requests二次封装_捕获异常

1、封装类无构造函数，实例化类不需传参，在使用对应方法时进行传参

```
import  requests
import  nnlog
import  time

class MyRequests:
    #类实例方法--post
    def post_method(self,url,data=None,header=None,file=None,format=1): #类实例方法带传参
        try: #post请求异常捕获
            self.result=requests.post(url,data,headers=header,files=file)
        except Exception as error_info:
            self.write_log(error_info) #如果有异常调用日志模块写入日志文件
        else:
            return self.format_result(format) #如果没有错误根据用户传入的格式ID，返回对应得格式
    def get_method(self,url,data=None,header=None,file=None,format=1): #类实例方法带传参
        try: #get请求异常捕获
            self.result= requests.get(url,data,headers=header,files=file)
        except Exception as error_info:
            self.write_log(error_info) #如果有异常调用日志模块写入日志文件
        else:
            return  self.format_result(format) #如果没有错误根据用户传入的格式ID，返回对应得格式

    @classmethod  #类方法
    def write_log(cls,error_info):
        log = nnlog.Logger('test.log', level='info', backCount=5, when='d') #实例化日志类
        log.info(error_info) #调用日志的信息级别方法
        time.sleep(4)

    def format_result(self,format): #结果返回类型类实例方法
        if  format==1:
            return  self.result.text   #返回字符串类型
        elif format==2:
            return  self.result.content.decode('utf-8') #返回byte类型方法
        elif format==3:
            return self.result.json() #返回字典类型
```

```
        else:
            return print("无您需要的格式") #输入的类型错误
```

2、封装类有构造函数，实例化类需传参，在使用对应方法时不传参直接调用方法

```
import  requests
import  nnlog
import  time


class Requests:
    #构造函数，实例化时需要把传参传入
    def __init__(self,url,data=None,header=None,file=None,format=1):
            self.url = url
            self.data = data
            self.header = header
            self.file = file
            self.format=format


    def post_method(self): #post类实例方法
        try: #post请求异常捕获
           self.result=requests.post(self.url,self.data,headers=self.header,files=self.file)
        except Exception as error_info:
            self.write_log(error_info)  #如果有异常调用日志模块写入日志文件
        else:
            return  self.format_result()#如果没有错误根据用户传入的格式ID，返回对应得格式


    def get_method(self):
        try:#get请求异常捕获
            self.result= requests.get(self.url, self.data, headers=self.header, files=self.file)
        except Exception as error_info:
            self.write_log(error_info) #如果有异常调用日志模块写入日志文件
        else:
            return self.format_result() #如果没有错误根据用户传入的格式ID，返回对应得格式


    @classmethod
    def write_log(cls,error_info):
        log = nnlog.Logger('test.log', level='info', backCount=5, when='d')#实例化日志类
```

```
        log.info(error_info)#调用日志的信息级别方法
        time.sleep(4)


    def format_result(self): #结果返回类型类实例方法
        if self.format==1:
            return  self.result.text #返回字符串类型
        elif self.format==2:
            return  self.result.content.decode('utf-8')#返回byte类型方法
        elif self.format==3:
            return self.result.json() #返回字典类型
        else:
            return print("无您需要的格式")
```