pytest-logger

Release 0.5.0

May 07, 2019

Contents

The pytest-logger is a pytest plugin which handles logs emitted by Python's logging module. Logs are output to terminal or files in configurable manner.

CHAPTER 1

Overview

1.1 Installation

As simple as:

```
$ [sudo] pip install pytest-logger
```

Project is hosted on github:

https://github.com/aurzenligl/pytest-logger

1.2 Rationale

I work with C++ application which logs copiously on its own and has multiple interfaces on which interesting events occur. Getting all this information as separate files in directory tree or filtered real-time logs proves to be invaluable in testing.

Unfortunately, contemporary state of pytest and plugins doesn't allow to do this out-of-the-box:

- real-time output to terminal which doesn't mix with regular pytest output ('-v' or not)
- possibility to enable and set levels on per logger basis
- test session logs persistent in filesystem and partitioned in fine-grained manner (per testcase and per logger)

Above problems require reacting on events such as session start, test start/teardown and inspecting some data stored by framework, e.g. test locations/names. This requires writing pytest plugin.

Plugin has a hook API, which means that if user doesn't implement hooks, nothing happens, and if he does - any cmdline options and logging configuration logic may be envisioned.

1.3 Contributing

Contibutions are welcome!

If you:

- find a bug in plugin,
- find important features missing,
- want to propose a new feature,
- or just like it,

please write to github issues or let me know via email.

CHAPTER 2

Usage

By default pytest-logger does nothing in scope of logging. It's only default action is to add new hooks and fixture. Terminal output, logfiles together with their directory structure, symlink appear only when explicitly requested by user via one of hooks or fixture.

2.1 Logging to stdout or files

Implement pytest-logger hooks in your conftest.py to direct logs to terminal or files.

Terminal output mixes with normal pytest's output in graceful manner. File output is stored in logs directory (see *logs dir layout* and *link to logs dir*).

There are two ways of doing this. *First one* simpler to use, *second one* more flexible. They cannot be mixed in given test session.

Things to note:

- **stdout capturing:** in order to see logs printed on terminal in real time during test execution, you need to disable output capturing by `-s` switch.
- **default root level:** by default root logger (and implicitly all its children) has warning level threshold set. If any logger via any hook is configured, root logger level will be set to NOTSET to pass all logs according to levels set by pytest_logger user.
- **no handlers warning:** if log wouldn't get filtered, but there are no handlers added to logger, unwanted message is printed. Add NullHandler to such loggers.
- **default logging config:** if root logger has no handlers, using module level logging functions will setup basic logging config. It makes no sense in combintion with this plugin. Be sure root logger has some handler (at least NullHandler) or just don't use these functions.
- **pytest-xdist:** stdout output is not printed to terminal in pytest-xdist runs. File output works as in single process mode.

2.1.1 High-level hook

```
def pytest_logger_config(logger_config):
    # adds two loggers, which will:
    #   - log to filesystem at all levels
    #   - log to terminal with default WARN level if provided in --loggers option
    logger_config.add_loggers(['foo', 'bar'], stdout_level='warn')

    # default --loggers option is set to log foo at WARN level and bar at NOTSET
    logger_config.set_log_option_default('foo,bar.notset')
```

- command line option *–loggers* is added.
- see *LoggerHookspec.pytest_logger_config()*
- note that *LoggerConfig.set_formatter_class()* can be used to set a custom `logging.Formatter` class

2.1.2 Low-level hooks

```
import logging

def pytest_logger_stdoutloggers(item):
    # handles foo logger at chosen level and bar at all levels
    return [('foo', logging.WARN), 'bar']

def pytest_logger_fileloggers(item):
    # handles root logger
    return ['']
```

- no command line options are added
- see *LoggerHookspec.pytest_logger_stdoutloggers()*
- see *LoggerHookspec.pytest_logger_fileloggers()*

2.2 The logs directory layout

Directory with logfiles is located

- under test session's basetemp directory named "logs"

```
tmp/
└── pytest-of-aurzenligl
    ├── pytest-0
    │   └── logs
    │       (...)
    ├── pytest-1
    │   └── logs
    │       (...)
    └── pytest-2
        └── logs
            (...)
```

or

- under predefined location, if *–logger-logsdir* option or *logger_logsdir* entry in configuration file defined

```
<logger_logsdir>/
└── (...)
```

It has structure following pytest test item's nodeid.

- test directories are directories
- test filenames are directories
- test classes are directories
- test functions are directories (each parametrized testcase variant is distinct)
- each registered logger is a file (root logger is called 'root')

```
logs/
├── classtests
│   └── test_y.py
│       └── TestClass
│           └── test_class
│               ├── daemon
│               └── setup
├── parametrictests
│   └── test_z.py
│       ├── test_param-2-abc
│       └── test_param-4.127-de
│           └── setup
└── test_p.py
    └── test_cat
        └── proc
```

2.3 Split logs by outcome

It is possible to split the logs by test outcome. If chosen to do so (by calling below method):

```
# content of conftest.py
def pytest_logger_config(logger_config):
    logger_config.split_by_outcome()
```

Will result in below directory structure:

```
logs/
├── classtests
│   └── test_y.py
│       └── TestClass
│           ├── test_class
│           |   ├── daemon
│           |   └── setup
|           └── test_that_failed_two
|               └── somelogfile
├── by_outcome
|   └── failed
|       ├── classtests
|       |   └── test_y.py
|       |       └── TestClass
```

(continues on next page)

(continued from previous page)

```
|       |           └── test_that_failed_two -> ../../../../../../classtests/test_y.
↪py/TestClass/test_that_failed_two
|       └── test_p.py
|           └── test_that_failed_one -> ../../../../test_p.py/test_that_failed_one
└── test_p.py
    ├── test_cat
    |   └── proc
    └── test_that_failed_one
        └── somelog
```

You can change the default *by_outcome* dirname to something else, as well as add more "per-outcome" subdirectories by passing proper arguments to the *split_by_outcome* method.

2.4 Link to logs directory

Implement link hook to have access to logfiles from place where you regularly run your tests. Link is created in a race-safe manner, even if multiple pytest processes run tests simultaneously.

```
# content of conftest.py
import os
def pytest_logger_logdirlink(config):
    return os.path.join(os.path.dirname(__file__), 'logs')
```

```
$ ls -o
total 80
drwxr-xr-x  9 aurzenligl 4096 Dec 22 21:09 .
drwxr-xr-x 28 aurzenligl 4096 Dec 14 23:33 ..
-rwxr-xr-x  1 aurzenligl 3028 Dec 11 02:18 conftest.py
lrwxrwxrwx  1 aurzenligl   39 Dec 22 21:09 logs -> /tmp/pytest-of-aurzenligl/pytest-2/
↪logs
-rwxr-xr-x  1 aurzenligl  817 Dec 11 02:13 test_x.py
```

- see *LoggerHookspec.pytest_logger_logdirlink()*

2.5 The logdir fixture

Like pytest's tmpdir it's a py.path.local object which offers os.path methods. Points to logs subdirectory related to particular testcase. Directory is ensured to exist and custom log files can be written into it:

```
def test_foo(logdir):
    logdir.join('myfile.txt').write('abc')
```

2.6 API reference

class pytest_logger.plugin.**LoggerHookspec**

pytest_logger_config(*logger_config*)
called before cmdline options parsing. Accepts terse configuration of both stdout and file logging, adds cmdline options to manipulate stdout logging. Cannot be used together with *loggers hooks.

Parameters **logger_config** – instance of *LoggerConfig*, allows setting loggers for stdout and file handling and their levels.

pytest_logger_stdoutloggers(*item*)
called before testcase setup. If implemented, given loggers will emit their output to terminal output. Cannot be used together with logger_config hook.

Parameters **item** – test item for which handlers are to be setup.

Return list List should contain logger name strings or tuples with logger name string and logging level.

pytest_logger_fileloggers(*item*)
called before testcase setup. If implemented, given loggers will emit their output to files within logs temporary directory. Cannot be used together with logger_config hook.

Parameters **item** – test item for which handlers are to be setup.

Return list List should contain logger name strings or tuples with logger name string and logging level.

pytest_logger_logdirlink(*config*)
called after cmdline options parsing. If implemented, symlink to logs directory will be created.

Parameters **config** – pytest config object, holds e.g. options

Return string Absolute path of requested link to logs directory.

class pytest_logger.plugin.**LoggerConfig**
Configuration of logging to stdout and filesystem.

add_loggers(*loggers*, *stdout_level=0*, *file_level=0*)
Adds loggers for stdout/filesystem handling.

Stdout: loggers will log to stdout only when mentioned in *loggers* option. If they're mentioned without explicit level, *stdout_level* will be used.

Filesystem: loggers will log to files at *file_level*.

Parameters

- **loggers** – List of logger names.
- **stdout_level** – Default level at which stdout handlers will pass logs. By default: *logging.NOTSET*, which means: pass everything.
- **file_level** – Level at which filesystem handlers will pass logs. By default: *logging.NOTSET*, which means: pass everything.

set_formatter_class(*formatter_class*)
Sets the *logging.Formatter* class to be used by all loggers.

Parameters **formatter_class** – The *logging.Formatter* class

set_log_option_default(*value*)
Sets default value of *log* option.

split_by_outcome(*outcomes=None*, *subdir='by_outcome'*)
Makes a directory inside main logdir where logs are further split by test outcome

Parameters

- **subdir** – name for the subdirectory in main log directory
- **outcomes** – list of test outcomes to be handled (failed/passed/skipped)

2.7 Command line options

–logger-logsdir=<logsdir> where <logsdir> is root directory where log files are created

–loggers=<loggers> where <loggers> are a comma delimited list of loggers optionally suffixed with level preceded by a dot. Levels can be lower or uppercase, or numeric. For example: "logger1,logger2.info,logger3.FATAL,logger4.25"

2.8 Configuration file parameters

logger_logsdir=<logsdir> where <logsdir> is root directory where log files are created

Index