**爬虫5 带参数请求数据**

复习

Network能够记录浏览器的所有请求。我们最常用的是：ALL（查看全部）/XHR（仅查看XHR）/Doc（Document，第0个请求一般在这里），有时候也会看看：Img（仅查看图片）/Media（仅查看媒体文件）/Other（其他）。最后，JS和CSS，则是前端代码，负责发起请求和页面实现；Font是文字的字体；而理解WS和Manifest，需要网络编程的知识，倘若不是专门做这个，你不需要了解。



在Network，有非常重要的一类请求是XHR（或Fetch），因为有它的存在，人们不必刷新/跳转网页，即可加载新的内容。随着技术发展，XHR的应用频率越来越高，我们常常需要在这里找我们想要的数据。

XHR的功能是传输数据，其中有非常重要的一种数据是用json格式写成的，和html一样，这种数据能够有组织地存储大量内容。json的数据类型是“文本”，在Python语言当中，我们把它称为字符串。我们能够非常轻易地将json格式的数据转化为列表/字典，也能将列表/字典转为json格式的数据。

如何解析json数据？答案如下：



总之，在上一关，我们最后把代码写成这副模样：

import requests

# 引用requests模块

res\_music = requests.get('https://c.y.qq.com/soso/fcgi-bin/client\_search\_cp?ct=24&qqmusic\_ver=1298&new\_json=1&remoteplace=txt.yqq.song&searchid=60997426243444153&t=0&aggr=1&cr=1&catZhida=1&lossless=0&flag\_qc=0&p=1&n=20&w=%E5%91%A8%E6%9D%B0%E4%BC%A6&g\_tk=5381&loginUin=0&hostUin=0&format=json&inCharset=utf8&outCharset=utf-8&notice=0&platform=yqq.json&needNewCode=0')

# 调用get方法，下载这个字典

json\_music = res\_music.json()

# 使用json()方法，将response对象，转为列表/字典

list\_music = json\_music['data']['song']['list']

# 一层一层地取字典，获取歌单列表

for music in list\_music:

# list\_music是一个列表，music是它里面的元素

print(music['name'])

# 以name为键，查找歌曲名

print('所属专辑：'+music['album']['name'])

# 查找专辑名

print('播放时长：'+str(music['interval'])+'秒')

# 查找播放时长

print('播放链接：https://y.qq.com/n/yqq/song/'+music['mid']+'.html\n\n')

# 查找播放链接

项目：狂热粉丝

接上一关接着说，仅仅只是拿到20首歌曲的相关信息，不能让一个狂热的粉丝感到喜悦。他会在心里呐喊：“我，全都要！”

那就全都给他吧！在这一关，我们要完成的项目是：让上一关的代码得到进化，使它能爬取很多很多歌曲，而不只是爬取20个。

同时，作为附赠，我们还要以歌曲《七里香》作为案例，去爬取它的歌曲评论。就像：



当然了，如果你开心，我们还可以把“爬取歌词”作为本关卡的练习。

我们相信这样做，能够让一个狂热的粉丝感到欢喜。

在这个过程当中，我们会学到带参数请求数据的知识，以及关于Requests Headers的知识。此刻你或许还不清楚我在说什么，没关系，答案很快会揭晓。

分析过程

爬取歌曲信息也好，爬取歌词评论也好，对今天的我们来说，都不算难题。难题是：怎么翻页啊！



解决这个问题，需要用到“带参数请求数据”。学会这个，就不再有障碍。

什么是带参数请求数据

我不知道你有没有认真地观察过一个完整url的组成，如果没有，我们现在来试试看：

当你在豆瓣搜索“海边的卡夫卡”，它的网址会是这样：

<https://www.douban.com/search?q=%E6%B5%B7%E8%BE%B9%E7%9A%84%E5%8D%A1%E5%A4%AB%E5%8D%A1>

当你在知乎搜索“宇宙大爆炸”，它的网址会是这样：

<https://www.zhihu.com/search?type=content&q=%E5%AE%87%E5%AE%99%E5%A4%A7%E7%88%86%E7%82%B8>

当你在QQ音乐搜索“周杰伦”，它的网址会是这样：

<https://y.qq.com/portal/search.html#page=1&searchid=1&remoteplace=txt.yqq.top&t=song&w=%E5%91%A8%E6%9D%B0%E4%BC%A6>

现在，我要揭晓规律：

在上面，我们能看到每个url都由两部分组成。前半部分大多形如：[https://xx.xx.xxx/xxx/xxx](https://xx.xx.xxx/xxx/xxx" \t "_blank)

后半部分，多形如：xx=xx&xx=xxx&xxxxx=xx&……

两部分使用?来连接。举例刚刚的豆瓣网址，前半部分就是：

<https://www.douban.com/search>

后半部分则是：q=%E6%B5%B7%E8%BE%B9%E7%9A%84%E5%8D%A1%E5%A4%AB%E5%8D%A1

它们的中间使用了?来隔开。

这前半部分是我们所请求的地址，它告诉服务器，我想访问这里。而后半部分，就是我们的请求所附带的参数，它会告诉服务器，我们想要什么。这参数的结构，会和字典很像，有键有值，键值用=连接；每组键值之间，使用&来连接。

就像豆瓣。我们请求的地址是[https://www.douban.com/search，而我们的请求所附带的参数是“海边的卡夫卡”(那段你看不懂的代码，它是“海边的卡夫卡”使用utf-8编码的结果)。](https://www.douban.com/search%EF%BC%8C%E8%80%8C%E6%88%91%E4%BB%AC%E7%9A%84%E8%AF%B7%E6%B1%82%E6%89%80%E9%99%84%E5%B8%A6%E7%9A%84%E5%8F%82%E6%95%B0%E6%98%AF%E2%80%9C%E6%B5%B7%E8%BE%B9%E7%9A%84%E5%8D%A1%E5%A4%AB%E5%8D%A1%E2%80%9D(%E9%82%A3%E6%AE%B5%E4%BD%A0%E7%9C%8B%E4%B8%8D%E6%87%82%E7%9A%84%E4%BB%A3%E7%A0%81%EF%BC%8C%E5%AE%83%E6%98%AF%E2%80%9C%E6%B5%B7%E8%BE%B9%E7%9A%84%E5%8D%A1%E5%A4%AB%E5%8D%A1%E2%80%9D%E4%BD%BF%E7%94%A8utf-8%E7%BC%96%E7%A0%81%E7%9A%84%E7%BB%93%E6%9E%9C)%E3%80%82)

现在，再回过头看我们上一关所请求的url，会不会感觉有不同的理解？

<https://c.y.qq.com/soso/fcgi-bin/client_search_cp?ct=24&qqmusic_ver=1298&new_json=1&remoteplace=txt.yqq.song&searchid=60997426243444153&t=0&aggr=1&cr=1&catZhida=1&lossless=0&flag_qc=0&p=1&n=20&w=%E5%91%A8%E6%9D%B0%E4%BC%A6&g_tk=5381&loginUin=0&hostUin=0&format=json&inCharset=utf8&outCharset=utf-8&notice=0&platform=yqq.json&needNewCode=0>

显然，这样一个长链接，阅读体验非常之差。Network面板提供了一个更友好的查看方式，我来带你看看它。

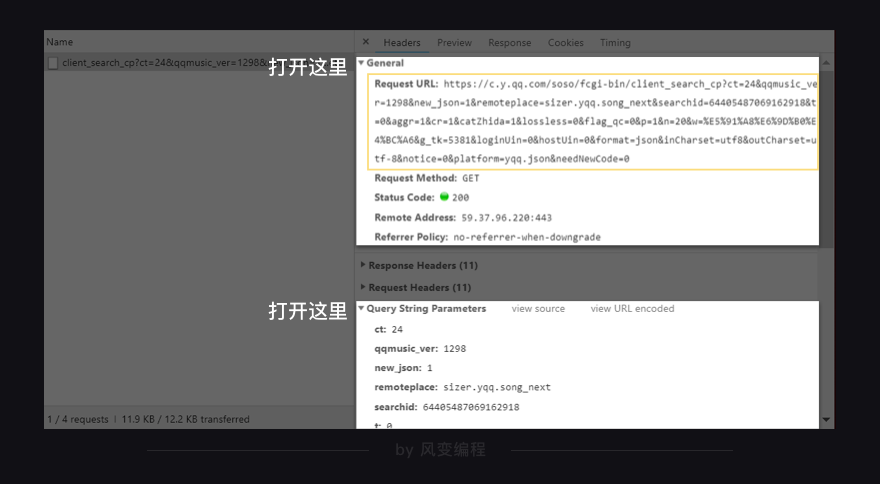
重温我们在上一关的步骤：请你使用浏览器打开QQ音乐官网，搜索周杰伦。然后打开检查面板里的Network，刷新这个页面。当然，你也可以点击这个链接直达：

<https://y.qq.com/portal/search.html#page=2&searchid=1&remoteplace=txt.yqq.top&t=song&w=%E5%91%A8%E6%9D%B0%E4%BC%A6>

找到隐藏有歌曲列表的那个XHR，它的名字应该叫client\_search……之类，点开它，选中Headers，你的页面看上去会像这样：



我们来到一个熟悉的界面，你能看到Request Url，很长的一个地址。现在，保持General打开，保持Response Headers和Request Headers关闭。我们点开Query String Parametres。



有没有感觉一些眼熟？它里面的内容，正是请求所附带的参数，Query String Parametres，它的中文翻译是：查询字符串参数。

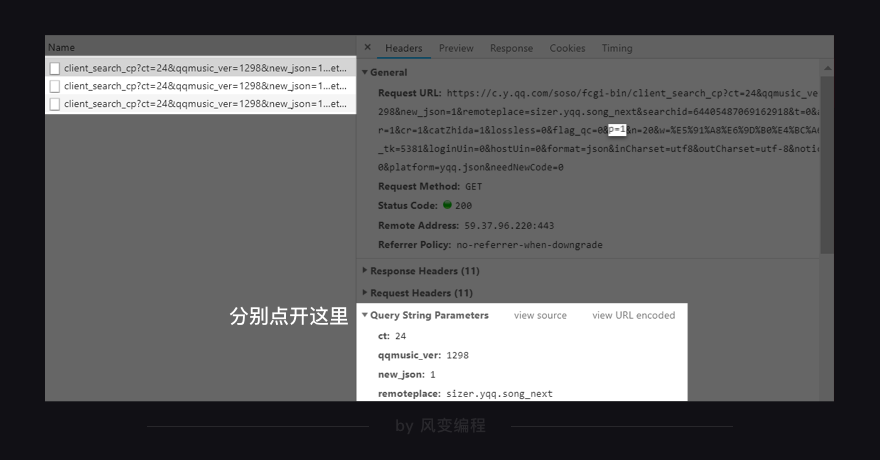
这个面板用类似字典的形式，呈现了各个参数的键值，阅读体验会好一些。我们可以常来此处看看。

如何带参数请求数据

在上一关当中，我们直接用requests.get()请求了url。在这一关，如果我们想拿到更多歌曲清单。就要读懂它的各个参数。修改它们，重新发起请求。

读懂参数，有两个重要的方法是“观察”和“比较”。“观察”指的是阅读参数的键与值，尝试理解它的含义。“比较”指的是比较两个相近的XHR——它们有哪些不同，对应的页面显示内容有什么不同。

现在，我们来观察比较，不同页码的歌曲列表，它们url的不同。依然在搜索“周杰伦”的结果页，点击第2页、第3页进行翻页，此时Network会多加载出2个XHR，它们的Name都是client\_search…。



分别点开它们的Query String Parametres，比较参数之间有什么不同。点击Enter（回车）键，告诉你答案。

在此，只有一个参数变化。这个参数是p。第1页p的值为1，第二、第三页的值则为2和3。说明p代表的应该就是页码。

如果你再去对比它们的url，也能发现同样的事实：url整个儿都是一样的，只有p的值发生变化。

按照之前的知识，你大约会想：我们去写一个循环，每次循环去更改p的值，这样不就能实现爬取好多好多歌曲了吗？

代码实现

这样想是完全正确的。你可以尝试这样做。来，改写下方代码，实现这一点。备注：虽然这样能爬到我们的数据，但我并不推荐你在课堂里把所有的歌曲信息都爬取下来。一来是等待时间较长，二来虽然QQ音乐没有对爬虫做限制，但是我们也应该尽可能减少对服务器的压力。所以，循环不要超过**5个**。

import requests

# 引用requests模块

for x in range(5):

res\_music = requests.get('https://c.y.qq.com/soso/fcgi-bin/client\_search\_cp?ct=24&qqmusic\_ver=1298&new\_json=1&remoteplace=txt.yqq.song&searchid=60997426243444153&t=0&aggr=1&cr=1&catZhida=1&lossless=0&flag\_qc=0&p='+str(x+1)+'&n=20&w=%E5%91%A8%E6%9D%B0%E4%BC%A6&g\_tk=5381&loginUin=0&hostUin=0&format=json&inCharset=utf8&outCharset=utf-8&notice=0&platform=yqq.json&needNewCode=0')

# 调用get方法，下载这个字典

json\_music = res\_music.json()

# 使用json()方法，将response对象，转为列表/字典

list\_music = json\_music['data']['song']['list']

# 一层一层地取字典，获取歌单列表

for music in list\_music:

# list\_music是一个列表，music是它里面的元素

print(music['name'])

# 以name为键，查找歌曲名

print('所属专辑：'+music['album']['name'])

# 查找专辑名

print('播放时长：'+str(music['interval'])+'秒')

# 查找播放时长

print('播放链接：https://y.qq.com/n/yqq/song/'+music['mid']+'.html\n\n')

# 查找播放链接

这样写代码，的确能够完成我们的目标。但是，这样写代码显然不够优雅——它太长了。

素有“人工智能之父”的唐纳德·克努特（Donald Ervin Knuth）讲过一句话让我印象深刻，他说：代码写出来是让人读的，只是顺便拿去让机器执行。



我们来让这个代码变好看些。事实上，requests模块里的requests.get()提供了一个参数叫params，可以让我们用字典的形式，把参数传进去。它的官方文档，是这样描述：



所以，其实我们可以把Query String Parametres里的内容，直接复制下来，封装为一个字典，传递给params。只是有一点要特别注意：要给他们打引号，让它们变字符串。

所以，代码最后可能长这样：

import requests

# 引用requests模块

url = 'https://c.y.qq.com/soso/fcgi-bin/client\_search\_cp'

for x in range(5):

params = {

'ct':'24',

'qqmusic\_ver': '1298',

'new\_json':'1',

'remoteplace':'sizer.yqq.song\_next',

'searchid':'64405487069162918',

't':'0',

'aggr':'1',

'cr':'1',

'catZhida':'1',

'lossless':'0',

'flag\_qc':'0',

'p':str(x+1),

'n':'20',

'w':'周杰伦',

'g\_tk':'5381',

'loginUin':'0',

'hostUin':'0',

'format':'json',

'inCharset':'utf8',

'outCharset':'utf-8',

'notice':'0',

'platform':'yqq.json',

'needNewCode':'0'

}

# 将参数封装为字典

res\_music = requests.get(url,params=params)

# 调用get方法，下载这个字典

json\_music = res\_music.json()

# 使用json()方法，将response对象，转为列表/字典

list\_music = json\_music['data']['song']['list']

# 一层一层地取字典，获取歌单列表

for music in list\_music:

# list\_music是一个列表，music是它里面的元素

print(music['name'])

# 以name为键，查找歌曲名

print('所属专辑：'+music['album']['name'])

# 查找专辑名

print('播放时长：'+str(music['interval'])+'秒')

# 查找播放时长

print('播放链接：https://y.qq.com/n/yqq/song/'+music['mid']+'.html\n\n')

# 查找播放链接

悄悄地告诉你，如果你将这个代码里'w'参数的值换成另一个喜欢的歌手，那么它也能爬到这个歌手的所有歌曲信息。如果你愿意，可以在本关卡结束后，练习做这件事。

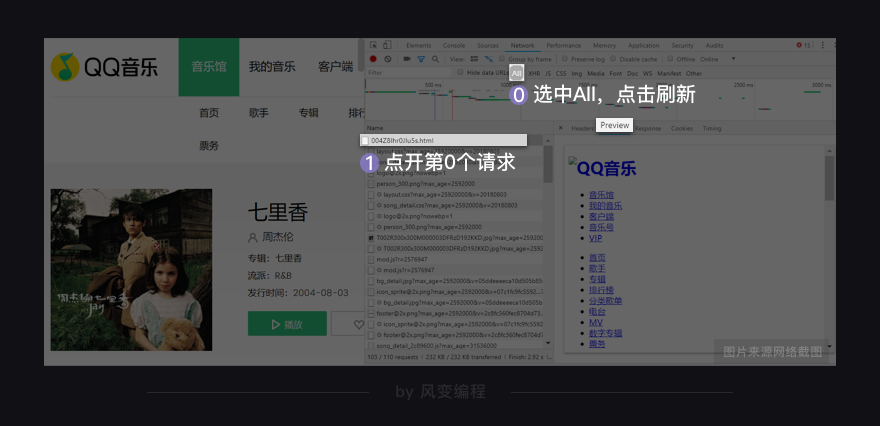
《七里香》的评论

截止当前，我们已经掌握如何爬取歌曲列表。下面，我们要以《七里香》为例，爬取用户评论。

因为你已经掌握这些知识，所以此刻对你来说，爬取评论会显得很轻易。我们来快捷地实现它。首先，进入网址：

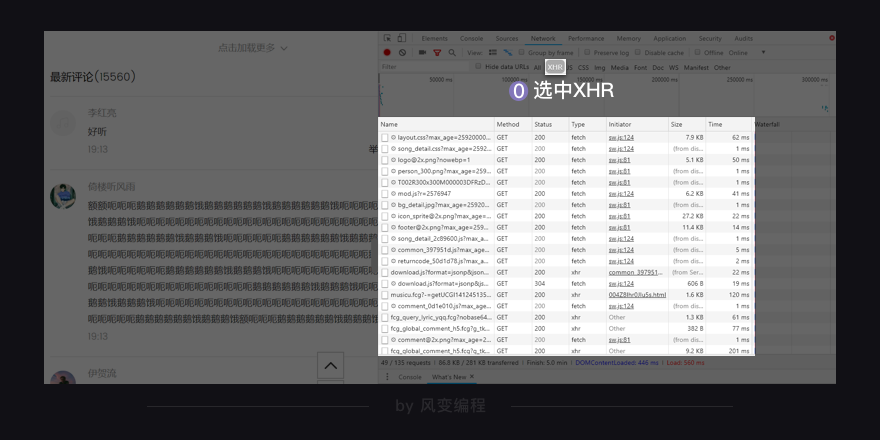
<https://y.qq.com/n/yqq/song/004Z8Ihr0JIu5s.html>

打开Network，选中All，点击刷新。



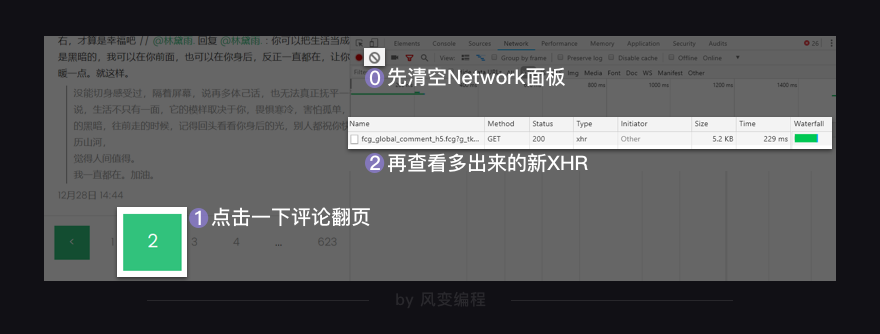
上一关我们说到，第0个请求一般都会是html。我们点开第0个请求来看看（看Preview或Response都可以），里面有没有我们想要的评论信息。

显然是没有的。我们现在去看XHR。

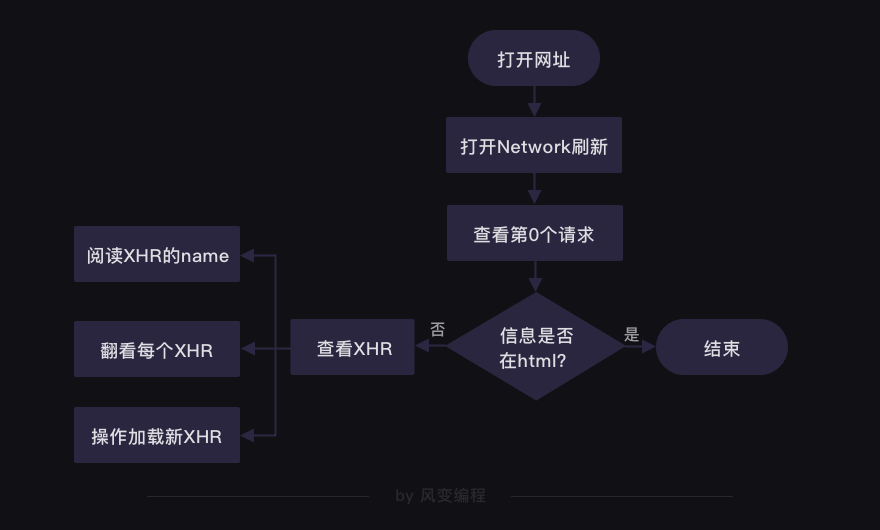


这次的XHR还挺多，有四五十个。常规来说我们有两种方法来寻找XHR：阅读它们的name看看哪个可能是评论；或者是一个一个翻。

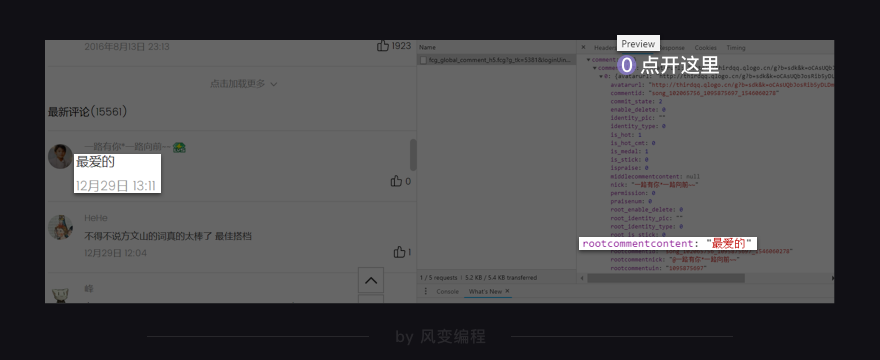
现在给你介绍一个简单的小技巧：先把Network面板清空，再点击一下评论翻页，看看有没有多出来的新XHR，多出来的那一个，就应该是和评论相关的啦。



总结这个技巧，就是：

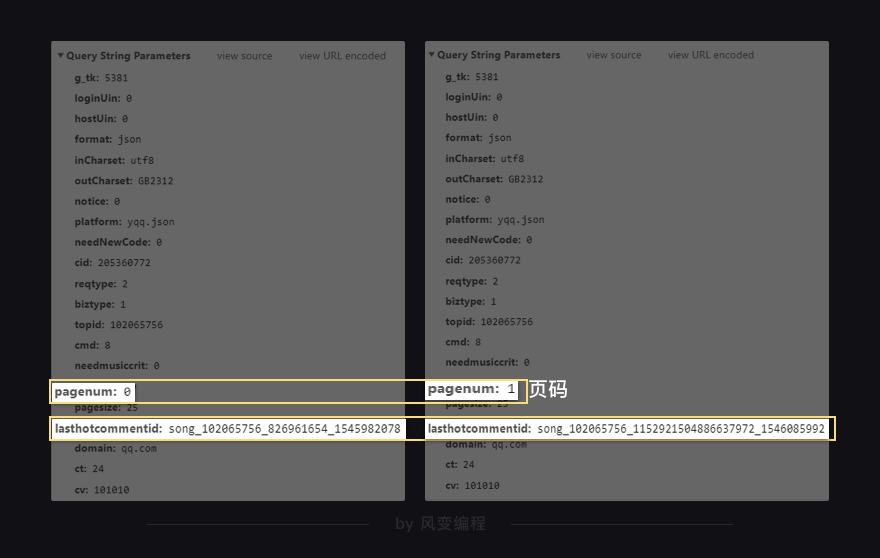


我们点开这个请求的Preview，能够在['comment']['commentlist']里找到评论列表。列表的每一个元素都是字典，字典里键rootcommentcontent对应的值，就是我们要找的评论。



剩下的事情就简单了。我们去模拟这个请求，解析json，提取想要的内容就好。

点击Headers，在General里看链接，在Query String Parametres里看参数，多翻几页评论列表，总结参数的规律。



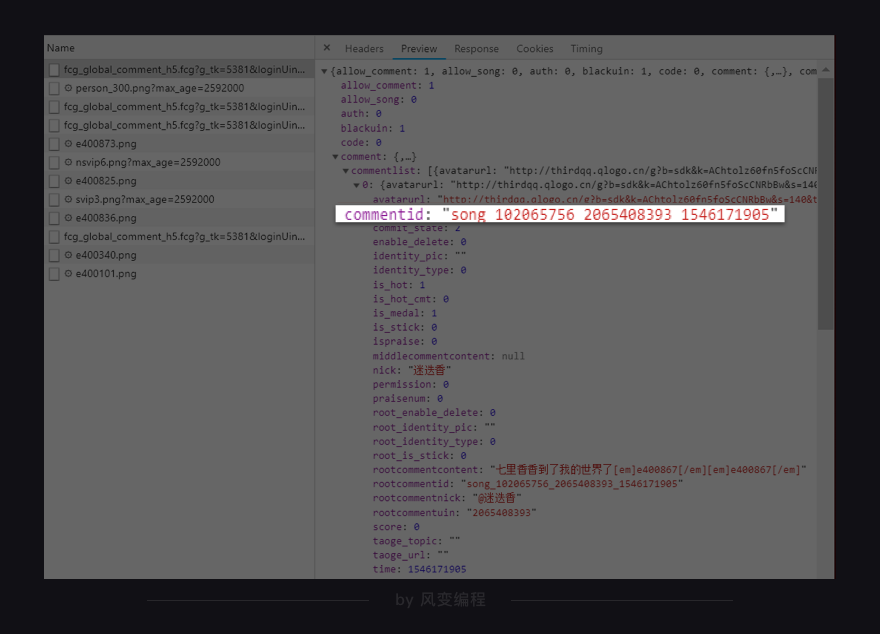
在这里你会遇到一个难点，XHR有两个参数在不断变化：一个是pagenum，一个lasthotcommentid。其中pagenum好理解，就是页码，但是lastcommentid是什么？

我们来阅读这个英文lasthotcommentid，它的含义是：上一条热评的评论id。

基于此，我们可以做一个猜想：每一页的请求，参数lasthotcommentid的值，是上一页的最后一条评论，所对应的id。

我们要验证这个猜想，需要确认：首先，每个评论存在commentid这个东西；其次，是验证这种对应关系。

前者是容易验证的，我们去翻一下就好。如下图：



是的，它存在。现在我们多翻几个XHR，看看它们请求的参数，能否和上一页最后一条评论的id对上号。

当我在写这个关卡时（你看到时候可能评论内已经发生过变化，不见得能对上），事情是这样的：

第2页，请求里lasthotcommentid值是：song\_102065756\_1152921504886637972\_1546085992。

而第1页最后一条评论，它的commentid也是song\_102065756\_1152921504886637972\_1546085992。

第3页，请求里lasthotcommentid值是：song\_102065756\_826961654\_1545982078。

而第2页最后一条评论，它的commentid也是song\_102065756\_826961654\_1545982078。

如此，代码能写了。代码应该如下（同上，不推荐循环超过5次）：

import requests

# 引用requests模块

url = 'https://c.y.qq.com/base/fcgi-bin/fcg\_global\_comment\_h5.fcg'

commentid = ''

# 设置一个初始commentid

for x in range(5):

params = {

'g\_tk':'5381',

'loginUin':'0',

'hostUin':'0',

'format':'json',

'inCharset':'utf8',

'outCharset':'GB2312',

'notice':'0',

'platform':'yqq.json',

'needNewCode':'0',

'cid':'205360772',

'reqtype':'2',

'biztype':'1',

'topid':'102065756',

'cmd':'8',

'needcommentcrit':'0',

'pagenum':str(x),

'pagesize':'25',

'lasthotcommentid':commentid,

'domain':'qq.com',

'ct':'24',

'cv':'101010 '

}

# 将参数封装为字典，其中pagenum和lastcommentid是特殊的变量

res\_comment = requests.get(url,params=params)

# 调用get方法，下载评论列表

json\_comment = res\_comment.json()

# 使用json()方法，将response对象，转为列表/字典

list\_comment = json\_comment['comment']['commentlist']

# 一层一层地取字典，获取评论列表

for comment in list\_comment:

# list\_comment是一个列表，comment是它里面的元素

print(comment['rootcommentcontent'])

# 输出评论

commentid = list\_comment[24]['commentid']

# 将最后一个评论的id赋值给comment，准备开始下一次循环

什么是Request Headers

在我们的案例当中，虽然这个爬虫已经具备了爬取所有评论的能力，但我们依然对它做了限制——只爬取5页的量。

但是如果有一天，你真的需要爬取这一万多条评论，将for循环执行成百上千次。

那么，你最好将自己的爬虫伪装成真实的浏览器——因为在这种情况下，服务器很可能拒绝爬虫访问。甚至有的网站，一开始就不允许爬虫访问。如，知乎、猫眼电影。

那么问题来了，服务器通过什么识别我们是真实浏览器，还是Python爬虫的呢？看这里：



每一个请求，都会有一个Requests Headers，我们把它称作请求头。它里面会有一些关于该请求的基本信息，比如：这个请求是从什么设备什么浏览器上发出？这个请求是从哪个页面跳转而来？

如上图，user-agent会记录你电脑的信息和浏览器版本（如我的，就是windows10的64位操作系统，使用谷歌浏览器）。origin和referer则记录了这个请求，最初的起源是来自哪个页面。它们的区别是referer会比origin携带的信息更多些。

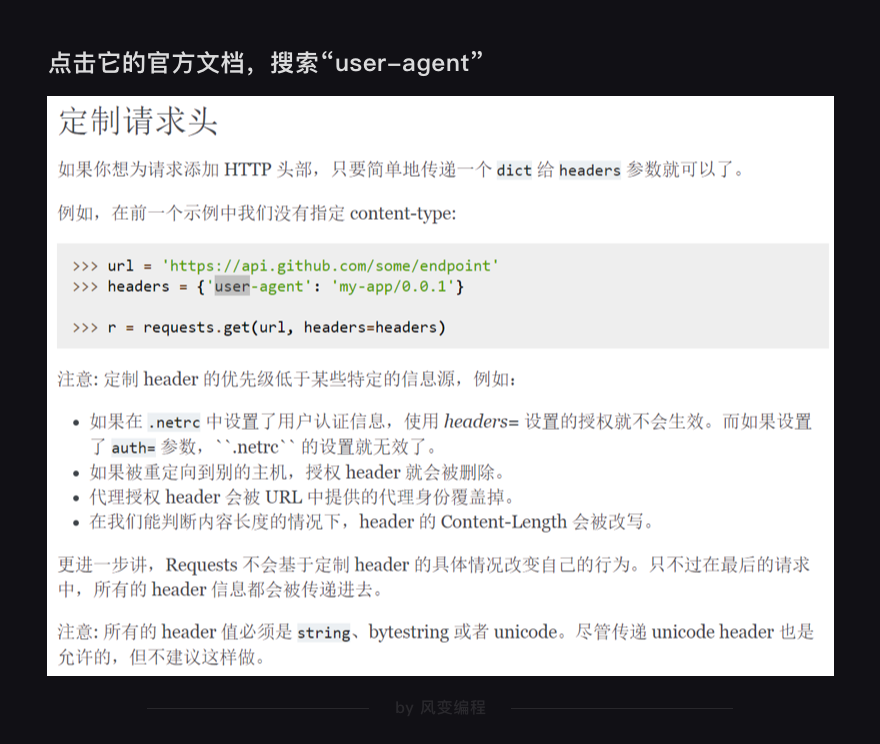
如果我们想告知服务器，我们不是爬虫是一个正常的浏览器，就要去修改user-agent。倘若不修改，那么这里的默认值就会是Python，会被浏览器认出来。

有趣的是，像百度的爬虫，它的user-agent就会是Baiduspider，谷歌的也会是Googlebot……如是种种。

而对于爬取某些特定信息，也要求你注明请求的来源，即origin或referer的内容。比如我有试过，在爬取歌曲详情页里的歌词时，就需要注明这个信息，否则会拿不到歌词。你可以在写练习的时候进行尝试。

如何添加Requests Headers

Requests模块允许我们去修改Headers的值。点击它的官方文档，搜索“user-agent”，你会看到：



如上，只需要封装一个字典就好了。和写params非常相像。

而修改origin或referer也和此类似，一并作为字典写入headers就好。就像这样：

import requests

url = 'https://c.y.qq.com/base/fcgi-bin/fcg\_global\_comment\_h5.fcg'

# 这是那个，请求歌曲评论的url

headers = {

'origin':'https://y.qq.com',

# 请求来源，本案例中其实是不需要加这个参数的，只是为了演示

'referer':'https://y.qq.com/n/yqq/song/004Z8Ihr0JIu5s.html',

# 请求来源，携带的信息比“origin”更丰富，本案例中其实是不需要加这个参数的，只是为了演示

'user-agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',

# 标记了请求从什么设备，什么浏览器上发出

}

params = {

'g\_tk':'5381',

'loginUin':'0',

'hostUin':'0',

'format':'json',

'inCharset':'utf8',

'outCharset':'GB2312',

'notice':'0',

'platform':'yqq.json',

'needNewCode':'0',

'cid':'205360772',

'reqtype':'2',

'biztype':'1',

'topid':'102065756',

'cmd':'8',

'needcommentcrit':'0',

'pagenum':0,

'pagesize':'25',

'lasthotcommentid':'',

'domain':'qq.com',

'ct':'24',

'cv':'101010 '

}

res\_music = requests.get(url,headers=headers,params=params)

# 发起请求

鉴于这个代码你已经写过也运行过，所以在这里我们不再做练习。 只需要记得Request Headers（请求头）的含义和用法就好。

复习

在本关卡，我们主要学习了带参数请求数据和Request Headers的用法。

对于前者，我们认识到一个url由两部分组成，?之前是我们请求的地址，?之后是我们的请求所附带的参数。通常，我们会把参数封装成一个字典，添加进请求中去。

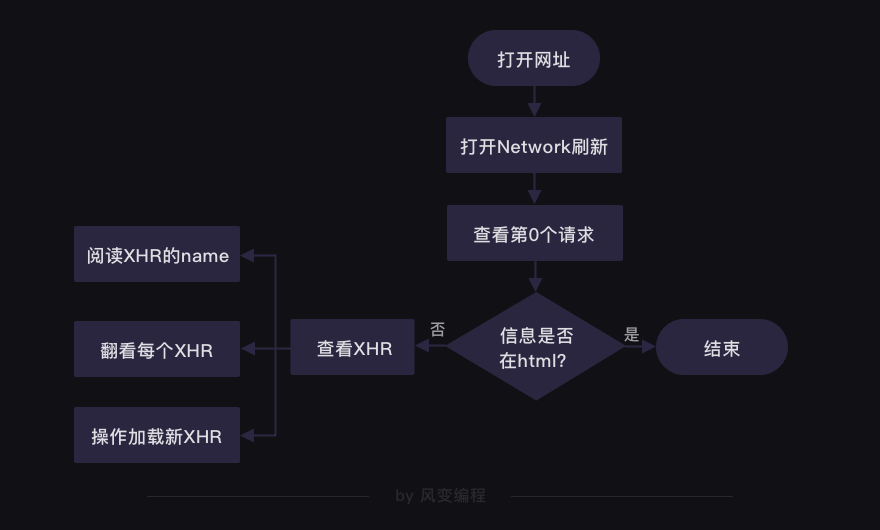
通过对参数进行修改，我们能爬到许多信息。

大约在第8关，我们还会学习另一种请求的类型post，它区别于我们今天所使用的get请求，但用法却是大同小异。

而对于后者，Requests Headers，我们把它称作请求头。它里面会有一些关于该请求的基本信息，比如：这个请求是从什么设备什么浏览器上发出？这个请求是从哪个页面跳转而来？

它最大的应用是帮助我们应对“反爬虫”技术，将Python爬虫伪装成真正的浏览器，不为服务器所辨识；同时也可以标记这个请求的来源是什么，最终帮助我们拿到想要的信息。

除此之外，我们还通过项目实操，学会如何判断我们想要的信息是在Html，还是在XHR里：



利用这张图来分析网页，你能找到几乎所有你想要的信息。剩下的，只是请求、解析、提取、存储罢了。其中解析和提取你已经学过，存储我们将在下一关学习。在未来，我们会学习更多的请求方式，更快的请求方式。

最后用一个代码复习本关卡所学，就是：

import requests

url = 'https://c.y.qq.com/base/fcgi-bin/fcg\_global\_comment\_h5.fcg'

# 这是那个，请求歌曲评论的url

headers = {

'origin':'https://y.qq.com',

# 请求来源，本案例中其实是不需要加这个参数的，只是为了演示

'referer':'https://y.qq.com/n/yqq/song/004Z8Ihr0JIu5s.html',

# 请求来源，携带的信息比“origin”更丰富，本案例中其实是不需要加这个参数的，只是为了演示

'user-agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36',

# 标记了请求从什么设备，什么浏览器上发出

}

params = {

'g\_tk':'5381',

'loginUin':'0',

'hostUin':'0',

'format':'json',

'inCharset':'utf8',

'outCharset':'GB2312',

'notice':'0',

'platform':'yqq.json',

'needNewCode':'0',

'cid':'205360772',

'reqtype':'2',

'biztype':'1',

'topid':'102065756',

'cmd':'8',

'needcommentcrit':'0',

'pagenum':0,

'pagesize':'25',

'lasthotcommentid':'',

'domain':'qq.com',

'ct':'24',

'cv':'101010 '

}

res\_music = requests.get(url,headers=headers,params=params)

# 发起请求