quicklogging Documentation

Release 0.2

Feth AREZKI

May 02, 2022

Contents

1 Resources **1**

2 What is quicklogging **3**

3 Licence, original authors **5**

4 Requirements **7**
- 4.1 Python versions . . . 7
- 4.2 Libs required . . . 7

5 Doc contents **9**
- 5.1 How to use . . . 9
- 5.2 Task oriented doc . . . 10
- 5.3 Source doc . . . 11
- 5.4 Annex: Quick survival guide with the logging module . . . 13

Python Module Index **15**

Index **17**

CHAPTER 1

Resources

- doc: - read the doc on readthedocs (github rst is broken)
- source code: on github
- package:
- ci tests:

CHAPTER 2

What is quicklogging

quicklogging is a Python `logging` wrapper to

- remove a bit of logging boilerplate,
- redirect print output.

`quicklogging` transparently provides a logger with a name relevant to the code at hand:

Important: The name of the logger is the name of the module making the call.

For instance, if you log from `project/models/chair.py`, your logger will be named `project.models.chair`.

This is a very important feature:

Advantage #1 of this naming scheme

> the *configuration* of the `logging.Logger` s and handlers is much easier —muting, changing verbosity for a particular piece of code etc

Advantage #2

> we can provide a *`quicklogging.catch_prints()`* and a *`quicklogging.warn_prints()`* functionality to catch calls to print() from specific modules (typically: the module you're editing).

CHAPTER 3

Licence, original authors

- MIT (see file `LICENCE`).
- authors: majerti - Feth AREZKI

CHAPTER 4

Requirements

4.1 Python versions

I tested 3.5 to 3.10 without annoyances.

Cannot test, but should work because I don't know of API changes:

- Python 2.7: *testing* NOT ok (Python 2.7 doesn't have `importlib.abc.SourceLoader`

4.2 Libs required

None ! unless you're running the tests (then you need stringimporter). . So I've got this easy badge: .

CHAPTER 5

Doc contents

(if toctree is not displayed, then build the docs with sphinx or read them on readthedocs : .

5.1 How to use

The following examples assume a log format of

```
[%(name)s - %(level)s] %(message)s
```

You may want to read the *Annex: Quick survival guide with the logging module* to achieve this.

5.1.1 Quick, log once

Say you're in `myapp/models/music.py`

```
import quicklogging
quicklogging.error("Hello world")
```

Your output will be:

```
[myapp.models.music - ERROR] Hello world
```

5.1.2 Log twice

Actually you're logging more than once in `myapp/views/music.py` and want to optimize:

```
import quicklogging
logger = quicklogging.get_logger()
logger.debug("Howdy?")
```

This produces:

```
[myapp.views.music - DEBUG] Howdy?
```

then,

```
logger.warning("plop")
```

produces:

```
[myapp.views.music - WARNING] plop
```

5.2 Task oriented doc

Note: Many functions take a *stackoverhead* param, which is documented in *quicklogging.base.get_logger()*.

5.2.1 Fetch a logger

Use *quicklogging.base.get_logger()* also available as `quicklogging.get_logger()`.

Then, you can use the logger normally.

5.2.2 Available log wrappers

If logging multiple times, it is slower to use these than to use `quicklogging.get_logger()` and log with received object.

- *quicklogging.debug()* → `logging.debug()`
- *quicklogging.info()* → `logging.info()`
- *quicklogging.warning()* → `logging.warning()`
- *quicklogging.error()* → `logging.error()`
- *quicklogging.critical()* → `logging.critical()`
- *quicklogging.exception()* → `logging.exception()`

5.2.3 print wrapping functionality

`quicklogging` provides

- *quicklogging.catch_prints()*
- *quicklogging.warn_prints()*

5.3 Source doc

5.3.1 Module quicklogging

(logging with a bit of cozyness)

log wrappers

Supplied convenience functions fetch a logger with the name of the module from which you're calling them.

quicklogging.**debug**(**args*, ***kwargs*)
wrapper for logging.Logger.debug()

Variadic parameters: see logging.Logger.debug()

Parameters **stackoverhead** (*int*) – see *quicklogging.base.get_logger()*

quicklogging.**info**(**args*, ***kwargs*)
wrapper for logging.Logger.info()

Variadic parameters: see logging.Logger.info()

Parameters **stackoverhead** (*int*) – see *quicklogging.base.get_logger()*

quicklogging.**warning**(**args*, ***kwargs*)
wrapper for logging.Logger.warning()

Variadic parameters: see logging.Logger.warning()

Parameters **stackoverhead** (*int*) – see *quicklogging.base.get_logger()*

quicklogging.**error**(**args*, ***kwargs*)
wrapper for logging.Logger.error()

Variadic parameters: see logging.Logger.error()

Parameters **stackoverhead** (*int*) – see *quicklogging.base.get_logger()*

quicklogging.**critical**(**args*, ***kwargs*)
wrapper for logging.Logger.critical()

Variadic parameters: see logging.Logger.critical()

Parameters **stackoverhead** (*int*) – see *quicklogging.base.get_logger()*

quicklogging.**exception**(**args*, ***kwargs*)
wrapper for logging.Logger.exception()

Variadic parameters: see logging.Logger.exception()

Parameters **stackoverhead** (*int*) – see *quicklogging.base.get_logger()*

print handlers

quicklogging.**catch_prints**(*catch_module=<object object>*, *catch_all=False*, *include_children=True*, *logfunc=<function info>*)
configure the print() catching: redirects calls to a logger

By default, only catches calls to print() from logger

- named after the calling module

- children of this logger

Note: API discussion welcome.

You anderstand the API is not stable.

Parameters

- **catch_module** (*string*) – include children of logger designed by name
- **catch_all** (*bool*) – should catch all print() diregarding where they're from ?

  Warning: take care of logging propagation (`Logger.propagate()`)

- **logfunc** (*function*) – function to use for logging messages

Possible extension ideas:

- wrap arbitrary output stream
- different log functions depending on regex applied on messages
- make it configurable from config file
- allow exclusion of specific modules

`quicklogging.`**warn_prints**(*catch_all=False*)

Activate warning when print is called

Parameters **catch_all** (*bool*) – defaults to False, ie. defaults to only warn about current module, ignoring imports

5.3.2 Module quicklogging.base

`quicklogging.base.`**get_logger**(*stackoverhead=0*)

wrapper for getLogger

Typical use, say you're in project/module/submodule.py and you want a logger.

```
l = get_logger()
print(l)
```

```
<logging.Logger at ... >
```

```
print(l.name)
```

```
project.module.submodule
```

Parameters **stackoverhead** (*int*) – defaults to 0. How deep to look in the stack for fetching the logger name.

Returns a logger named after the module at depth `stackoverhead`.

Return type logging.Logger

quicklogging.base.**_log_with_level**(*func_name*, **args*, ***kwargs*)

Internal convenience function to log with appropriate level

This function is called by the main log wrappers.

Fetches the appropriate logger, then the function named after the param `func_name`. This is slow, you'd better use :py:func`get_logger`.

Parameters

- **func_name** (*str*) – One of 'debug', 'info', 'error', etc
- **stackoverhead** (*int*) – defaults to 0. How deep to look in the stack for

Return type None

5.3.3 Module quicklogging.stream_wrapper

class quicklogging.stream_wrapper.**StreamWrapper**(*original_stream*, *logfunc*, *catch_all=False*, *warn=False*)

Implementation detail

5.4 Annex: Quick survival guide with the logging module

Sadly, some config still needs to take place for the `logging` module.

Disclaimer: we are not responsible for the boilerplate required by the logging API

You still need to configure logging somewhere at the start of your script -for complex apps (pyramid for example), initial scaffolding is likely to do this for you (see generated .ini file).

Example boilerplate for a script logging to `my_app.log`:

```
import logging
import logging.handlers
import quicklogging

# basicConfig is only effective once (if the root logger was not configured before)
logging.basicConfig(
    handlers=[logging.handlers.WatchedFileHandler('my_app.log')],
    format="%(asctime)s %(pathname)s:%(lineno)s%(message)s\\n",
    level=logging.WARNING
)
```

Python Module Index

q

quicklogging, 11
quicklogging.base, 12
quicklogging.stream_wrapper, 13

Index

Symbols

_log_with_level() (*in module quicklogging.base*), 12

C

catch_prints() (*in module quicklogging*), 11
critical() (*in module quicklogging*), 11

D

debug() (*in module quicklogging*), 11

E

error() (*in module quicklogging*), 11
exception() (*in module quicklogging*), 11

G

get_logger() (*in module quicklogging.base*), 12

I

info() (*in module quicklogging*), 11

Q

quicklogging (*module*), 11
quicklogging.base (*module*), 12
quicklogging.stream_wrapper (*module*), 13

S

StreamWrapper (*class in quicklogging.stream_wrapper*), 13

W

warn_prints() (*in module quicklogging*), 12
warning() (*in module quicklogging*), 11