Python Log Messages With Dynamic Formatter Variables

Thu, Apr 2, 2020

In this post I will explain how it is possible to replace logger formatter variables with dynamic values. Take the following log message for example:

```
[2020-04-02 15:37:01] myapp INFO: Logged in successfully.
```

What if we wanted to add some additional data after the `INFO` part, *but* this data is dynamic, or is only available at runtime. Like a username or an app name for example.

Let's take a look at how to achieve this.

Log Filters

Let's assume that this data is passed to a class's initialization arguments. For example:

```
class Session:

    def __init__(self, username, *args, **kwargs):
        self.username = username
```

We want to use this `username` value in the log message's **formatter**. There are different ways to achieve this in Python, but in this post I will explain how to achieve this using **Log Filters**.

Log filters are used to add contextual information to log output. We can declare a custom log filter as follows:

```
import logging


class MyCustomLogFilter(logging.Filter):

    def __init__(self, username: str, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.username = username

    def filter(self, record):
        record.username = username
        return True
```

We can now add this filter along with the username value to our loggers. The log messages generated by the loggers will automatically place the username value in the specified location of the log formatter.

Here is an example of a formatter:

```
import logging


formatter = logging.Formatter(
    '[%(asctime)s] %(name)-5s %(levelname)s [%(username)s]: %(message)s',
```

```
    datefmt='%Y-%m-%d %H:%M:%S',
)
```

As you can see, the `[%(username)s]` part will now contain the actual username in the final log messages.

⚠ Log filters can be added to both **loggers** and **handlers**. However logging filters do not propagate like handlers and levels do. This means that child loggers will not be able to inherit logging filters from the parent if the filter is added to the parent. Therefore we will add our logging filter to the handler(s).

Setting Up The Loggers

We need to setup the logger inside our `Session` class. Because this is the only place where we have access to the `username` variable:

```
import logging

logger = logging.getLogger('myapp')


class Session:

    # ...

    def _setup_logger(self):
        logger.setLevel(logging.DEBUG)

        handler = logging.StreamHandler()
        handler.setFormatter(formatter)  # See the previous formatter
```

```
        # Filters attached to the parent logger DO NOT propagate to
        # child loggers, this is why we attach the filter to the
        # handler instead.
        handler.addFilter(MyCustomLogFilter(username=self.username))
        logger.addHandler(handler)
```

With the above setup we are ready to see the magic. In the same module we can try logging a message:

```
class Session:

    # ...

    def do_something(self):
        logger.info('HELLO WORLD')


if __name__ == '__main__':
    s = Session(username='madkilla')
    s.do_something()
```

This produces the following message (nevermind the timestamp):

```
[2020-04-02 15:37:01] myapp INFO [madkilla]: HELLO WORLD
```

Success! 🎉 🎉

Logging Accross Different Modules

To still be able to log similar messages accross different Python modules we only need to inherit from the parent logger. We do this like so:

```
# myothermodule.py
import logging


# Assuming that the parent log was called `myapp`
logger = logging.getLogger('myapp.' + __name__)
logger.info('This is a log from another module')
```

The above will allow us to know in which module exactly the log message was generated:

```
[2020-04-02 15:37:01] myapp.myothermodule INFO [madkilla]: This is a log from another module
```

Hope you found this useful!

References

1. https://docs.python.org/3.6/howto/logging-cookbook.html
2. https://www.saltycrane.com/blog/2014/02/python-logging-filters-do-not-propagate-like-handlers-and-levels-do/

python

Comments

ALSO ON **ANDRES ALVAREZ'S BLOG**

Django Dynamic Queryset Filtering …

3 years ago • 1 comment

Django Dynamic Queryset Filtering With Pagination This blog article does a …

Custom File Types in Netbeans Platform · …

3 years ago • 3 comments

Custom File Types in Netbeans Platform If your desktop Java application …

Custom Notification Action Types in …

4 years ago • 1 comment

Custom Notification Action Types in Zenoss In a previous post I talked …

Testin With

4 years

Testing RSpec improv

What do you think?

0 Responses

Upvote Funny Love Surprised Angry Sad

0 Comments **Andres Alvarez's Blog** **Disqus' Privacy Policy** **Login**

Recommend Tweet Share **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS

Name