LoggerServer

Release 1.0.5

Jan 02, 2018

Contents

logger_server is an async TCP logging server base on tornado.ioloop, which serve for logging.handlers.SocketHandler.

More information, see logging-cookbook.

CHAPTER 1

Usage

1.1 1. Install

Install `LoggerServer` by `pip`

```
pip install logger_server
```

1.2 2. Configuration

After `logger_server` package installed, you will got a command `logger-server`

`logger-server` options:

```
$ logger_server -h
usage: logger-server [-h] [-f CONF] [-p PORT] [--log LOG] [--when WHEN]
                     [--interval INTERVAL] [--backup BACKUP] [--fmt FMT]
                     [--datefmt DATEFMT]

LoggerServer help documentation

optional arguments:
  -h, --help            show this help message and exit
  -f CONF, --conf CONF  The config file path for LoggerServer. (default: None)
  -p PORT, --port PORT  LoggerServer port. (default: 9876)
  --log LOG             The log output file. (default: ./logserver.log)
  --when WHEN           specify the type of TimedRotatingFileHandler
                        interval.other options:('S', 'M', 'H', 'D', 'W0'-'W6')
                        (default: midnight)
  --interval INTERVAL   The interval value of timed rotating. (default: 1)
  --backup BACKUP       Number of log files to keep. (default: 14)
  --fmt FMT             The log output formatter of logging. (default:
                        [%(levelname)1.1s %(asctime)s %(ip)s %(name)s
```

```
                        %(module)s:%(lineno)d] %(message)s)
  --datefmt DATEFMT     The log output date formatter of logging. (default:
                        %Y-%m-%d %H:%M:%S.%f)
  --detached            Running on detached mode. (default: False)
  --separated           Output log into every single file. logger name will in
                        the suffix of every logger file. (default: False)
```

Use param `-f` to specify a config file for `logger-server`. logserver.conf.template is a configuration template.

New in version 1.0.3: Use param `--detached` to set `logger-server` runnning as daemon.

Warning: By default, the config file setttings over command settings.

Start LoggerServer

```
$ logger_server
```

Output:

```
> LoggerServer is binding on 0.0.0.0:9876
```

Note: `logger-server` bind on port `9876` by default. if this port is not avalible, use param `-p` specify a new port.

1.3 3. Hello word

After start `logger-server`, now you can write your logging code, like below:

Example1 use root logger:

```
import logging
import logging.handlers

# root logger setting
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)
# change localhost to your `logger-server` ip
socketHandler = logging.handlers.SocketHandler('localhost', 9876)
logger.addHandler(socketHandler)

# use root logger
logging.debug('The root logger is working!')
logging.info('The root logger is working!')
logging.warn('The root logger is working!')
logging.error('The root logger is working!')
logging.critical('The root logger is working!')

# unsetted logger inherit the settings of root logger by default.
logger = logging.getLogger('default_logger')
logger.debug('The default logger is working!')
logger.info('The default logger is working!')
logger.warn('The default logger is working!')
logger.error('The default logger is working!')
logger.critical('The default logger is working!')
```

Output:

```
[D 2017-11-16 11:13:19.700965 127.0.0.1 root root_logger:23] The root logger is␣
↪working!
[I 2017-11-16 11:13:19.702461 127.0.0.1 root root_logger:24] The root logger is␣
↪working!
[W 2017-11-16 11:13:19.702587 127.0.0.1 root root_logger:25] The root logger is␣
↪working!
[E 2017-11-16 11:13:19.702661 127.0.0.1 root root_logger:26] The root logger is␣
↪working!
[C 2017-11-16 11:13:19.702759 127.0.0.1 root root_logger:27] The root logger is␣
↪working!
[D 2017-11-16 11:13:19.702844 127.0.0.1 default_logger root_logger:31] The default␣
↪logger is working!
[I 2017-11-16 11:13:19.702919 127.0.0.1 default_logger root_logger:32] The default␣
↪logger is working!
[W 2017-11-16 11:13:19.702979 127.0.0.1 default_logger root_logger:33] The default␣
↪logger is working!
[E 2017-11-16 11:13:19.703068 127.0.0.1 default_logger root_logger:34] The default␣
↪logger is working!
[C 2017-11-16 11:13:19.703124 127.0.0.1 default_logger root_logger:35] The default␣
↪logger is working!
```

Get this example code root_logger.py.

Example2 use custom logger:

```
import logging
import logging.handlers

# logger setting
logger = logging.getLogger('test')
logger.setLevel(logging.DEBUG)
# change localhost to your `logger-server` ip
socketHandler = logging.handlers.SocketHandler('localhost', 9876)
logger.addHandler(socketHandler)

logger.debug('The test logger is working!')
logger.info('The test logger is working!')
logger.warn('The test logger is working!')
logger.error('The test logger is working!')
logger.critical('The test logger is working!')
```

Output:

```
[D 2017-11-16 11:19:48.623884 127.0.0.1 test custom_logger:22] The test logger is␣
↪working!
[I 2017-11-16 11:19:48.625533 127.0.0.1 test custom_logger:23] The test logger is␣
↪working!
[W 2017-11-16 11:19:48.625658 127.0.0.1 test custom_logger:24] The test logger is␣
↪working!
[E 2017-11-16 11:19:48.625739 127.0.0.1 test custom_logger:25] The test logger is␣
↪working!
[C 2017-11-16 11:19:48.625821 127.0.0.1 test custom_logger:26] The test logger is␣
↪working!
```

Get this example code custom_logger.py.

CHAPTER 2

Module

2.1 logger_server

@Project: LoggerServer @Filename: server.py @Author: Kehr <kehr.china@gmail.com> @Created Date: 2017-11-14T19:20:37+08:00 @Last modified time: 2017-11-20T16:19:06+08:00 @License: Apache License <http://www.apache.org/licenses/LICENSE-2.0>

class logger_server.server.**AttrDict**

Attribute dict

Make `dict` value can be accessed by attribute:

```
from logger_server import AttrDict

people = AttrDict({'name':'Joe', 'age': 23})
print people.name
print people.age
```

class logger_server.server.**LoggerFormatter**(*fmt='%(color)s[%(levelname)1.1s %(asctime)s %(module)s:%(lineno)d]%(end_color)s %(message)s'*, *datefmt='%y%m%d %H:%M:%S'*, *style='%'*, *color=True*, *colors={40: 1, 10: 4, 20: 2, 30: 3}*)

format

Parameters

- **color** (*bool*) – Enables color support.
- **fmt** (*string*) – Log message format. It will be applied to the attributes dict of log records. The text between `%(color)s` and `%(end_color)s` will be colored depending on the level if color support is on.
- **colors** (*dict*) – color mappings from logging level to terminal color code
- **datefmt** (*string*) – Datetime format. Used for formatting `(asctime)` placeholder in `prefix_fmt`.

Changed in version 3.2: Added `fmt` and `datefmt` arguments.

formatTime(*record*, *datefmt=None*)
Rewrite default *formatTime* to support `%f`

Parameters

- **record** (*logging.LogRecord*) – `logging.LogRecord` object which Contains all the information pertinent to the event being logged.
- **datefmt** – Datetime format string.

class logger_server.server.**LoggerServer**(*config=None*, **args*, ***kwargs*)
A logging server serve for `logging.handlers.SocketHandler`

Parameters **config** (*argparse.Namespace*) – The command line parse result.

Reference: tornado.tcpserver.TCPServer

handle_stream(**args*, ***kwargs*)
Reslove Socket stream data.

Parameters

- **stream** – scoket stream
- **address** (*tuple*) – client resquest ip and port `(ip, port)`

handleLogRecord(*record*, *address*)
Config the decoded `record`

This will add `ip` and `port` param by `logging.Filter`. You can add them into logging format:

```
[%(levelname)1.1s %(asctime)s %(ip)s %(port)s %(name)s %(module)s:%(lineno)d]
↪%(message)s
```

Parameters **record** (*logging.LogRecord*) – `logging.LogRecord` object which

Contains all the information pertinent to the event being logged. :arg tuple address: client resquest ip and port `(ip, port)`

start()
Start LoggerServer

init_config_options(*config=None*)
Initialize `logger-server` config.

All settings will be merged in `self.options`

Parameters **config** (*argparse.Namespace*) – The command line parse result.

add_logger_file_handler(*logger=None*, *name=''*)
Add `TimedRotatingFileHandler` for every single logger

logger_server.server.**parse_command**()
Parses the command args

logger_server.server.**main**()
LoggerServer Entry

CHAPTER 3

Release history

3.1 Version 1.0.5, Nov 20 2017

- Add `--separated` param. Output log to every single file.

3.2 Version 1.0.4, Nov 20 2017

- Add `--version` param.
- Change default port from `9000` to `9876`

3.3 Version 1.0.3, Nov 16 2017

- Add `--detached` param.

3.4 Version 1.0.0, Nov 15 2017

- Release v1.0.0

Python Module Index

l

Index