python的日志模块logging

ritow整理自 www.stamhe.com

1.简单的将日志打印到屏幕

```
1 import logging
2
3 logging.debug('This is debug message')
4 logging.info('This is info message')
5 logging.warning('This is warning message')
```

屏幕上打印:

```
WARNING:root:This is warning message
```

默认情况下，logging将日志打印到屏幕，日志级别为WARNING

日志级别大小关系为：CRITICAL > ERROR > WARNING > INFO > DEBUG > NOTSET

当然也可以自己定义日志级别。

2.通过logging.basicConfig函数对日志的输出格式及方式做相关配置

```
1  import logging
2
3  logging.basicConfig(level=logging.DEBUG,
4                      format='%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)
5                      datefmt='%a, %d %b %Y %H:%M:%S',
6                      filename='myapp.log',
7                      filemode='w')
8
9  logging.debug('This is debug message')
10 logging.info('This is info message')
11 logging.warning('This is warning message')
```

./myapp.log文件中内容为:

```
Sun, 24 May 2009 21:48:54 demo2.py[line:11] DEBUG This is debug message
Sun, 24 May 2009 21:48:54 demo2.py[line:12] INFO This is info message
Sun, 24 May 2009 21:48:54 demo2.py[line:13] WARNING This is warning message
```

logging.basicConfig函数各参数:
filename: 指定日志文件名
filemode: 和file函数意义相同，指定日志文件的打开模式，' w'或' a'
format: 指定输出的格式和内容，format可以输出很多有用信息，如上例所示:
%(levelno)s: 打印日志级别的数值
%(levelname)s: 打印日志级别名称
%(pathname)s: 打印当前执行程序的路径，其实就是sys.argv[0]
%(filename)s: 打印当前执行程序名
%(funcName)s: 打印日志的当前函数
%(lineno)d: 打印日志的当前行号
%(asctime)s: 打印日志的时间
%(thread)d: 打印线程ID
%(threadName)s: 打印线程名称
%(process)d: 打印进程ID
%(message)s: 打印日志信息
datefmt: 指定时间格式，同time.strftime()
level: 设置日志级别，默认为logging.WARNING
stream: 指定将日志的输出流，可以指定输出到sys.stderr,sys.stdout或者文件，默认输出到sys.stderr，当stream和filename同时指定时，stream被忽略

3.将日志同时输出到文件和屏幕

```python
import logging
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(me
                    datefmt='%a, %d %b %Y %H:%M:%S',
                    filename='myapp.log',
                    filemode='w')

################################################################################
#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
console = logging.StreamHandler()
console.setLevel(logging.INFO)
formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
console.setFormatter(formatter)
logging.getLogger('').addHandler(console)
```

```
15 ##########################################################################################
16
17 logging.debug('This is debug message')
18 logging.info('This is info message')
19 logging.warning('This is warning message')
```

屏幕上打印:

```
1 root        : INFO     This is info message
2 root        : WARNING  This is warning message
```

./myapp.log文件中内容为:

```
1 Sun, 24 May 2009 21:48:54 demo2.py[line:11] DEBUG This is debug message
2 Sun, 24 May 2009 21:48:54 demo2.py[line:12] INFO This is info message
3 Sun, 24 May 2009 21:48:54 demo2.py[line:13] WARNING This is warning message
```

4.logging之日志回滚

```
1  import logging
2  from logging.handlers import RotatingFileHandler
3
4  ##########################################################################################
5  #定义一个RotatingFileHandler，最多备份5个日志文件，每个日志文件最大10M
6  Rthandler = RotatingFileHandler('myapp.log', maxBytes=10*1024*1024,backupCount=
7  Rthandler.setLevel(logging.INFO)
8  formatter = logging.Formatter('%(name)-12s: %(levelname)-8s %(message)s')
9  Rthandler.setFormatter(formatter)
10 logging.getLogger('').addHandler(Rthandler)
11 ##########################################################################################
12
13 从上例和本例可以看出，logging有一个日志处理的主对象，其它处理方式都是通过addHandler添加进去
14 logging的几种handle方式如下：
15
16 logging.StreamHandler: 日志输出到流，可以是sys.stderr、sys.stdout或者文件
17 logging.FileHandler: 日志输出到文件
```

```
18
19 日志回滚方式，实际使用时用RotatingFileHandler和TimedRotatingFileHandler
20 logging.handlers.BaseRotatingHandler
21 logging.handlers.RotatingFileHandler
22 logging.handlers.TimedRotatingFileHandler
23
24 logging.handlers.SocketHandler: 远程输出日志到TCP/IP sockets
25 logging.handlers.DatagramHandler:  远程输出日志到UDP sockets
26 logging.handlers.SMTPHandler:  远程输出日志到邮件地址
27 logging.handlers.SysLogHandler: 日志输出到syslog
28 logging.handlers.NTEventLogHandler: 远程输出日志到Windows NT/2000/XP的事件日志
29 logging.handlers.MemoryHandler: 日志输出到内存中的制定buffer
30 logging.handlers.HTTPHandler: 通过"GET"或"POST"远程输出到HTTP服务器
```

由于StreamHandler和FileHandler是常用的日志处理方式，所以直接包含在logging模块中，而其他方式则包含在logging.handlers模块中，
上述其它处理方式的使用请参见python2.5手册！

5.通过logging.config模块配置日志

```
1  #logger.conf
2
3  ###############################################
4
5  [loggers]
6  keys=root,example01,example02
7
8  [logger_root]
9  level=DEBUG
10 handlers=hand01,hand02
11
12 [logger_example01]
13 handlers=hand01,hand02
14 qualname=example01
15 propagate=0
16
17 [logger_example02]
18 handlers=hand01,hand03
19 qualname=example02
20 propagate=0
```

```
21
22 #############################################
23
24 [handlers]
25 keys=hand01,hand02,hand03
26
27 [handler_hand01]
28 class=StreamHandler
29 level=INFO
30 formatter=form02
31 args=(sys.stderr,)
32
33 [handler_hand02]
34 class=FileHandler
35 level=DEBUG
36 formatter=form01
37 args=('myapp.log', 'a')
38
39 [handler_hand03]
40 class=handlers.RotatingFileHandler
41 level=INFO
42 formatter=form02
43 args=('myapp.log', 'a', 10*1024*1024, 5)
44
45 #############################################
46
47 [formatters]
48 keys=form01,form02
49
50 [formatter_form01]
51 format=%(asctime)s %(filename)s[line:%(lineno)d] %(levelname)s %(message)s
52 datefmt=%a, %d %b %Y %H:%M:%S
53
54 [formatter_form02]
55 format=%(name)-12s: %(levelname)-8s %(message)s
56 datefmt=
```

上例3：

```
1 import logging
2 import logging.config
```

```
3
4 logging.config.fileConfig("logger.conf")
5 logger = logging.getLogger("example01")
6
7 logger.debug('This is debug message')
8 logger.info('This is info message')
9 logger.warning('This is warning message')
```

上例4：

```
1 import logging
2 import logging.config
3
4 logging.config.fileConfig("logger.conf")
5 logger = logging.getLogger("example02")
6
7 logger.debug('This is debug message')
8 logger.info('This is info message')
9 logger.warning('This is warning message')
```

6.logging是线程安全的
7.logging在低版本（如 2.3.4）中用法有少许差别：

（因为我用 RedHat 4.3 里面自带的版本就是这个）

下面脚本将信息全部输入日志，无终端显示：

```
1  #!/usr/bin/env python
2  import logging
3  import os,sys
4  import time
5
6  MYNAME = os.path.splitext(os.path.basename(sys.argv[0]))[0]
7  MYDATE = time.strftime('%y%m%d')
8  LOG_FILE = os.path.join('/tmp',MYNAME + '.log.' + MYDATE)
9
10 logger = logging.getLogger()
11 logger.setLevel(logging.DEBUG)
```

```
12 fh = logging.FileHandler(LOG_FILE)
13 formatter = logging.Formatter("%(asctime)-15s %(filename)s [%(levelname)-8s] %(messa
14 fh.setFormatter(formatter)
15 logger.addHandler(fh)
16
17 logger.debug("This is debug message!!!")
18 logger.info("This is info message!!!")
19 logger.warn("This is warn message!!!")
20 logger.error("This is error message!!!")
21 logger.critical("This is critical message!!!")
```

下面脚本将日志输入日志，并可选择输出终端日志的级别：

```
1  #!/usr/bin/env python
2  import logging
3  import os,sys
4  import time
5
6  MYNAME = os.path.splitext(os.path.basename(sys.argv[0]))[0]
7  MYDATE = time.strftime('%y%m%d')
8  LOG_FILE = os.path.join('/tmp',MYNAME + '.log.' + MYDATE)
9
10 logger = logging.getLogger()
11 logger.setLevel(logging.DEBUG)
12 fh = logging.FileHandler(LOG_FILE)
13 formatter = logging.Formatter("%(asctime)-15s %(filename)s [%(levelname)-8s] %(messa
14 fh.setFormatter(formatter)
15 logger.addHandler(fh)
16
17 logger.debug("This is debug message!!!")
18 logger.info("This is info message!!!")
19 logger.warn("This is warn message!!!")
20 logger.error("This is error message!!!")
21 logger.critical("This is critical message!!!")
```