** – Maximum number of attempts to try and start the daemon in case of failures
- **start_timeout** – The maximum number of seconds to wait before considering that the daemon did not start

Returns
A boolean indicating if the start was successful or not.

Return type
bool

terminate() → *ProcessResult*

Terminate the daemon.

get_start_arguments() → *StartDaemonCallArguments*

Return the arguments and keyword arguments used when starting the daemon.

class pytestshellutils.shell.**Daemon**(*, *cwd: str | Path = _Nothing.NOTHING*, *environ: EnvironDict = _Nothing.NOTHING*, *slow_stop: bool = True*, *system_encoding: str = _Nothing.NOTHING*, *timeout: int | float = _Nothing.NOTHING*, *script_name: str*, *base_script_args: List[str] = _Nothing.NOTHING*, *check_ports: List[int] = _Nothing.NOTHING*, *stats_processes: StatsProcesses = None*, *start_timeout: int | float*, *max_start_attempts: int = 3*, *extra_cli_arguments_after_first_start_failure: List[str] = _Nothing.NOTHING*, *start_checks_callbacks: List[Callback] = _Nothing.NOTHING*)

Bases: *ScriptSubprocess*

Base daemon factory.

Keyword Arguments

- **check_ports** – List of ports to try and connect to while confirming that the daemon is up and running
- **extra_cli_arguments_after_first_start_failure** – Extra arguments to pass to the CLI that starts the daemon after the first failure
- **max_start_attempts** – Maximum number of attempts to try and start the daemon in case of failures
- **start_timeout** – The maximum number of seconds to wait before considering that the daemon did not start

Please look at *Subprocess* for the additional supported keyword arguments documentation.

impl: *DaemonImpl*

script_name: str

base_script_args: List[str]

check_ports: List[int]

stats_processes: StatsProcesses

start_timeout: int | float

max_start_attempts: int

extra_cli_arguments_after_first_start_failure: List[str]

listen_ports: List[int]

__attrs_post_init__() → None

Post `attrs` class initialization routines.

before_start(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run before the daemon starts.

Parameters

callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

`after_start`(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run after the daemon starts.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

`before_terminate`(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run before the daemon terminates.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

`after_terminate`(*callback: Callable[[], None]*, **args: Any*, ***kwargs: Any*) → None

Register a function callback to run after the daemon terminates.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

`start_check`(*callback: Callable[[...], bool]*, **args: Any*, ***kwargs: Any*) → None

Register a function to run after the daemon starts to confirm readiness for work.

The callback must accept as the first argument `timeout_at` which is a float. The callback must stop trying to confirm running behavior once `time.time() > timeout_at`. The callback should return `True` to confirm that the daemon is ready for work.

Parameters
callback – The function to call back

Keyword Arguments

- **args** – The arguments to pass to the callback
- **kwargs** – The keyword arguments to pass to the callback

Returns
Nothing.

Example

```
def check_running_state(timeout_at: float) -> bool:
    while time.time() <= timeout_at:
        # run some checks
        ...
        # if all is good
        break
    else:
        return False
    return True
```

get_check_ports() → List[int]

Return a list of ports to check against to ensure the daemon is running.

get_start_check_callbacks() → List[*Callback*]

Return a list of the start check callbacks.

start(**extra_cli_arguments: str*, *max_start_attempts: int | None = None*, *start_timeout: int | float | None = None*) → bool

Start the daemon.

started(**extra_cli_arguments: str*, *max_start_attempts: int | None = None*, *start_timeout: int | float | None = None*) → Generator[*Daemon*, None, None]

Start the daemon and return it's instance so it can be used as a context manager.

stopped(*before_stop_callback: Callable[[*Daemon*], None] | None = None*, *after_stop_callback: Callable[[*Daemon*], None] | None = None*, *before_start_callback: Callable[[*Daemon*], None] | None = None*, *after_start_callback: Callable[[*Daemon*], None] | None = None*) → Generator[*Daemon*, None, None]

Stop the daemon and return it's instance so it can be used as a context manager.

Keyword Arguments

- **before_stop_callback** – A callable to run before stopping the daemon. The callback must accept one argument, the daemon instance.
- **after_stop_callback** – A callable to run after stopping the daemon. The callback must accept one argument, the daemon instance.
- **before_start_callback** – A callable to run before starting the daemon. The callback must accept one argument, the daemon instance.
- **after_start_callback** – A callable to run after starting the daemon. The callback must accept one argument, the daemon instance.

This context manager will stop the factory while the context is in place, it re-starts it once out of context.

Example

```
assert factory.is_running() is True

with factory.stopped():
    assert factory.is_running() is False
```

(continues on next page)

(continued from previous page)

```
assert factory.is_running() is True
```

run_start_checks(*started_at: float*, *timeout_at: float*) → bool

Run checks to confirm that the daemon has started.

__enter__() → *Daemon*

Use class as a context manager.

__exit__(**_: Any*) → None

Exit the class context manager.

1.7 pytestshellutils.version module

CHAPTER

TWO

CHANGELOG

Versions follow Semantic Versioning (*<major>.<minor>.<patch>*).

Backward incompatible (breaking) changes will only be introduced in major versions with advance notice in the **Deprecations** section of releases.

2.1 shell-utilities [UNRELEASED DRAFT] (2023-07-02)

No significant changes.

2.2 shell-utilities 1.8.0 (2023-07-02)

2.2.1 Breaking Changes

- Drop support for python versions older than 3.7 (#38)

2.2.2 Improvements

- Support Python 3.11 (#40)

2.2.3 Bug Fixes

- Set minimal attrs version to 22.1.0 (#28)

2.2.4 Trivial/Internal Changes

- Update the GitHub actions versions and stop using *::set-output* (#38)
- Several project internal changes
  - Start running tests against Py3.11 and Pytest *7.3.x* and *7.4.x*
  - Update copyright headers
  - Upgrade to *coverage==7.2.7*
  - Switch to *codecov/codecov-action* (#39)

2.3 shell-utilities 1.7.0 (2022-09-23)

2.3.1 Bug Fixes

- `Subprocess.run()` now accepts `shell` keyword argument like `subprocess.Popen`. (#32)
- The *Subprocess.run()* method can now override the *cwd* (#33)

2.3.2 Trivial/Internal Changes

- Update pre-commit hook versions (#34)

2.4 shell-utilities 1.6.0 (2022-07-28)

2.4.1 Improvements

- The `shell` fixture is now `session` scoped (#29)

2.5 shell-utilities 1.5.0 (2022-06-02)

2.5.1 Improvements

- The minimum python for the code base is now 3.7(we still provide support to Py3.5 and Py3.6 by providing a downgraded source, transparent to the user), and the project is now 100% typed, including the test suite. (#26)

2.5.2 Improved Documentation

- Improve and switch to google style docstrings (#24)

2.6 shell-utilities 1.4.0 (2022-05-26)

2.6.1 Improvements

- `Daemon.started()` is now a context manager (#22)

2.7 shell-utilities 1.3.0 (2022-05-26)

2.7.1 Improvements

- Support user provided callable functions to confirm that the daemon is up and running (#20)

2.8 shell-utilities 1.2.1 (2022-05-23)

2.8.1 Bug Fixes

- Account for `ProcessLookupError` when terminating the underlying process. (#18)

2.9 shell-utilities 1.2.0 (2022-05-20)

2.9.1 Improvements

- Revert "Skip test when the GLIBC race conditions are met, instead of failing."

  It wasn't the right fix/workaround. The right fix can be seen in the Salt repo (#16)

2.9.2 Trivial/Internal Changes

- Remove the redundant *wheel* dependency from pyproject.toml.

  The setuptools backend takes care of adding it automatically via *setuptools.build_meta.get_requires_for_build_wheel()* since day one. The documentation has historically been wrong about listing it, and it has been fixed since.

  See https://github.com/pypa/setuptools/commit/f7d30a9529378cf69054b5176249e5457aaf640a (#15)

2.10 shell-utilities 1.1.0 (2022-05-16)

2.10.1 Improvements

- Skip test when the GLIBC race conditions are met, instead of failing (#13)

2.10.2 Trivial/Internal Changes

- Update pre-commit hooks and test against PyTest 7.0.x and 7.1.x. (#13)

2.11 shell-utilities 1.0.5 (2022-02-21)

2.11.1 Bug Fixes

- Fix deprecation message telling to use the wrong property. (#12)

2.12 shell-utilities 1.0.4 (2022-02-17)

2.12.1 Improvements

- State from which library the `DeprecationWarning` is coming from. (#9)

2.12.2 Bug Fixes

- Handle `None` values for `.stdout` and `.stderr` on `ProcessResult.__str__()` (#8)

2.13 shell-utilities 1.0.3 (2022-02-16)

2.13.1 Bug Fixes

- Fixed issue with `sdist` recompression for reproducible packages not iterating though subdirectories contents. (#7)

2.14 shell-utilities 1.0.2 (2022-02-05)

2.14.1 Bug Fixes

- Set lower required python to *3.5.2* and avoid issues with *flake8-typing-imports*. (#6)

2.15 shell-utilities 1.0.1 (2022-01-25)

2.15.1 Bug Fixes

- Stop casting `None` to a string for `ProcessResult.std{out,err}` (#4)

2.16 shell-utilities 1.0.0 (2022-01-25)

No significant changes.

2.17 shell-utilities 1.0.0rc7 (2022-01-25)

2.17.1 Trivial/Internal Changes

- Improvements before final RC
  - Add `ProcessResult.std{out,err}.matcher` example
  - Also generate reproducible packages when uploading a release to pypi
  - The `twine-check` nox target now call's the `build` target (#3)

2.18 shell-utilities 1.0.0rc6 (2022-01-24)

No significant changes.

2.19 shell-utilities 1.0.0rc5 (2022-01-24)

2.19.1 Trivial/Internal Changes

- Provide a way to create reproducible distribution packages.
  - Stop customizing the `towncrier` template. (#1)

2.20 shell-utilities 1.0.0rc4 (2022-01-23)

- `ProcessResult.stdout` and `ProcessResult.stderr` are now instances of `pytestshellutils.utils.processes.MatchString` which provides a `.matcher` attribute that returns an instance of `pytest.LineMatcher`.

2.21 shell-utilities 1.0.0rc3 (2022-01-21)

- `cwd` and `environ` are now defined on `BaseFactory`
- Add `py.typed` to state that the package is fully typed
- Fix the `stacklevel` value to point to the actual caller of the `warn_until` function.
- Fix the deprecated `ProcessResult.json` property.

2.22 shell-utilities 1.0.0rc2 (2022-01-21)

- When passed a string, cast it to `pathlib.Path` before calling `.resolve()`
- Extract `BaseFactory` from `Factory`. It's required on pytest-salt-factories container implementation.

2.23 shell-utilities 1.0.0rc1 (2022-01-21)

Pre-release of the first working version of the pytest plugin.

PYTHON MODULE INDEX

p

pytestshellutils, 1
pytestshellutils.customtypes, 5
pytestshellutils.exceptions, 6
pytestshellutils.plugin, 7
pytestshellutils.shell, 7
pytestshellutils.utils, 1
pytestshellutils.utils.ports, 2
pytestshellutils.utils.processes, 2
pytestshellutils.utils.socket, 4
pytestshellutils.utils.time, 4
pytestshellutils.version, 16

INDEX

Symbols

__attrs_post_init__() (*pytestshellutils.shell.Daemon method*), 13
__attrs_post_init__() (*pytestshellutils.shell.Factory method*), 9
__call__() (*pytestshellutils.customtypes.Callback method*), 5
__call__() (*pytestshellutils.customtypes.DaemonCallback method*), 5
__call__() (*pytestshellutils.customtypes.GenericCallback method*), 5
__enter__() (*pytestshellutils.shell.Daemon method*), 16
__exit__() (*pytestshellutils.shell.Daemon method*), 16
__ne__() (*pytestshellutils.shell.Factory method*), 9
__str__() (*pytestshellutils.customtypes.Callback method*), 5
__str__() (*pytestshellutils.customtypes.EnvironDict method*), 5
__str__() (*pytestshellutils.exceptions.ProcessFailed method*), 6
__str__() (*pytestshellutils.utils.processes.ProcessResult method*), 3

A

after_start() (*pytestshellutils.shell.Daemon method*), 14
after_start() (*pytestshellutils.shell.DaemonImpl method*), 11
after_terminate() (*pytestshellutils.shell.Daemon method*), 14
after_terminate() (*pytestshellutils.shell.DaemonImpl method*), 12
args (*pytestshellutils.customtypes.Callback attribute*), 5
args (*pytestshellutils.shell.StartDaemonCallArguments attribute*), 11

B

base_script_args (*pytestshellutils.shell.Daemon attribute*), 13
base_script_args (*pytestshellutils.shell.ScriptSubprocess attribute*), 10
BaseFactory (*class in pytestshellutils.shell*), 7
before_start() (*pytestshellutils.shell.Daemon method*), 13
before_start() (*pytestshellutils.shell.DaemonImpl method*), 11
before_terminate() (*pytestshellutils.shell.Daemon method*), 14
before_terminate() (*pytestshellutils.shell.DaemonImpl method*), 11

C

Callback (*class in pytestshellutils.customtypes*), 5
CallbackException, 6
check_ports (*pytestshellutils.shell.Daemon attribute*), 13
cmdline (*pytestshellutils.utils.processes.ProcessResult attribute*), 3
cmdline() (*pytestshellutils.shell.Factory method*), 9
cmdline() (*pytestshellutils.shell.ScriptSubprocess method*), 10
cmdline() (*pytestshellutils.shell.SubprocessImpl method*), 8
collect_child_processes() (*in module pytestshellutils.utils.processes*), 3
convert_string_to_match_string() (*in module pytestshellutils.utils.processes*), 2
cwd (*pytestshellutils.shell.BaseFactory attribute*), 7

D

Daemon (*class in pytestshellutils.shell*), 12
DaemonCallback (*class in pytestshellutils.customtypes*), 5
DaemonImpl (*class in pytestshellutils.shell*), 11
data (*pytestshellutils.utils.processes.ProcessResult attribute*), 3
data_key (*pytestshellutils.utils.processes.ProcessResult attribute*), 3

E

environ (*pytestshellutils.shell.BaseFactory attribute*), 7

EnvironDict (*class in pytestshellutils.customtypes*), 5
exitcode (*pytestshellutils.utils.processes.ProcessResult property*), 3
extra_cli_arguments_after_first_start_failure (*pytestshellutils.shell.Daemon attribute*), 13

F

Factory (*class in pytestshellutils.shell*), 9
factory (*pytestshellutils.shell.DaemonImpl attribute*), 11
factory (*pytestshellutils.shell.SubprocessImpl attribute*), 8
FactoryFailure, 6
FactoryNotRunning, 6
FactoryNotStarted, 6
FactoryTimeout, 6
format_callback_to_string() (*in module pytestshellutils.utils*), 1
func (*pytestshellutils.customtypes.Callback attribute*), 5

G

GenericCallback (*class in pytestshellutils.customtypes*), 5
get_base_script_args() (*pytestshellutils.shell.ScriptSubprocess method*), 10
get_check_ports() (*pytestshellutils.shell.Daemon method*), 15
get_connectable_ports() (*in module pytestshellutils.utils.ports*), 2
get_display_name() (*pytestshellutils.shell.Factory method*), 9
get_display_name() (*pytestshellutils.shell.ScriptSubprocess method*), 10
get_script_args() (*pytestshellutils.shell.ScriptSubprocess method*), 10
get_script_path() (*pytestshellutils.shell.ScriptSubprocess method*), 10
get_start_arguments() (*pytestshellutils.shell.DaemonImpl method*), 12
get_start_check_callbacks() (*pytestshellutils.shell.Daemon method*), 15
get_unused_localhost_port() (*in module pytestshellutils.utils.ports*), 2

I

impl (*pytestshellutils.shell.Daemon attribute*), 13
impl (*pytestshellutils.shell.Factory attribute*), 9
init_terminal() (*pytestshellutils.shell.SubprocessImpl method*), 8
is_running() (*pytestshellutils.shell.Factory method*), 9
is_running() (*pytestshellutils.shell.SubprocessImpl method*), 8

J

json (*pytestshellutils.utils.processes.ProcessResult property*), 3

K

kwargs (*pytestshellutils.customtypes.Callback attribute*), 5
kwargs (*pytestshellutils.shell.StartDaemonCallArguments attribute*), 11

L

listen_ports (*pytestshellutils.shell.Daemon attribute*), 13

M

matcher (*pytestshellutils.utils.processes.MatchString property*), 2
MatchString (*class in pytestshellutils.utils.processes*), 2
max_start_attempts (*pytestshellutils.shell.Daemon attribute*), 13
module
    pytestshellutils, 1
    pytestshellutils.customtypes, 5
    pytestshellutils.exceptions, 6
    pytestshellutils.plugin, 7
    pytestshellutils.shell, 7
    pytestshellutils.utils, 1
    pytestshellutils.utils.ports, 2
    pytestshellutils.utils.processes, 2
    pytestshellutils.utils.socket, 4
    pytestshellutils.utils.time, 4
    pytestshellutils.version, 16

P

pid (*pytestshellutils.shell.Factory property*), 9
pid (*pytestshellutils.shell.SubprocessImpl property*), 8
process_output() (*pytestshellutils.shell.Subprocess method*), 10
ProcessFailed, 6
ProcessNotStarted, 6
ProcessResult (*class in pytestshellutils.utils.processes*), 2
pytestshellutils
    module, 1
pytestshellutils.customtypes
    module, 5
pytestshellutils.exceptions
    module, 6
pytestshellutils.plugin
    module, 7
pytestshellutils.shell
    module, 7
pytestshellutils.utils

module, 1
pytestshellutils.utils.ports
module, 2
pytestshellutils.utils.processes
module, 2
pytestshellutils.utils.socket
module, 4
pytestshellutils.utils.time
module, 4
pytestshellutils.version
module, 16

R

resolved_pathlib_path() (*in module pytestshellutils.utils*), 1
returncode (*pytestshellutils.utils.processes.ProcessResult attribute*), 3
run() (*pytestshellutils.shell.Subprocess method*), 9
run() (*pytestshellutils.shell.SubprocessImpl method*), 8
run_start_checks() (*pytestshellutils.shell.Daemon method*), 16

S

script_name (*pytestshellutils.shell.Daemon attribute*), 13
script_name (*pytestshellutils.shell.ScriptSubprocess attribute*), 10
ScriptSubprocess (*class in pytestshellutils.shell*), 10
shell() (*in module pytestshellutils.plugin*), 7
ShellUtilsException, 6
slow_stop (*pytestshellutils.shell.Factory attribute*), 9
start() (*pytestshellutils.shell.Daemon method*), 15
start() (*pytestshellutils.shell.DaemonImpl method*), 12
start_check() (*pytestshellutils.shell.Daemon method*), 14
start_timeout (*pytestshellutils.shell.Daemon attribute*), 13
StartDaemonCallArguments (*class in pytestshellutils.shell*), 10
started() (*pytestshellutils.shell.Daemon method*), 15
stats_processes (*pytestshellutils.shell.Daemon attribute*), 13
stderr (*pytestshellutils.utils.processes.ProcessResult attribute*), 3
stdout (*pytestshellutils.utils.processes.ProcessResult attribute*), 3
stopped() (*pytestshellutils.shell.Daemon method*), 15
Subprocess (*class in pytestshellutils.shell*), 9
SubprocessImpl (*class in pytestshellutils.shell*), 7
system_encoding (*pytestshellutils.shell.Factory attribute*), 9

T

terminate() (*pytestshellutils.shell.DaemonImpl method*), 12
terminate() (*pytestshellutils.shell.Factory method*), 9
terminate() (*pytestshellutils.shell.SubprocessImpl method*), 8
terminate_process() (*in module pytestshellutils.utils.processes*), 4
terminate_process_list() (*in module pytestshellutils.utils.processes*), 3
timeout (*pytestshellutils.shell.Factory attribute*), 9

W

warn_until() (*in module pytestshellutils.utils*), 1