layout: post
title: "python日志处理模块_logging"
date: 2020-07-08
description: "python常用技巧"

tag: python常用技巧

日志记录的重要性

日志是对软件执行时所发生事件的一种追踪方式。软件开发人员对他们的代码添加日志调用，借此来指示某事件的发生。一个事件通过一些包含变量数据的描述信息来描述（比如：每个事件发生时的数据都是不同的）。开发者还会区分事件的重要性，重要性也被称为 等级 或 严重性。利用日志可以将事件分为不同的等级。

日志等级简列如下（以严重性递增）：

级别	何时使用
DEBUG	细节信息，仅当诊断问题时适用
INFO	确认程序按预期运行
WARNING	表明有已经或即将发生的意外（例如：磁盘空间不足）。程序仍按预期进行
ERROR	由于严重的问题，程序的某些功能已经不能正常执行
CRITICAL	严重的错误，表明程序已不能继续执行

默认的级别是`WARNING`，意味着只会追踪该级别及以上的事件，除非更改日志配置。

所追踪事件可以以不同形式处理。最简单的方式是输出到控制台。另一种常用的方式是写入磁盘文件。

logging模块的基本使用方法

```
import os
import logging
if __name__  == '__main__':
    # 如果不指定filename，将会把信息输出到console控制台。（类似于print的操作）
    logging.basicConfig(filename='example.log',format='%(asctime)s  %(process)s
%(message)s',level=logging.DEBUG)
    logging.debug('This is a debug message.')
    logging.info('this is a info message.')
    logging.warning('this is a warning message.')
    logging.warning('this is a warning message, too!')
    logging.info('this is a info message. too!')
    logging.error('Error')
```

程序最终会产生example.log的文件，文件中显示的信息如下所示：

```
2020-07-07 15:59:35,470  12024  This is a debug message.
2020-07-07 15:59:35,470  12024  this is a info message.
2020-07-07 15:59:35,470  12024  this is a warning message.
2020-07-07 15:59:35,471  12024  this is a warning message, too!
2020-07-07 15:59:35,471  12024  this is a info message. too!
2020-07-07 15:59:35,471  12024  Error
```

进阶使用方式

日志库采用模块化方法，并提供几类组件：记录器、处理程序、过滤器和格式化程序。

- 记录器暴露了应用程序代码直接使用的接口。
- 处理器将日志记录（由记录器创建）发送到适当的目标。
- 过滤器提供了更精细的附加功能，用于确定要输出的日志记录。
- 格式器指定最终输出中日志记录的样式。

日志事件信息在 LogRecord 实例中的记录器、处理程序、过滤器和格式化程序之间传递。

```
# 同时输出到文件和屏幕. 可以在多个模块中logging
import logging

# set up logging to file - see previous section for more details
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(name)-12s %(levelname)-8s %
(message)s',
                    datefmt='%m-%d %H:%M',
                    filename='myapp.log',
                    filemode='w')  # filemode默认为a，追加信息到日志文件，指定为‘w'，重
新写入 文件，之前的文件信息丢失
# 定义一个handler来将信息输出到控制台，StreamHandler与FileHandler
console = logging.StreamHandler()
console.setLevel(logging.INFO)
# 设置在控制台输出格式
formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
console.setFormatter(formatter)
#  将handler加入到根记录器
logging.getLogger('').addHandler(console)

# 根记录器输出信息
logging.info('this is an info message.')

#定义其他的记录器输出信息

logger1 = logging.getLogger('myapp.area1')
logger2 = logging.getLogger('myapp.area2')

logger1.debug('logger1 debug.')
logger1.info('logger1.info.')
logger2.warning('logger2 warning.')
logger2.error('logger2 error.')
```