**9.2日志处理·请求日志**

请求日志，即访问Flask服务的每一个请求，我们需要记录请求的基本信息，比如请求地址、请求时间、请求时长、请求头、请求参数等，以便后续对请求进行统计分析，当然，也方便我们对问题进行核查。

# 记录的信息

|  |  |
| --- | --- |
| 名称 | 描述 |
| current\_user | 当前用户信息 |
| ip | 客户端ip |
| url | 请求地址 |
| start\_time | 请求开始时间 |
| process | 当前进程id |
| thread | 当前线程id |
| body | 请求正文参数 |
| res\_data | 响应数据 |
| end\_time | 结束时间 |
| interval | 请求响应时间 |

# 关于请求拦截

请求拦截可以使用前面的@app.before\_request和@app.after\_request装饰器的方式，为了app.py文件更加简洁，本文使用了新的编写方式。

# 新增mlogging/request\_log.py

配置说明：

* REQUEST\_LOG\_ENABLE 是否启用请求日志记录
* IGNORE\_LIST 不做请求日志的接口列表，例：['/user/get','/user/list']

import datetime  
import os  
import threading  
  
from flask import request, g  
  
  
class RequestLog:  
 """  
 请求日志处理  
 """  
 # 是否启用请求日志  
 request\_log\_enable = False  
 # 不做请求日志的接口列表，例：['/user/get','/user/list']  
 ignore\_list = []  
  
 def \_\_init\_\_(self, app=None):  
 self.app = app  
 if app:  
 self.init\_app(app)  
  
 def init\_app(self, app):  
 self.app = app  
 self.request\_log\_enable = app.config.get("REQUEST\_LOG\_ENABLE")  
 if app.config.get('IGNORE\_LIST'):  
 self.ignore\_list = app.config.get('IGNORE\_LIST')  
 if self.request\_log\_enable:  
 self.app.before\_request(self.\_before\_request)  
 self.app.after\_request(self.\_after\_request)  
  
 def \_before\_request(self):  
 if request.path not in self.ignore\_list:  
 g.ip = request.remote\_addr # 客户端ip  
 g.url = request.url # 请求地址  
 g.start\_time = datetime.datetime.now() # 请求开始时间  
 g.process = os.getpid() # 进程id  
 g.thread = threading.currentThread().ident # 线程id  
 if request.is\_json:  
 g.body = request.get\_json() # 请求正文参数  
 else:  
 g.body = request.get\_data()  
 if g.body:  
 g.body = str(g.body, encoding="utf8")  
 g.body = g.body.replace("\n\r", "")  
  
 def \_after\_request(self, response):  
 if request.path not in self.ignore\_list:  
 g.res\_data = str(response.data, encoding="utf8") # 响应数据  
 g.end\_time = datetime.datetime.now() # 请求结束时间  
 g.interval = g.end\_time - g.start\_time # 请求处理时长  
 g.interval = str(g.interval).replace("0:00:", "")  
 if hasattr(g, 'current\_user'): # 当前登录用户信息  
 self.app.logger.info(  
 f"[{g.current\_user}][{g.ip}][{g.url}][{g.start\_time}][{g.process}][{g.thread}][{g.body}][{g.res\_data}][{g.end\_time}][{g.interval}]")  
 else:  
 self.app.logger.info(  
 f"[][{g.ip}][{g.url}][{g.start\_time}][{g.process}][{g.thread}][{g.body}][{g.res\_data}][{g.end\_time}][{g.interval}]")  
 return response

# 修改config.py

新增请求日志配置

代码片段：

class BaseConfig(ｏｂｊｅｃｔ):  
 """  
 基础配置  
 """  
 #### 省略其他配置……  
 # ------请求日志配置------  
 # 是否启用请求日志记录  
 REQUEST\_LOG\_ENABLE = bool(os.getenv("REQUEST\_LOG\_ENABLE", True))  
 # 不做请求日志的接口，例：['/user/get','/user/list']  
 IGNORE\_LIST = []  
 # ------请求日志配置------

# 修改app.py

初始化日志配置

代码片段：

from mlogging.request\_log import RequestLog  
  
# 配置请求日志  
RequestLog(app)

# 运行Flask服务

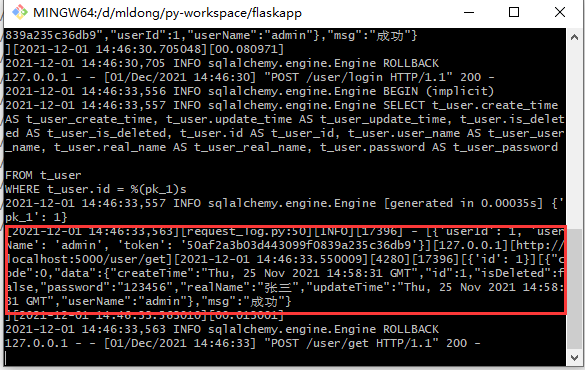
默认为控制台输出

flask run

# 使用Postman接口测试工具访问

## 访问/user/login

## 访问/user/get



样例数据：

[2021-12-01 14:24:16,736][request\_log.py:50][INFO][11628] - [{'userId': 1, 'userName': 'admin', 'token': '8b4b555f5bf04fd9be9a0f0064883e31'}][127.0.0.1][http://localhost:5000/user/get][2021-12-01 14:24:16.666431][1488][11628][{'id': 1}][{"code":0,"data":{"createTime":"Thu, 25 Nov 2021 14:58:31 GMT","id":1,"isDeleted":false,"password":"123456","realName":"张三","updateTime":"Thu, 25 Nov 2021 14:58:31 GMT","userName":"admin"},"msg":"成功"}  
][2021-12-01 14:24:16.736428][00.069997]

# 一些优化建议

* 日志数据脱敏

像一些手机号、邮箱、身份证号、密码等信息，正常来说，是不允许在日志上以明文的方式打印的，需要做一些脱敏策略。

* 请求日志持久化

请求日志是规范化的数据，如有需要，可以存放到数据库中。当然为了减小IO操作对程序效率的影响，也为了避免多进程进行IO操作时带来的冲突，可以先把日志存到redis中，之后再用异步线程或者单独的进程将redis中的日志按照日期读到磁盘文件中。