**Python 关于requests 关闭连接，优化内存**

**问题场景**  
项目需要通过一台主机搭上代理向目标服务器定时交换数据，主机上同时跑着比较占用资源的其他项目。

**项目设计思路**

因为主机上已经运行了比较占用线程内存资源的其他项目，所以计划本项目就应该在不能太占用资源，的前提下保持比较流畅的运行。

设计了在线程下加上定时器的思路，定时运行相关网络请求方法。

请求量 ：每隔5秒钟，发送10 ~ 30个请求。

**核心代码**

headers = {"Content-Type": "application/json;charset=uf8"}

url = config.baseurl + path

try:

response = requests.post(url, data= data, headers=headers ,stream= False,timeout= 10)

except Exception as indentfier:

time.sleep(5)

getLoaclObjs(False)

par = json.loads(data)

if response != None:

print(" method name = ", par["IPCType"],"\n sendData = ",json.dumps(par), "\n response.state.code = ",response.status\_code,"\n response.text = ",response.text,"\n\n", url, "\n\n")

**问题：**

**报错：**

HTTPConnectionPool(host='172.19.5.101', port=8085): Max retries exceeded with url: <url >(Caused by ConnectTimeoutError(<urllib3.connection.HTTPConnection object at 0x006DB1F0>, 'Connection to <server ip> timed out. (connect timeout=10)'))

**分析：**  
报错的重点在于 connect timeout

**解决方案**

1.由于是timeout的报错，所以为了项目稳定首先要做的就是做好重连操作，在代码中其实已经做好了 try … except 中 except中为错误处理。再次调用了方法。

2.处理过后数据交互没有了毛病，但是报错仍然频繁，甚至后边有了新的错误：

requests.exceptions.ConnectionError: ('Connection aborted.', ConnectionAbortedError(10053, '你的主机中的软件中止了一个已建立的连接。', None, 10053, None))

**度娘和overflow都告诉我添加一下代码：**

#设置重连次数

requests.adapters.DEFAULT\_RETRIES = 15

# 设置连接活跃状态为False

s = requests.session()

s.keep\_alive = False

基本上意思就是发出的网络请求比较频繁，requests内置的urllibs3不能及时的为我们断开连接，于是就有了以上报错。  
代码的基本意思就是断开连接、增加重试次数。

headers = {"Content-Type": "application/json;charset=uf8"}

url = config.baseurl + path

try:

#设置重连次数

requests.adapters.DEFAULT\_RETRIES = 15

# 设置连接活跃状态为False

s = requests.session()

s.keep\_alive = False

response = requests.post(url, data= data, headers=headers ,stream= False,timeout= 10)

except Exception as indentfier:

time.sleep(5)

getLoaclObjs(False)

par = json.loads(data)

if response != None:

print(" method name = ", par["IPCType"],"\n sendData = ",json.dumps(par), "\n response.state.code = ",response.status\_code,"\n response.text = ",response.text,"\n\n", url, "\n\n")

还有别的处理方法就是增加代理，由于主机是用代理的，所以项目换别的代理就不太现实，否则有可能直接连接不上目标服务器。

于是想到了直接管理内存,文档上是这么表述的：

好消息——归功于 urllib3，同一会话内的持久连接是完全自动处理的！同一会话内你发出的任何请求都会自动复用恰当的连接！  
注意：只有所有的响应体数据被读取完毕连接才会被释放为连接池；所以确保将 stream 设置为 False 或读取 Response 对象的  
content 属性。

也就是说除非stream=False或者response.content /response.text被操作，不然该请求不会被释放。  
同其他的自动管理内存的语言一样，不论垃圾回收还是引用计数，都存在一个问题就是内存释放比较缓慢，于是我就在代码中手动关闭了连接。

headers = {'Content-Type': "application/json;charset=uf8"}

url = baseurl + path

response = None

try:

#设置重连次数

requests.adapters.DEFAULT\_RETRIES = 5

s = requests.session()

# 设置连接活跃状态为False

s.keep\_alive = False

response = requests.post(url, data=data, headers=headers,stream=False,timeout= 10)

# 关闭请求 释放内存

response.close()

del(response)

except Exception as indentfier:

time.sleep(5)

timer = threading.Timer(timerFlag, upload\_position)

timer.start()

最终有效减少了报错次数。

**上下文管理器**

在overflow我还看到另一种解决办法：  
https://stackoom.com/question/gRPC/Python-Requests关闭http连接

我来到这个问题寻找解决"too many open files" error ，但我在我的代码中使用requests.session() 。  
稍后进行一些搜索，我想出了Python请求文档的答案，该文档建议使用with块，以便即使存在未处理的异常，会话也会关闭：

with requests.Session() as s:

s.get('http://google.com')

如果您没有使用Session，您实际上可以做同样的事情： http ：  
//docs.python-requests.org/en/master/api/#requests.Response.close

with requests.get('http://httpbin.org/get', stream=True) as r:

# Do something