**http://john16blog.blogspot.ru/2011/07/python-logging.html**

[**Python: Модуль logging. Примеры конфигураций.**](http://john16blog.blogspot.ru/2011/07/python-logging.html)

Данный модуль, поставляемый в базовой комплектации Питона, предоставляет средства для ведения лог-журналов. Он поддерживает иерархичную цепочку уровней важности сообщений, различные обработчики сообщений, такие как различные лог файлы, HTTP, TCP, UDP, SMTP, Memory. За более подробным описанием необходимо обратиться в справочные средства Питона.

Базовая простая настройка

Самая простая и быстрая конфигурация логирования состоит в базовой настройке модуля, как показано ниже. Указывается куда будут выводиться сообщения (в файл или поток), формат сообщения и даты, уровень важности сообщений.

# -\*- encoding: utf-8 -\*-   
  
import logging, os, sys   
  
logging.basicConfig(format='%(asctime)s.%(msecs)d %(levelname)s in \'%(module)s\' at line %(lineno)d: %(message)s',   
                    datefmt='%Y-%m-%d %H:%M:%S',   
                    level=logging.DEBUG,   
                    filename='my\_app.log')   
  
# Некоторый код   
logging.debug('this is test message')   
logging.log(1, 'this is message you will never see')   
logging.critical(u'критическая ошибка! Вследствии чего Мир , возможно, был удален')

в результате в текущей директории будет создан файл my\_app.log в котором будут содержаться следующие строчки:

2011-07-20 12:45:19.513 DEBUG in 'logging\_test1' at line 11: this is test message   
2011-07-20 12:45:19.576 CRITICAL in 'logging\_test1' at line 13: критическая ошибка! Вследствии чего Мир , возможно, был удален

Типичная настройка для простых приложений

Суть ее заключается в разделении обработки сообщений различной важности. Например сообщения с уровнем DEBUG представляют интерес только в случае отладки работы программы, в повсеместной ее работе они, скорее всего, лишние и будут загромождать лог файл. Поэтому необходимо сделать для них вывод на консоль. Сообщения с уровнями INFO и выше должны отображаться как в консоли так и в лог файле. Сообщения CRITICAL являются серьезными и важными сообщениями в сбое работы программы, поэтому желателно сразу уведомить разработчика о проблеме, для этого необходимо, например, отослать ему письмо на email или сообщение на веб-сервер.   
В общем случае подобное поведение можно задать двумя способами: задать настройки непосредственно в коде или вынести их в отдельный файл, для последующей загрузки.

Пример кода:

# -\*- coding: utf-8 -\*-   
import logging, logging.handlers, sys, os   
  
log=logging.getLogger('main')   
log.setLevel(logging.DEBUG)   
  
formatter=logging.Formatter('%(asctime)s.%(msecs)d %(levelname)s in \'%(module)s\' at line %(lineno)d: %(message)s','%Y-%m-%d %H:%M:%S')   
  
handler=logging.StreamHandler(sys.stderr)   
handler.setFormatter(formatter)   
handler.setLevel(logging.DEBUG)   
log.addHandler(handler)   
handler=logging.FileHandler('my.log', 'a')   
handler.setLevel(logging.INFO)   
handler.setFormatter(formatter)   
log.addHandler(handler)   
  
handler=logging.handlers.SMTPHandler('example.ru','user@example.ru','developer@example.ru','Critical error found')   
handler.setLevel(logging.CRITICAL)   
handler.setFormatter(formatter)   
log.addHandler(handler)   
  
log.log(1, 'low level message')   
log.debug("debug message")   
log.info("info message")   
log.warn("warn message")   
log.error("error message")   
try:   
    a=1/0   
except:   
    log.exception("exception message")   
log.critical("critical message")

Однако, более удобно и практично настройки логирования вывести в отдельный конфигурационный файл, а затем прочитать их. Приемущество такого подхода заключается в хорошем тоне программирования - в возможности изменения базового уровня сообщений не исправляя сам код программы.

Пример такого файла log.conf:

[loggers]   
keys=root   
  
[handlers]   
keys=handler1, handler2, handler3   
  
[formatters]   
keys = formatter1   
  
[logger\_root]   
qualname=main   
level=DEBUG   
handlers=handler1,handler2,handler3   
  
[handler\_handler1]   
class=StreamHandler   
level=DEBUG   
formatter=formatter1   
args=(sys.stdout,)   
  
[handler\_handler2]   
class=FileHandler   
level=INFO   
formatter=formatter1   
args=('my\_app.log','a')   
  
[handler\_handler3]   
class=handlers.SMTPHandler   
level=CRITICAL   
formatter=formatter1   
args=('mail.exammple.ru','user@exammple.ru','user@exammple.ru','Critical error found')   
  
[formatter\_formatter1]   
format=%(asctime)s.%(msecs)d %(levelname)s in '%(module)s' at line %(lineno)d: %(message)s   
datefmt=%Y-%m-%d %H:%M:%S   
class=logging.Formatter

Программный код будет выглядеть следующим образом:

# -\*- coding: utf-8 -\*-   
import logging, logging.config   
  
logging.config.fileConfig('log.conf')   
  
log=logging.getLogger('main')   
  
log.log(1, 'low level message')   
log.debug("debug message")   
log.info("info message")   
log.warn("warn message")   
log.error("error message")   
try:   
    a=1/0   
except:   
    log.exception("exception message")   
log.critical("critical message")

Результат в обоих случаях, как и следовало ожидать, идентичный. На консоль выведутся все сообщения с уровнем выше DEBUG (в нашем случае сообшение с уровнем 1 будет проигнорированно), в лог файл запишутся сообщения только с уровнем INFO и выше,а по почте отправится сообшение с уровнем CRITICAL:

2011-07-21 12:02:50.240 DEBUG in 'logging\_test2' at line 9: debug message   
2011-07-21 12:02:50.303 INFO in 'logging\_test2' at line 10: info message   
2011-07-21 12:02:50.303 WARNING in 'logging\_test2' at line 11: warn message   
2011-07-21 12:02:50.303 ERROR in 'logging\_test2' at line 12: error message   
2011-07-21 12:02:50.303 ERROR in 'logging\_test2' at line 16: exception message   
Traceback (most recent call last):   
  File "C:\Documents and Settings\mer\workspace\test\src\logging\_test2.py", line 14, in   
    a=1/0   
ZeroDivisionError: integer division or modulo by zero   
2011-07-21 12:02:50.303 CRITICAL in 'logging\_test2' at line 17: critical message

Настройки для сложных приложений

Для более стуктурно сложных и масштабных приложений logging предоставляет механизм иерархии логгеров, с фильтрацией сообщений. Однако я не увидел в этом достойного решения, поэтому не буду рассматривать его. Скажу что если вы испытываете нехватку функционала, попробуйте обратить внимание на другие библиотеки.