



logging-endpoint Documentation

Propylon Release 1.0.0

Propylon

Jan 10, 2020

Contents

1	Introduction	3
1	Installation	3
2	Endpoints	3
2	Settings	5
3	Changelog	7
1	1.0.1	7
2	1.0.0	7
4	Indices and tables	9
	Index	11

Contents:

Chapter 1

Introduction

1 Installation

1. Install the package:

```
pip install django-logging-endpoint
```

2. Install the application by adding it to the INSTALLED_APPS setting:

```
INSTALLED_APPS += ('logging_endpoint',)
```

3. Set the logger name, if you want to send the messages to a specific one:

```
LOGGING_ENDPOINT_LOGGER = 'LoggingEndpoint'
```

4. Set the log message handler function, if you want to customize the parsing of your log messages:

```
LOGGING_ENDPOINT_MESSAGE_HANDLER = 'logging_endpoint.message_handler.default_  
↩handler'
```

5. Add the url to your urls.py:

```
from django.conf.urls import include  
  
urlpatterns += url(r'^logs', include('logging_endpoint.urls'))
```

2 Endpoints

2.1 root

The root endpoint of `django-logging-endpoint` receives a json message with the logs to be sent to the configured logger:

```
{  
    'message': 'my log message',  
    'logger': 'user interaction',  
    'loglevel': 'error',  
    'timestamp': '2020-01-01T12:00Z'  
}
```

By default, a list of logs can be received and will be expanded to the Django logger. See the documentation's settings chapter for more information on that.

Chapter 2

Settings

logging_endpoint is configured by adding the following settings to the Django settings.

LOGGER

The name of the logger to send the received logs to.

Default 'LoggingEndpoint'

Examples:

```
LOGGING_ENDPOINT_LOGGER = 'MyLogger'
```

MESSAGE_HANDLER

Function to process the incoming message by the application. Takes the original message and request under these names as keywords and should return a tuple of

- logger name (or None)
- loglevel
- log message
- args for the log call
- kwargs for the log call

```
def default_handler(**kwargs):
    """Return the message as is as level INFO on the default logger."""
    log_data = kwargs.get('message')
    return None, INFO, log_data.decode(), tuple(), dict()
```

Default logging_endpoint.message_handler.default_handler

Examples:

```
LOGGING_ENDPOINT_MESSAGE_HANDLER = log_message_handler
LOGGING_ENDPOINT_MESSAGE_HANDLER = 'path.to.handler'
```

OVERWRITE_LOGGER

If set to true an incoming json message will be sent to the logger specified under the logger key. Otherwise the message is sent to the standard logger (see setting LOGGER) and the logger value is added to the message.

Default False

Examples:

```
LOGGING_ENDPOINT_OVERWRITE_LOGGER = False
```

DECOMPOSE_JSON_LIST

If set to true an incoming json list will be decomposed into separate messages:

True:

[“log1”, “log2”] => INFO log1 INFO log2

False: [“log1”, “log2”] => INFO [“log1”, “log2”]

Default True

Examples:

LOGGING_ENDPOINT_DECOMPOSE_JSON_LIST = True
--

Chapter 3

Changelog

1 1.0.1

- Remove Herodotus dependency
- Omit latex glossary
- Stringify log message

2 1.0.0

- Initial version

Chapter 4

Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

Index

D

DECOMPOSE_JSON_LIST, [5](#)

L

LOGGER, [5](#)

M

MESSAGE_HANDLER, [5](#)

O

OVERWRITE_LOGGER, [5](#)