logging模块

一、日志级别

```
import logging

logging.debug('调试日志')
logging.info('消息日志')
logging.warning('警告日志')
logging.error('错误日志')
logging.critical('严重错误日志')
```

二、日志配置

```
import logging

# 日志基本配置
logging.basicConfig(
    # 1、日志级别
    level=30,
    # DEBUG:10
    # INFO:20
    # WARNING:30
    # ERROR:40
    # CRITICAL:50

    # 2、日志输出格式
    # format='%(asctime)s %(name)s [%(pathname)s line:%(lineno)d] %(levelname)s %
(message)s',

    # 3、asctime的时间格式
    # datefmt='%Y-%m-%d %H:%M:%S',

    # 4、日志输出位置：终端/文件
    # filename='user.log',  # 不指定此配置，默认打印到终端
)


'''
%(name)s         Logger的名字(getlogger时指定的名字)
%(levelno)s 数字形式的日志级别
%(levelname)s 文本形式的日志级别
%(pathname)s     调用日志输出日志的完整路径名
%(filename)s     调用日志输出日志的文件名
%(module)s   调用日志输出日志的模块名
%(funcName)s     调用日志输出日志的函数名
%(lineno)d   调用日志输出函数的语句所在的代码行
```

```
%(created)f 当前时间，用UNIX标准的表示时间的浮 点数表示
%(relativeCreated)d 输出日志信息时的，自Logger创建以 来的毫秒数
%(asctime)s 字符串形式的当前时间,默认格式是 “2022-07-30 22:15:53,394”
%(thread)d  线程ID,可能没有
%(threadName)s  线程名,可能没有
%(process)d 进程ID,可能没有
%(message)s 用户输出的消息
'''



'''
logging模块有三个比较重要的功能组件：
    1、loggers 配置文件可定义一些输出日志的appname
    2、handler 配置日志的分隔大小,输出位置,日志文件创建等
    3、formatters 配置日志输出的格式
'''
# 日志配置字典
LOGGING_DIC = {
    'version': 1.0,
    'disable_existing_loggers': False,
    # 日志格式
    'formatters': {
        'standard': {
            'format': '%(asctime)s %(threadName)s:%(thread)d [%(name)s] %
(levelname)s [%(pathname)s:%(lineno)d] %(message)s',
            'datefmt': '%Y-%m-%d %H:%M:%S',
        },
        'simple': {
            'format': '%(asctime)s [%(name)s] %(levelname)s  %(message)s',
            'datefmt': '%Y-%m-%d %H:%M:%S',
        },
        'test': {
            'format': '%(asctime)s %(message)s',
        },
    },
    'filters': {},
    # 日志处理器
    'handlers': {
        'console_debug_handler': {
            'level': 'DEBUG',  # 日志处理的级别限制
            'class': 'logging.StreamHandler',  # 输出到终端
            'formatter': 'simple'  # 日志格式
        },
        'file_info_handler': {
            'level': 'INFO',
            'class': 'logging.handlers.RotatingFileHandler',  # 保存到文件,日志轮转
            'filename': 'user.log',
            'maxBytes': 1024*1024*10,  # 日志大小 10M
            'backupCount': 10,  # 日志文件保存数量限制
            'encoding': 'utf-8',
            'formatter': 'standard',
        },
        'file_debug_handler': {
            'level': 'DEBUG',
```

```
            'class': 'logging.FileHandler',  # 保存到文件
            'filename': 'test.log',  # 日志存放的路径
            'encoding': 'utf-8',    # 日志文件的编码
            'formatter': 'test',
        },
    },
    # 日志记录器
    'loggers': {
        'logger1': {  # 导入时logging.getLogger时使用的app_name
            'handlers': ['console_debug_handler'],  # 日志分配到哪个handlers中
            'level': 'DEBUG',  # 日志记录的级别限制
            'propagate': False,  # 默认为True，向上（更高级别的logger）传递，设置为
False即可，否则会一份日志向上层层传递
        },
        'logger2': {
            'handlers': ['console_debug_handler', 'file_debug_handler'],
            'level': 'INFO',
            'propagate': False,
        },
    }
}

# 注：进行日志轮转的日志文件，不能和其他handler共用，不然会导致文件被占用无法更名而报错！
```

三、日志使用

```
import logging.config
import settings

logging.config.dictConfig(settings.LOGGING_DIC)

logger1 = logging.getLogger('logger1')
logger1.info('xxx登录了')

logger2 = logging.getLogger('logger2')
logger2.info('xxx充值了5毛钱')
```

配置文件方式

1、创建一个,以.cfg结尾 或以.ini结尾的配置文件

> log.cfg

2、配置文件内容

```
# 定义logger
[loggers]
```

```
keys=root,error,info  #创建三个app名,root是父类，必需存在的(和配置字典里面的空名字logger一样，在getLogger的时候，找不到对应的logger名字的，则使用root这个logger)


[logger_root]       #创建完的app名我们要定义一些规则,严格要求格式为"logger_appname"
level=DEBUG        #设置日志级别
qualname=root       #在"root"下的appname可填可不填
handlers=console,file     #设置指定过滤器,多个以逗号分隔

[logger_error]
level=ERROR
qualname=error      #除了root下的appname以外，所有的app必须要设置这个属性，用于定义打印输出时候的app名
handlers=console
propagate=0      # 禁止日志向父类传递

[logger_info]
level=INFO
qualname=info
handlers=file,rotating_file
propagate=0


# 定义handler
[handlers]
keys=console,file,rotating_file  #定义过滤器名称（keysname），下面定义以handler_keysname格式定义，引用的时候用keysname

[handler_console]
class=StreamHandler
level=DEBUG
formatter=simple


[handler_file]
class=FileHandler
level=DEBUG
formatter=standard
args=('info.log','a', 'utf-8')     #创建文件名字,以什么方式打开，并指定编码方式

[handler_rotating_file]
class=handlers.RotatingFileHandler  # 日志轮转
level=DEBUG
formatter=standard
args=('user.log','a')
kwargs={'maxBytes':1024*1024*10,'backupCount':2,'encoding':'utf-8'} #日志大小10M,文件保存数量2，编码方式utf-8



# 定义formatter
[formatters]
keys=standard,simple   #定义格式名称（keysname），下面定义以formatter_keysname格式定义，引用的时候用keysname
```

```
[formatter_standard]
format=%(asctime)s %(threadName)s:%(thread)d [%(name)s] %(levelname)s [%
(pathname)s:%(lineno)d] %(message)s
datefmt=%Y-%m-%d %H:%M:%S

[formatter_simple]
format=%(asctime)s [%(name)s] %(levelname)s  %(message)s
datefmt=%Y-%m-%d %H:%M:%S

# 注：注释必须单独写在一行，不能跟在配置项后面，我这里只是为了方便注释说明,你用的时候需要把注释去掉，或者改到单独一行去。
```

3、 使用

```
import logging.config
logging.config.fileConfig('log.conf')    # 和日志配置字典用法一样，只是这一步使用fileConfig来加载篇日志文件就好了

logger = logging.getLogger('info')
logger.info('xxx充值了5毛钱')
```

注：使用fileConfig读取log.conf时，无法设置字符编码，如果你的文件是utf-8编码，且在windows系统中运行，就报如下错误：

```
Traceback (most recent call last):
  File "D:/teach/42-日志管理.py", line 9, in <module>
    logging.config.fileConfig('logging.cfg')
  File "D:\Python37\lib\logging\config.py", line 69, in fileConfig
    cp.read(fname)
  File "D:\Python37\lib\configparser.py", line 696, in read
    self._read(fp, filename)
  File "D:\Python37\lib\configparser.py", line 1014, in _read
    for lineno, line in enumerate(fp, start=1):
UnicodeDecodeError: 'gbk' codec can't decode byte 0xa8 in position 1044: illegal
 multibyte sequence
```

```
'''
解决方法：
    需要改源码(把read函数的encoding参数默认的None，改成utf-8)：
    fileConfig() -> cp.read(fname) -> def read(self, filenames, encoding='utf-
8'):
    只有windows系统需要改此源码，其他系统不用改，或者你的配置文件本身就是gbk编码的，也不用改此选项
    总之改源码不太好，所以我们一般用配置字典，比较少用配置文件
'''
```