python logging模块介绍

logging介绍

Python的logging模块提供了通用的日志系统，可以方便第三方模块或者是应用使用。这个模块提供不同的日志级别，并可以采用不同的方式记录日志，比如文件，HTTP GET/POST，SMTP，Socket等，甚至可以自己实现具体的日志记录方式。

logging模块与log4j的机制是一样的，只是具体的实现细节不同。模块提供logger，handler，filter，formatter。

- logger
  提供日志接口，供应用代码使用。logger最长用的操作有两类：配置和发送日志消息。可以通过logging.getLogger(name)获取logger对象，如果不指定name则返回root对象，多次使用相同的name调用getLogger方法返回同一个logger对象。

- handler
  将日志记录（log record）发送到合适的目的地（destination），比如文件，socket等。一个logger对象可以通过addHandler方法添加0到多个handler，每个handler又可以定义不同日志级别，以实现日志分级过滤显示。

- filter
  提供一种优雅的方式决定一个日志记录是否发送到handler。

- formatter
  指定日志记录输出的具体格式。formatter的构造方法需要两个参数：消息的格式字符串和日期字符串，这两个参数都是可选的。

与log4j类似，logger，handler和日志消息的调用可以有具体的日志级别（Level），只有在日志消息的级别大于logger和handler的级别。

封装一个自己的日志类

下面我们把其复杂的功能进行简化封装，只使用最基本的的功能，来自定义成我们自己的日志管理类，并演示如何使用。

```
#-*- coding:utf-8 -*-


__author__ = "苦叶子"

import logging

LOGGING_FORMAT = '%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %
(message)s'

class LYMLogging:
    def __init__(self,
```

```
            level=logging.DEBUG, # 日志级别
            format=LOGGING_FORMAT, # 日志格式
            datefmt='%a, %d %b %Y %H:%M:%S', # 日期格式
            filename='LYM.log', # 日志文件名
            filemode='w' # 文件打开模式
            ):
            self.level = level
            self.format = format
            self.datefmt = datefmt
            self.filename = filename
            self.filemode = filemode

            # 初始化日志同时输出到console和日志文件
            logging.basicConfig(level=self.level,
                format=self.format,
                datefmt=self.datefmt,
                filename=self.filename,
                filemode=self.filemode)

            #定义一个StreamHandler，将INFO级别或更高的日志信息打印到标准错误，并将其添加到
当前的日志处理对象
            console = logging.StreamHandler()
            console.setLevel(logging.INFO)
            formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %
(message)s')
            console.setFormatter(formatter)
            logging.getLogger('LYMLogger').addHandler(console)
            self.log = logging.getLogger("LYMLogger")

        # 日志输出
        def output(self, msg, level=logging.DEBUG):
            if level == logging.DEBUG:
                # 调试信息
                self.log.debug(msg)
            elif level == logging.INFO:
                # 一般的信息
                self.log.info(msg)
            elif level == logging.WARNING:
                # 警告信息
                self.log.warning(msg)
            elif level == logging.ERROR:
                # 错误信息
                self.log.error(msg)
            else:
                # 尼玛
                self.log.critical(msg)



        def set_level(self, level=logging.DEBUG):
            self.log.set_level(level)
```

```
if __name__ == "__main__":
    print("python logging实例")

    log = LYMLogging()

    log.output("it's debug msg", level=logging.DEBUG)
    log.output("it's info msg", level=logging.INFO)
    log.output("it's warning msg", level=logging.WARNING)
    log.output("it's error msg", level=logging.ERROR)
    log.output("it's fuck msg", level=logging.CRITICAL)
```

结果输出

在控制台输出一下内容

> python logging实例
> LYMLogger : INFO it's info msg
> LYMLogger : WARNING it's warning msg
> LYMLogger : ERROR it's error msg
> LYMLogger : CRITICAL it's fuck msg

在LYM.log日志文件中写入了以下内容：

> Wed, 23 Aug 2017 10:04:05 logging_demo.py[line:43] DEBUG it's debug msg
> Wed, 23 Aug 2017 10:04:05 logging_demo.py[line:46] INFO it's info msg
> Wed, 23 Aug 2017 10:04:05 logging_demo.py[line:49] WARNING it's warning msg
> Wed, 23 Aug 2017 10:04:05 logging_demo.py[line:52] ERROR it's error msg
> Wed, 23 Aug 2017 10:04:05 logging_demo.py[line:55] CRITICAL it's fuck msg

请思考为什么控制台和日志文件的内容有区别？

小结

这里只是对日志模块logging进行了简单的分享，对于其更强大的功能请自行去学习和实践。例如：

- 用配置文件来控制日志输出
- 实现日志回滚

扫一扫关注微信公众号：