Driftwood Documentation

Release

Hurricane Labs

September 22, 2015

Contents

CHAPTER 1

Driftwood

A collection of python logging extensions

CHAPTER 2

Features

- Compatible with Python 3 (only)
- Provides Dictionary, JSON, and MongoDB logging
- Features for logging custom attributes
- Can notify of status changes based on the level of messages being logged

CHAPTER 3

Get Driftwood

```
pip install driftwood
```

If you wish to use the MongoDB functionality, you must also have mongoengine installed. Version 0.8.7 is recommended.

CHAPTER 4

Run the tests

You must install tox, then run:

```
git clone https://github.com/HurricaneLabs/driftwood.git
cd driftwood
tox
```

CHAPTER 5

JSON Formatter

This code example:

```
import logging
from driftwood.formatters import JSONFormatter

log = logging.getLogger("test")
handler = logging.StreamHandler()

json_formatter = JSONFormatter()
handler.setFormatter(json_formatter)

log.addHandler(handler)
log.warning("uh oh")
```

Produces (as a string, not a dict):

```
{"created": 1422386241.4394472, "pathname": "<stdin>", "message": "uh oh", "threadName": "MainThread"
```

CHAPTER 6

MongoHandler

This handler is used to log records to MongoDB. The following code:

```
import logging
import os

from driftwood.handlers.mongo import MongoHandler, LogRecord
import mongoengine

mongoengine.connect("testdb", host=os.environ["MONGO_PORT_27017_TCP_ADDR"])
MongoClient('172.17.0.50', 27017)

mongo_handler = MongoHandler()
log = logging.getLogger("test")
log.addHandler(mongo_handler)

log.error("something bad happened")
print(LogRecord.objects)
print(LogRecord.objects[0].message)
```

Produces:

```
[<LogRecord: LogRecord object>]
something bad happened
```

Your message has been logged to mongodb and includes all standard logging attributes except asctime. See the full documentation for including extra attributes, as provided by the DictHandler base class.

CHAPTER 7

StatusUpdateAdapter

This logging.LoggerAdapter is used to track the status of an operation based on the level of messages being logged. Every time a message is logged, if the level is higher than any previous message, a callback is triggered to alert of the status change.

```
import logging

from driftwood.adapters import StatusUpdateAdapter

def status_update(levelno, levelname):
    print("The status has changed to {0}".format(levelname))

log = logging.getLogger("test")
log.setLevel(logging.CRITICAL)
adapter = StatusUpdateAdapter(status_update, log)

adapter.info("info test")
adapter.warning("warning test")
adapter.error("error test")

adapter.info("won't trigger the callback")
```

Produces:

```
The status has changed to INFO
The status has changed to WARNING
The status has changed to ERROR
```

CHAPTER 8

Table of Contents

8.1 driftwood.adapters package

8.1.1 Submodules

8.1.2 driftwood.adapters.status module

class driftwood.adapters.status.**StatusUpdateAdapter**(*status_update_func*, *logger*, *extra={}*)

Bases: logging.LoggerAdapter

Used to notify a callback about changes in the loglevel of a program.

Parameters **callable** (status_update_func) – Called when the status changes. See *status_update_func()* for the arguments this function should accept.

log(**args*, ***kwargs*)

status_update_func(*levelno*, *levelname*)

Example interface for the update function you pass to __init__().

Parameters

- **int** (*levelno*) – Logging level numeric value of this call to the LoggerAdapter. See Python Logging Levels
- **str** (*levelname*) – Logging level name of this call to the LoggerAdapter. See Python Logging Levels

8.1.3 Module contents

class driftwood.adapters.**StatusUpdateAdapter**(*status_update_func*, *logger*, *extra={}*)

Bases: logging.LoggerAdapter

Used to notify a callback about changes in the loglevel of a program.

Parameters **callable** (status_update_func) – Called when the status changes. See *status_update_func()* for the arguments this function should accept.

log(**args*, ***kwargs*)

status_update_func(*levelno*, *levelname*)

Example interface for the update function you pass to __init__().

Parameters

- **int** (*levelno*) – Logging level numeric value of this call to the LoggerAdapter. See Python Logging Levels
- **str** (*levelname*) – Logging level name of this call to the LoggerAdapter. See Python Logging Levels

8.2 driftwood.formatters package

8.2.1 Submodules

8.2.2 driftwood.formatters.dict module

class driftwood.formatters.dict.**DictFormatter**(**args*, *regular_attrs=None*, *extra_attrs=None*, *preserve_order=False*, *specific_order=None*, ***kwargs*)

Bases: logging.Formatter

Used for formatting log records into a dict.

Parameters

- **regular_attrs** (*list*) – A list of strings specifying built-in python logging args that should be included in each output dict. If not specified, all args will be used. Setting to an empty list will disable regular args.
- **extra_attrs** (*list*) – A list of strings specifying additional arguments that may exist on the log record instances and should be included in the messages. By default this will be calculated for you to include all available extra attributes.
- **preserve_order** (*bool*) – If `True`, will cause format() to return an OrderedDict with the keys in the same order every time. Default: `False`.
- **specific_order** (*list*) – If set to a list and preserve_order is `True`, will cause the keys to be in this order if they exist. Keys to be included that are not listed here will be included after the ones which are specified. Default is `None`, which will cause the default_regular_attrs to appear first in the order they are in the class attribute, and then extra attributes to be included in alphabetical order.

Note: The attribute useful_attrs is deprecated and maintained for backward compatibility. You should stop using it.

default_regular_attrs = ['name', 'message', 'levelno', 'levelname', 'pathname', 'filename', 'module', 'lineno', 'fun

format(*record*)
Formats a log record into a dictionary using the arguments given to __init__.

ignore_builtin_attrs = ['args', 'msg', 'exc_info', 'exc_text', 'stack_info']

8.2.3 driftwood.formatters.json module

class driftwood.formatters.json.**JSONFormatter**(**args*, *regular_attrs=None*, *extra_attrs=None*, *preserve_order=False*, *specific_order=None*, ***kwargs*)

Bases: *driftwood.formatters.dict.DictFormatter*

Formats messages as JSON.

Accepts the same arguments as *DictFormatter*

Parameters

- **regular_attrs** (*list*) – A list of strings specifying built-in python logging args that should be included in each output dict. If not specified, all args will be used. Setting to an empty list will disable regular args.
- **extra_attrs** (*list*) – A list of strings specifying additional arguments that may exist on the log record instances and should be included in the messages. By default this will be calculated for you to include all available extra attributes.
- **preserve_order** (*bool*) – If `True`, will cause format() to return an OrderedDict with the keys in the same order every time. Default: `False`.
- **specific_order** (*list*) – If set to a list and preserve_order is `True`, will cause the keys to be in this order if they exist. Keys to be included that are not listed here will be included after the ones which are specified. Default is `None`, which will cause the default_regular_attrs to appear first in the order they are in the class attribute, and then extra attributes to be included in alphabetical order.

Note: The attribute useful_attrs is deprecated and maintained for backward compatibility. You should stop using it.

format(*record*)

8.2.4 driftwood.formatters.keyval module

This is a formatter that outputs records in a key=value format.

class driftwood.formatters.keyval.**KeyValFormatter**(**args*, *regular_attrs=None*, *extra_attrs=None*, *preserve_order=False*, *specific_order=None*, ***kwargs*)

Bases: *driftwood.formatters.dict.DictFormatter*

Outputs records in the following format: `message='foo',created='1429896792.703648'`

Note: This formatter will replace spaces with underscores in log record keys, and remove single quotes in record values.

Parameters

- **regular_attrs** (*list*) – A list of strings specifying built-in python logging args that should be included in each output dict. If not specified, all args will be used. Setting to an empty list will disable regular args.
- **extra_attrs** (*list*) – A list of strings specifying additional arguments that may exist on the log record instances and should be included in the messages. By default this will be calculated for you to include all available extra attributes.
- **preserve_order** (*bool*) – If `True`, will cause format() to return an OrderedDict with the keys in the same order every time. Default: `False`.
- **specific_order** (*list*) – If set to a list and preserve_order is `True`, will cause the keys to be in this order if they exist. Keys to be included that are not listed here will be included after the ones which are specified. Default is `None`, which will cause the default_regular_attrs to

appear first in the order they are in the class attribute, and then extra attributes to be included in alphabetical order.

Note: The attribute useful_attrs is deprecated and maintained for backward compatibility. You should stop using it.

format(**args*, ***kwargs*)

8.2.5 driftwood.formatters.splunk module

class driftwood.formatters.splunk.**SplunkFormatter**(**args*, ***kwargs*)
Bases: *driftwood.formatters.json.JSONFormatter*

Formats messages as JSON with order preserved for splunk

Accepts the same arguments as *JSONFormatter*

8.2.6 Module contents

class driftwood.formatters.**DictFormatter**(**args*, *regular_attrs=None*, *extra_attrs=None*, *preserve_order=False*, *specific_order=None*, ***kwargs*)
Bases: logging.Formatter

Used for formatting log records into a dict.

Parameters

- **regular_attrs** (*list*) – A list of strings specifying built-in python logging args that should be included in each output dict. If not specified, all args will be used. Setting to an empty list will disable regular args.
- **extra_attrs** (*list*) – A list of strings specifying additional arguments that may exist on the log record instances and should be included in the messages. By default this will be calculated for you to include all available extra attributes.
- **preserve_order** (*bool*) – If True, will cause format() to return an OrderedDict with the keys in the same order every time. Default: False.
- **specific_order** (*list*) – If set to a list and preserve_order is True, will cause the keys to be in this order if they exist. Keys to be included that are not listed here will be included after the ones which are specified. Default is None, which will cause the default_regular_attrs to appear first in the order they are in the class attribute, and then extra attributes to be included in alphabetical order.

Note: The attribute useful_attrs is deprecated and maintained for backward compatibility. You should stop using it.

default_regular_attrs = ['name', 'message', 'levelno', 'levelname', 'pathname', 'filename', 'module', 'lineno', 'fun

format(*record*)
Formats a log record into a dictionary using the arguments given to __init__.

ignore_builtin_attrs = ['args', 'msg', 'exc_info', 'exc_text', 'stack_info']

class driftwood.formatters.**JSONFormatter**(**args*, *regular_attrs=None*, *extra_attrs=None*, *preserve_order=False*, *specific_order=None*, ***kwargs*)

Bases: *driftwood.formatters.dict.DictFormatter*

Formats messages as JSON.

Accepts the same arguments as *DictFormatter*

Parameters

- **regular_attrs** (*list*) – A list of strings specifying built-in python logging args that should be included in each output dict. If not specified, all args will be used. Setting to an empty list will disable regular args.
- **extra_attrs** (*list*) – A list of strings specifying additional arguments that may exist on the log record instances and should be included in the messages. By default this will be calculated for you to include all available extra attributes.
- **preserve_order** (*bool*) – If `True`, will cause format() to return an OrderedDict with the keys in the same order every time. Default: `False`.
- **specific_order** (*list*) – If set to a list and preserve_order is `True`, will cause the keys to be in this order if they exist. Keys to be included that are not listed here will be included after the ones which are specified. Default is `None`, which will cause the default_regular_attrs to appear first in the order they are in the class attribute, and then extra attributes to be included in alphabetical order.

Note: The attribute useful_attrs is deprecated and maintained for backward compatibility. You should stop using it.

format(*record*)

class driftwood.formatters.**KeyValFormatter**(**args*, *regular_attrs=None*, *extra_attrs=None*, *preserve_order=False*, *specific_order=None*, ***kwargs*)

Bases: *driftwood.formatters.dict.DictFormatter*

Outputs records in the following format: `message='foo',created='1429896792.703648'`

Note: This formatter will replace spaces with underscores in log record keys, and remove single quotes in record values.

Parameters

- **regular_attrs** (*list*) – A list of strings specifying built-in python logging args that should be included in each output dict. If not specified, all args will be used. Setting to an empty list will disable regular args.
- **extra_attrs** (*list*) – A list of strings specifying additional arguments that may exist on the log record instances and should be included in the messages. By default this will be calculated for you to include all available extra attributes.
- **preserve_order** (*bool*) – If `True`, will cause format() to return an OrderedDict with the keys in the same order every time. Default: `False`.
- **specific_order** (*list*) – If set to a list and preserve_order is `True`, will cause the keys to be in this order if they exist. Keys to be included that are not listed here will be included after the ones which are specified. Default is `None`, which will cause the default_regular_attrs to appear first in the order they are in the class attribute, and then extra attributes to be included in alphabetical order.

Note: The attribute useful_attrs is deprecated and maintained for backward compatibility. You should stop using it.

format(**args*, ***kwargs*)

class driftwood.formatters.**SplunkFormatter**(**args*, ***kwargs*)
Bases: *driftwood.formatters.json.JSONFormatter*

Formats messages as JSON with order preserved for splunk

Accepts the same arguments as *JSONFormatter*

8.3 driftwood.handlers package

8.3.1 Submodules

8.3.2 driftwood.handlers.dict module

class driftwood.handlers.dict.**DictHandler**(**args*, *regular_attrs=None*, *extra_attrs=[]*, ***kwargs*)
Bases: logging.Handler

Formats log records into a dict.

Meant to be subclassed. This is just a convenience wrapper around *driftwood.formatters.dict.DictFormatter*.

Parameters **extra_attrs** (*list*) – String names of extra attributes that may exist on the log record.

emit(*record*)
Super this in your subclass to format the record into a dict

8.3.3 driftwood.handlers.mongo module

8.3.4 Module contents

class driftwood.handlers.**DictHandler**(**args*, *regular_attrs=None*, *extra_attrs=[]*, ***kwargs*)
Bases: logging.Handler

Formats log records into a dict.

Meant to be subclassed. This is just a convenience wrapper around *driftwood.formatters.dict.DictFormatter*.

Parameters **extra_attrs** (*list*) – String names of extra attributes that may exist on the log record.

emit(*record*)
Super this in your subclass to format the record into a dict

d

D

E

F

I

J

K

L

S