Elegant way to make logging.LoggerAdapter available to other modules

Ask Question

I use a `LoggerAdapter` to let my python logging output Linux TIDs instead of the long unique IDs. But this way I don't modify an existing `logger` but I create a new object:

```
    new_logger = logging.LoggerAdapter(
                    logger=logging.getLogger('mylogger'),
                    extra=my_tid_extractor())
```

Now I want this `LoggerAdapter` be used by certain modules. As long as I know a global variable being used as logger I can do something like this:

```
    somemodule.logger = new_logger
```

But this is not nice - it works only in a couple of cases and you need to know the logger variables used by the modules.

Do you know a way to make a `LoggerAdapter` available globally e.g. by calling s.th. like

```
    logging.setLogger('mylogger', new_logger)
```

Or alternatively: is there some other way to let Python `logging` output Linux thread IDs like

asked Jan 20 '15 at 16:30

frans

2,434 ● 2 ● 21 ● 58

2 Answers

Alternatively, you can to implement custom logger, and make it default in logging module.

Here is example:

```
import logging
import ctypes

SYS_gettid = 186
libc = ctypes.cdll.Load

FORMAT = '%(asctime)-15
logging.basicConfig(lev

def my_tid_extractor():
    tid = libc.syscall(
    return {'tid': tid}

class CustomLogger(logg

    def _log(self, leve
        if extra is Non
            extra = my_
        super(CustomLog

logging.setLoggerClass(


logger = logging.getLog
logger.debug('test')
```

Output sample:

```
2015-01-20 19:24:09,782
```

dited Jan 20 '15 at 17:20

nswered Jan 20 '15 at 16:49

Dmitry Nedbaylo

1,758 ● 16 ● 19

Beautiful solution. :) Thanks! – erip May 16 '16 at 12:40

I think you need override LoggerAdapter.process() method Because the default LoggerAdapter.process method will do nothing, Here is example：

```
import logging
import random
L=logging.getLogger('na

class myLogger(logging.
    def process(self,ms
        return '(%d),%s

#put the randint functi
LA=myLogger(L,{'name1':

#now,do some logging
LA.debug('some_loging_m

out>>DEBUG:name:(167),s
```

nswered Dec 28 '15 at 11:46

user4554063

1