logging-mv-integrations Documentation

Release 0.5.4

TUNE Inc.

Feb 18, 2018

Contents:

CHAPTER 1

Introduction

Extension to Python logging functionality intended for TUNE Multiverse Integrations.

CHAPTER 2

Badges

docs	
info	
tests	
package	
other	

CHAPTER 3

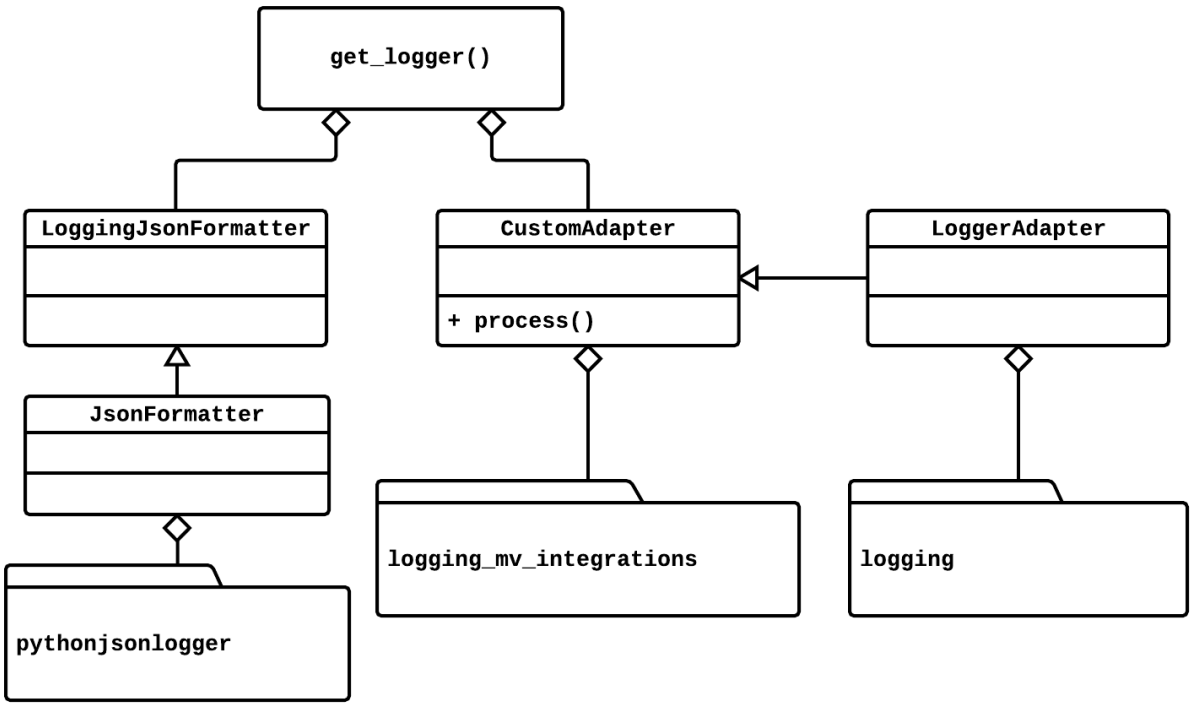
Install

```
pip install logging_mv_integrations
```

CHAPTER 4

Architecture

`logging-mv-integrations` is an extension of the logging facility for Python used for TUNE Multiverse Integrations providing custom logger levels, format, and output.



CHAPTER 5

Function: get_logger()

```
def get_logger(
    logger_name,
    logger_version=None,
    logger_level=logging.INFO,
    logger_format=LoggingFormat.JSON,
    logger_output=LoggingOutput.STDOUT_COLOR,
    logger_handler=None
):
```

5.1 get_logger(): Parameters

Parameter	Purpose
logger_name	Return a logger with the specified name or, if name is None, return a logger which is the root logger of the hierarchy.
logger_version	
logger_format	LoggingFormat
logger_output	LoggingOutput
logger_handler	logging.StreamHandler() or logging.FileHandler()

5.2 Logging Levels

Same Python logging levels, including one additional level NOTE.

Level	Purpose
DEBUG	Detailed information, typically of interest only when diagnosing problems.
NOTE	Detailed information, request processing, for example, request using cURL.
INFO	Confirmation that things are working as expected. *[DEFAULT]*
WARNING	An indication that something unexpected happened, or indicative of some problem in the near future. The software is still working as expected.
ERROR	Due to a more serious problem, the software has not been able to perform some function.
CRITICAL	A serious error, indicating that the program itself may be unable to continue running.

5.3 Logging Format

Format	Purpose
STANDARD	Standard logging format.
JSON	JSON logging format. *[DEFAULT]*

```
class LoggingFormat(object):
    """TUNE Logging Format ENUM
    """
    STANDARD = "standard"
    JSON = "json"
```

5.4 Logging Output

Output	Purpose
STDOUT	Standard Output to terminal
STDOUT_COLOR	Standard Output using colored terminal
FILE	Standard Output to file created within *./tmp/log_<epoch time seconds>.json*.

```
class LoggingOutput(object):
    """TUNE Logging Output ENUM
    """
    STDOUT = "stdout"
    STDOUT_COLOR = "color"
    FILE = "file"
```

5.5 Logging JSON Format

```
import logging
from logging_mv_integrations import (LoggingFormat, get_logger, __version__)

log = get_logger(
    logger_name=__name__,
    logger_version=__version__,
```

```
    logger_format=LoggingFormat.JSON,
    logger_level=logging.NOTE
)

log.info("logging: info", extra={'test': __name__})
log.note("logging: note", extra={'test': __name__})
log.debug("logging: debug", extra={'test': __name__})
log.warning("logging: warning", extra={'test': __name__})
log.error("logging: error", extra={'test': __name__})
log.critical("logging: critical", extra={'test': __name__})
log.exception("logging: exception", extra={'test': __name__})
```

5.6 Logging JSON Example Output

```
$ python3 examples/example_logging_json.py

{"asctime": "2017-10-20 08:31:14 -0700", "levelname": "INFO", "name": "__main__",
"version": "0.1.6", "message": "logging: info", "test": "__main__"}
{"asctime": "2017-10-20 08:31:14 -0700", "levelname": "NOTE", "name": "__main__",
"version": "0.1.6", "message": "logging: note", "test": "__main__"}
{"asctime": "2017-10-20 08:31:14 -0700", "levelname": "WARNING", "name": "__main__",
"version": "0.1.6", "message": "logging: warning", "test": "__main__"}
{"asctime": "2017-10-20 08:31:14 -0700", "levelname": "ERROR", "name": "__main__",
"version": "0.1.6", "message": "logging: error", "test": "__main__"}
{"asctime": "2017-10-20 08:31:14 -0700", "levelname": "CRITICAL", "name": "__main__",
"version": "0.1.6", "message": "logging: critical", "test": "__main__"}
{"asctime": "2017-10-20 08:31:14 -0700", "levelname": "ERROR", "name": "__main__",
"version": "0.1.6", "message": "logging: exception", "exc_info": "NoneType: None",
"test": "__main__"}
```

CHAPTER 6

Requirements

`logging-mv-integrations` module is built upon Python 3 and has dependencies upon several Python modules available within Python Package Index PyPI.

```
make install-requirements
```

or

```
python3 -m pip uninstall --yes --no-input -r requirements.txt
python3 -m pip install --upgrade -r requirements.txt
```

6.1 Dependencies

- coloredlogs: https://pypi.python.org/pypi/coloredlogs
- pprintpp: https://pypi.python.org/pypi/pprintpp
- python-json-logger: https://pypi.python.org/pypi/python-json-logger
- Pygments: https://pypi.python.org/pypi/Pygments
- safe-cast: https://pypi.python.org/pypi/safe-cast
- wheel: https://pypi.python.org/pypi/wheel