**爬虫04**

复习

在上一关，我们使用两种方式，爬取了热门菜谱清单，内含：菜名、原材料、详细烹饪流程的URL。代码如下：

import requests

# 引用requests库

from bs4 import BeautifulSoup

# 引用BeautifulSoup库

res\_foods = requests.get('http://www.xiachufang.com/explore/')

# 获取数据

bs\_foods = BeautifulSoup(res\_foods.text,'html.parser')

# 解析数据

tag\_name = bs\_foods.find\_all('p',class\_='name')

# 查找包含菜名和URL的<p>标签

tag\_ingredients = bs\_foods.find\_all('p',class\_='ing ellipsis')

# 查找包含食材的<p>标签

list\_all = []

# 创建一个空列表，用于存储信息

for x in range(len(tag\_name)):

# 启动一个循环，次数等于菜名的数量

list\_food = [tag\_name[x].text[18:-14],tag\_name[x].find('a')['href'],tag\_ingredients[x].text[1:-1]]

# 提取信息，封装为列表。注意此处[18:-14]切片和之前不同，是因为此处使用的是<p>标签，而之前是<a>

list\_all.append(list\_food)

# 将信息添加进list\_all

print(list\_all)

# 打印

# 以下是另外一种解法

list\_foods = bs\_foods.find\_all('div',class\_='info pure-u')

# 查找最小父级标签

list\_all = []

# 创建一个空列表，用于存储信息

for food in list\_foods:

tag\_a = food.find('a')

# 提取第0个父级标签中的<a>标签

name = tag\_a.text[17:-13]

# 菜名，使用[17:-13]切掉了多余的信息

URL = 'http://www.xiachufang.com'+tag\_a['href']

# 获取URL

tag\_p = food.find('p',class\_='ing ellipsis')

# 提取第0个父级标签中的<p>标签

ingredients = tag\_p.text[1:-1]

# 食材，使用[1:-1]切掉了多余的信息

list\_all.append([name,URL,ingredients])

# 将菜名、URL、食材，封装为列表，添加进list\_all

print(list\_all)

# 打印

将想要的数据分别提取，再做组合是一种不错的思路。但是，如果数据的数量对不上，就会让事情比较棘手。比如，在我们的案例里，如果一个菜有多个做法，其数量也没规律，那么菜名和URL的数量就会对不上。

寻找最小共同父级标签是一种很常见的提取数据思路，它能有效规避这个问题。但有时候，可能需要你反复操作，提取数据。

所以在实际项目实操中，需要根据情况，灵活选择，灵活组合。我们本关卡所做的项目，只是刚刚好两种方式都可以爬取。

text获取到的是该标签内的纯文本信息，即便是在它的子标签内，也能拿得到。但提取属性的值，只能提取该标签本身的。

from bs4 import BeautifulSoup

bs = BeautifulSoup('<p><a>惟有痴情难学佛</a>独无媚骨不如人</p>','html.parser')

tag = bs.find('p')

print(tag.text)

在爬虫实践当中，其实常常会因为标签选取不当，或者网页本身的编写没做好板块区分，你可能会多提取到出一些奇怪的东西。

当遇到这种糟糕的情况，一般有两种处理方案：数量太多而无规律，我们会换个标签提取；数量不多而有规律，我们会对提取的结果进行筛选——只要列表中的若干个元素就好。

项目：寻找周杰伦

就像标题里描述的那样，这是一个和周杰伦相关的关卡。我还记得自己年少时，沉迷于收集他的专辑、歌单，生怕有缺漏……在当时，互联网不像今天这样普及，做这事可一点都不容易——你必须和小镇上卖CD的老板，非常熟稔才行。

但在今天，我能借助爬虫非常轻松地满足自己的收藏癖。接下来，我也会教给你怎么去做。这就是本关项目：寻找周杰伦，爬取周杰伦的歌曲清单。

我们会尝试用前几关的知识，去完成这个项目。很快，你会发现事情仿佛不是那样简单。你需要一些新工具的帮助，它们的名字叫Network，XHR，json。稍后，我会为你一一介绍。

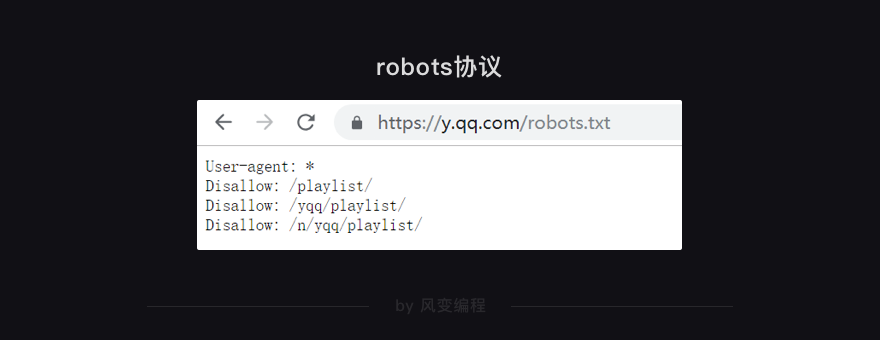
分析过程

当接手一个新项目，开发人员们并不会一上来就去写代码，他们会先去思考这个项目应当如何实现。我们，也是如此。

如果说我们是要爬取周杰伦的歌，那么首先要思考的是：哪家网站，拥有周杰伦的歌曲版权？

获取这个问题答案的方法有两种：其一是自己上网搜，其二是听我这个资深乐迷讲——答案是QQ音乐。

请你务必新建一个浏览器标签，跟随我操作。首先，我们先去QQ音乐的官网，看看它的robots协议[https://y.qq.com/robots.txt](https://y.qq.com/robots.txt" \t "_blank)，结果应如下：



从robots协议看，只是禁止了playlist相关的信息爬取，问题不大，放心去吧！我们来进入QQ音乐的官网首页：[https://y.qq.com](https://y.qq.com/)



接着，在上图的搜索框内输入“周杰伦”，然后点击回车。此时，页面会发生跳转，结果如下图所示：



你能看到，我们想要的歌曲信息，就在这个页面里。这个页面，它的网址会是：

<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>

剩下的事情就简单了，根据我们已经学过的知识，我们可以借助requests和BeautifulSoup，来爬取想要的数据。它的过程，大概会是这样：

根据爬虫四步，我们会利用requests.get()去请求该网址；使用BeautiSoup对请求结果进行解析；利用find\_all方法拿到我们想要的标签；提取歌曲清单。

现在，我们可以尝试写代码。

代码实现

根据前两关所学的知识，如果不出意外，我们的代码大概可以写成这幅模样：

mport requests

from bs4 import BeautifulSoup

res\_music = requests.get('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')

# 请求html，得到response

bs\_music = BeautifulSoup(res\_music.text,'html.parser')

# 解析html

list\_music = bs\_music.find\_all('a',class\_='js\_song')

# 查找class属性值为“js\_song”的a标签，得到一个由标签组成的列表

for music in list\_music:

# 对查找的结果执行循环

print(music['title'])

# 打印出我们想要的音乐名

看上去仿佛没什么问题，但其实这个代码是没办法工作的。你可以先试试看，我再为你解释原因：

程序运行的结果，是什么都找不到……当我们写代码遇到这种情况，我们首先要确认自己的代码是否有问题。

我们可以从下往上，倒推着一步一步排查：看提取是不是出错，看解析是不是出错，看请求是不是出错。现在，我们先去print(list\_music)看看它里面的值。请点击运行下方代码：

import requests

from bs4 import BeautifulSoup

res\_music = requests.get('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')

# 请求html，得到response

bs\_music = BeautifulSoup(res\_music.text,'html.parser')

# 解析html

list\_music = bs\_music.find\_all('a',class\_='js\_song')

# 查找class属性值为“js\_song”的a标签，得到一个由标签组成的列表

print(list\_music)

# 打印它

list\_music，空无一物，它是一个空列表。解析不太可能出问题，因为就一行代码而且符合规范。难道说请求本身就错误了，网页源代码中，根本没有我们要找的歌曲名？我们来print(res\_music)。

import requests

from bs4 import BeautifulSoup

res\_music = requests.get('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')

# 请求html，得到response

print(res\_music.text)

# 打印它

认真翻找它，果然！网页源代码里根本没有我们想要的歌曲清单。

事已至此，已经验证不是代码本身的问题，但目标却未能得到实现。我们就得往前回滚一步：思考，是不是上一步的分析出了问题？

重新分析过程

网页源代码里没有我们想要的数据，那它究竟藏到了哪里呢？

想找到答案，需要用到一项新技能——翻找Network！下面，我来一步步带你做

什么是Network

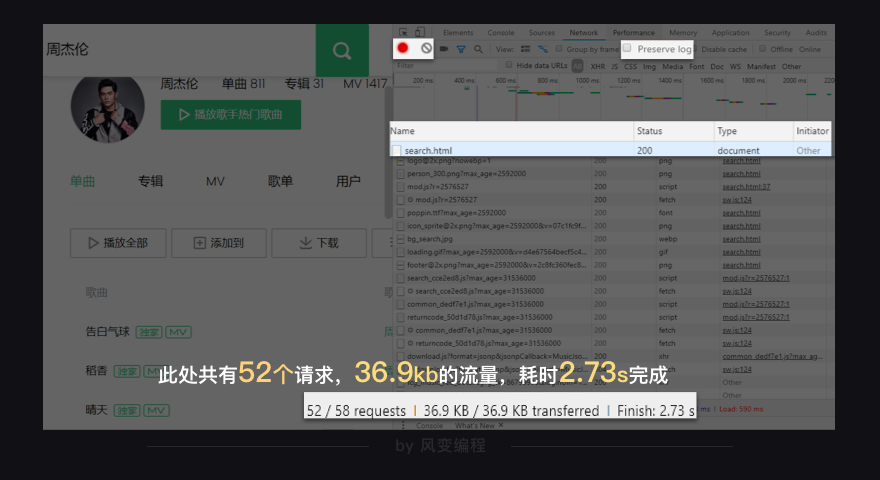
我们先去看看Network的页面。在你刚才打开的QQ音乐页面，调用“检查”（ctrl+shift+i）工具，然后点击Network。



如上图左边框框里的是Elements，我们在那里查看网页源代码。右边框框是我们现在要关注的Network。

Network的功能是：记录在当前页面上发生的所有请求。现在看上去好像空空如也的样子，这是因为Network记录的是实时网络请求。现在网页都已经加载完成，所以不会有东西。

我们点击一下刷新，浏览器会重新访问网络，这样就会有记录。如下图：

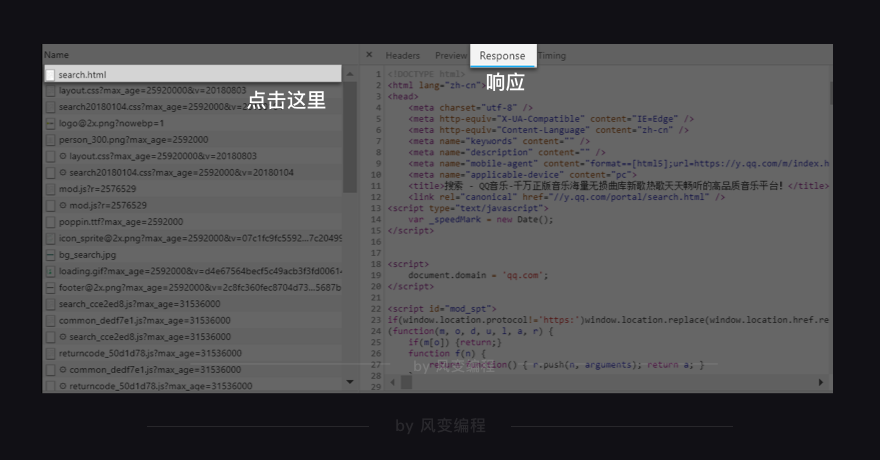


哗~密密麻麻地出来了许多，在图最下面，它告诉我们：此处共有52个请求，36.9kb的流量，耗时2.73s完成。

这个，正是我们的浏览器每时每刻工作的真相：它总是在向服务器，发起各式各样的请求。当这些请求完成，它们会一起组成我们在Elements中看到的网页源代码。

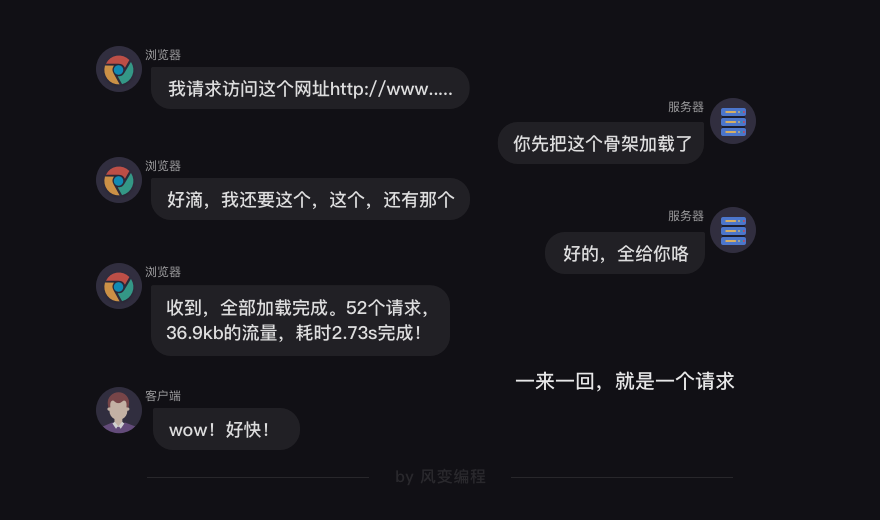
为什么我们刚才没办法拿到歌曲清单呢？答，这是因为我们刚刚写的代码，只是模拟了这52个请求中的一个（准确来说，就是第0个请求），而这个请求里并不包含歌曲清单。

现在请挪动鼠标，找到这个页面的第0个请求：search.html，然后点击它，如下图，我们来查看它的Response（官方翻译叫“响应”，你可以理解为服务器对浏览器这个请求的回应内容，即请求的结果）。



其实，它就是我们刚刚用requests.get()获取到的网页源代码，它里面不包含歌曲清单。

一般来说，都是这种第0个请求先启动了，其他的请求才会关联启动，一点点地将网页给填充起来。做一个比喻，第0个请求就好比是人的骨架，确定了这个网页的结构。在此之后，众多的请求接连涌入，作为人的血脉经络。如此，人就变好看。

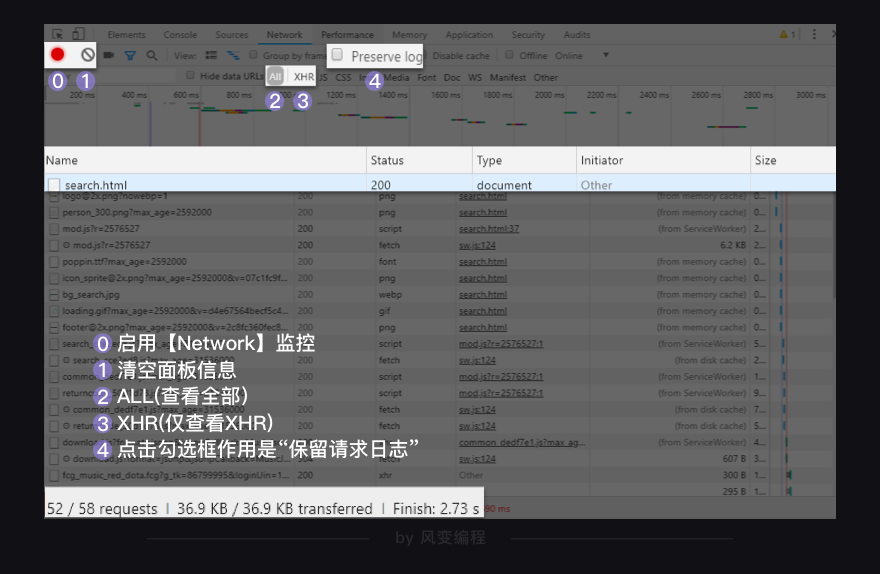


当然啦，也有一些网页，直接把所有的关键信息都放在第0个请求里，尤其是一些比较老（或比较轻量）的网站，我们用requests和BeautifulSoup就能解决它们。比如我们体验过的“这个书苑不太冷”，比如你看过的“人人都是蜘蛛侠”博客，比如豆瓣。

总之，为了成功抓取到歌曲清单。我们得先找到，歌名藏在哪一个请求当中。再用requests库，去模拟这个请求。

Network怎么用

想做这个，我们需要先去了解下Network面板怎么用。回头看我们之前给的图：



从上往下，只看我圈起来的内容的话，它有四行信息。下面，我来为你介绍它。

第0行的左侧，红色的圆钮是启用Network监控（默认高亮打开），灰色圆圈是清空面板上的信息。右侧勾选框Preserve log，它的作用是“保留请求日志”。如果不点击这个，当发生页面跳转的时候，记录就会被清空。所以，我们在爬取一些会发生跳转的网页时，会点亮它。

第1行，是对请求进行分类查看。我们最常用的是：ALL（查看全部）/XHR（仅查看XHR，我们等会重点讲它）/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。



夹在第2行和第1行中间的，是一个时间轴。记录什么时间，有哪些请求。而第2行，就是各个请求，你可以看下面这张表来理解（读，但不需要记忆）。



在第3行，我们讲过了，是个统计：有多少个请求，一共多大，花了多长时间。

什么是XHR？

在Network中，有一类非常重要的请求叫做XHR（当你把鼠标在XHR上悬停，你可以看到它的完整表述是XHR and Fetch），未来我们几乎每一关都要和它打交道。下面，我来为你重点介绍它。

我们平时使用浏览器上网的时候，经常有这样的情况：浏览器上方，它所访问的网址没变，但是网页里却新加了内容。

典型代表：如购物网站，下滑自动加载出更多商品。在线翻译网站，输入中文实时变英文。比如，你正在使用的教学系统，每点击一次Enter就有新的内容弹出。

再比如，我们正在爬取的QQ音乐案例，如果你对“周杰伦”的搜索结果进行翻页，浏览器上方显示的网址，也不会发生变化。对此，你可以试试看。

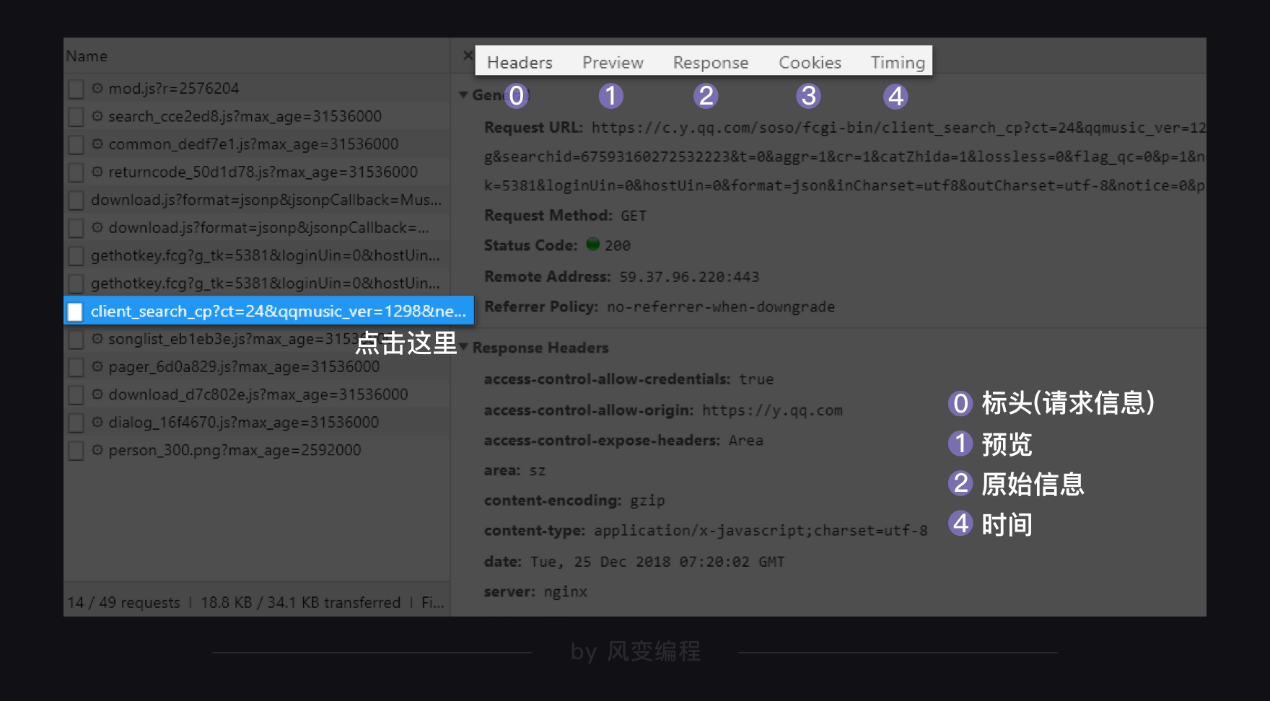
这个，叫做Ajax技术（技术本身和爬虫关系不大，在此不做展开，你可以通过搜索了解）。应用这种技术，好处是显而易见的——更新网页内容，而不用重新加载整个网页。又省流量又省时间的，何乐而不为。

如今，比较新潮的网站都在使用这种技术来实现数据传输。只剩下一些特别老，或是特别轻量的网站，还在用老办法——加载新的内容，必须要跳转一个新网址。

这种技术在工作的时候，会创建一个XHR（或是Fetch）对象，然后利用XHR对象来实现，服务器和浏览器之间传输数据。在这里，XHR和Fetch并没有本质区别，只是Fetch出现得比XHR更晚一些，所以对一些开发人员来说会更好用，但作用都是一样的。

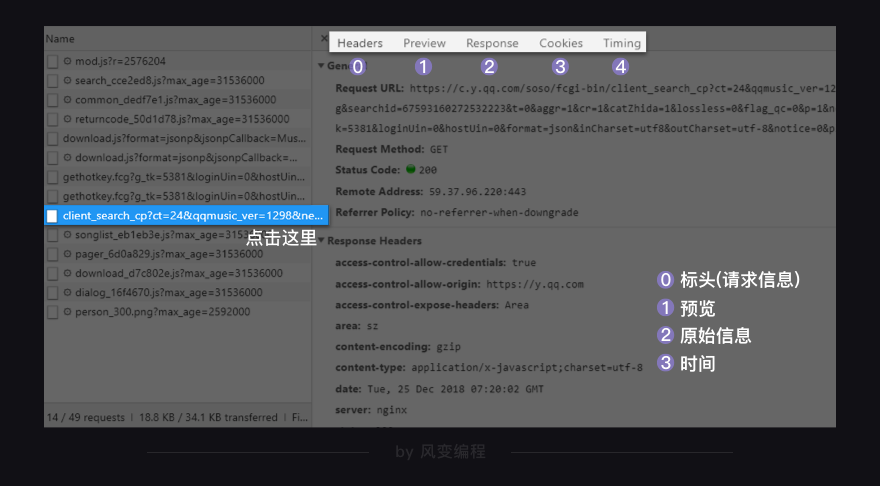
XHR怎么请求？

显而易见，对照前面的表单。我们的歌曲清单不在网页源代码里，而且也不是图片，不是媒体文件，自然只会是在XHR里。我们现在去找找看，点击XHR按钮。



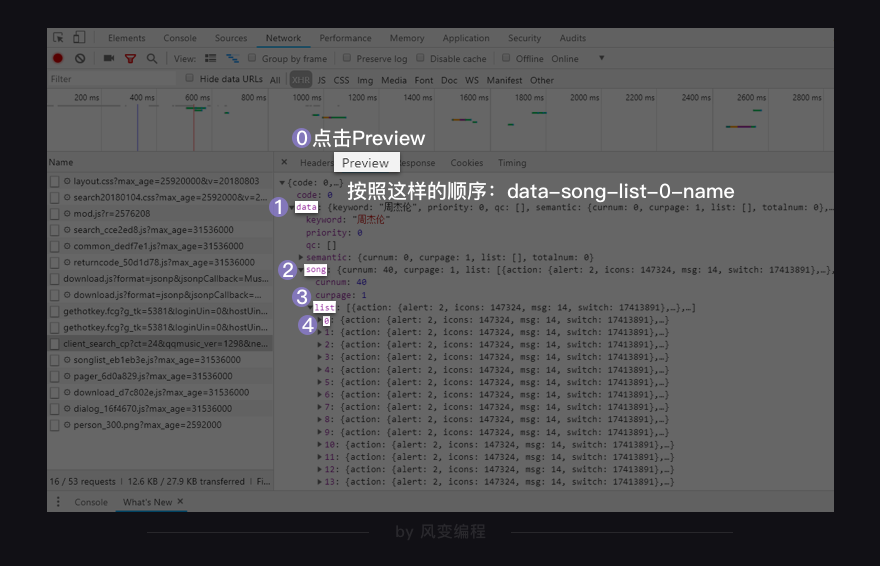
这个网页里一共有10个XHR或Fetch，我们要从里面找出带有歌单的那一个。

笨办法当然是一个一个实验，但聪明的办法是去尝试阅读它们的名字。比如你一眼就看到：client\_search（客户端搜索）……而且它最大，有10.9KB，我们来点击它。

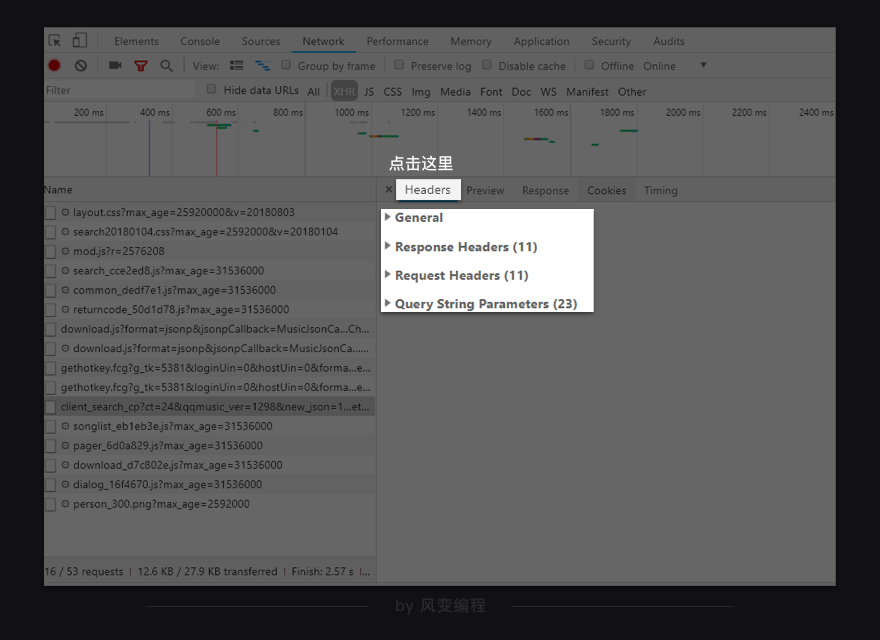


出现了如上图这样的一个窗口，我们先来看蓝框里面的内容，从左往右分别是：Headers：标头（请求信息）、Preview：预览、Response：原始信息、Timing：时间。

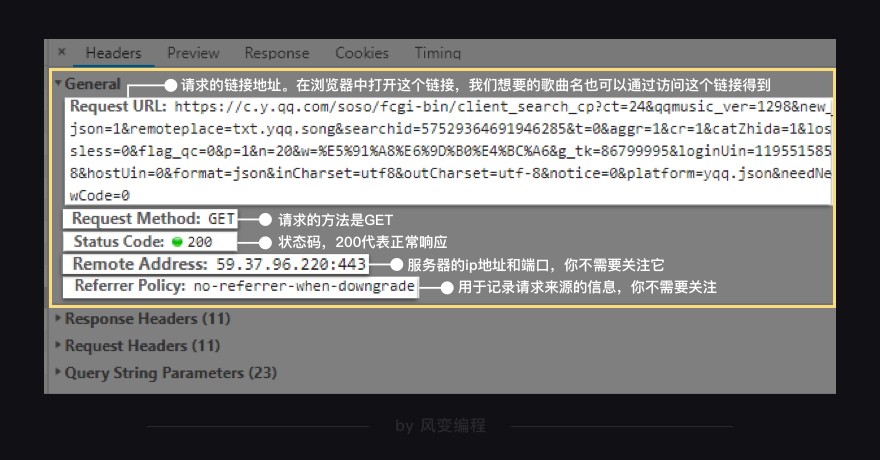
点击Preview，你能在里面发现我们想要的信息：歌名就藏在里面！（只是有点难找，需要你一层一层展开：data-song-list-0-name，然后就能看到“告白气球”）



那如何把这些歌曲名拿到呢？这就需要我们去看看最左侧的Headers，点击它。如下所示，它被分为四个板块。



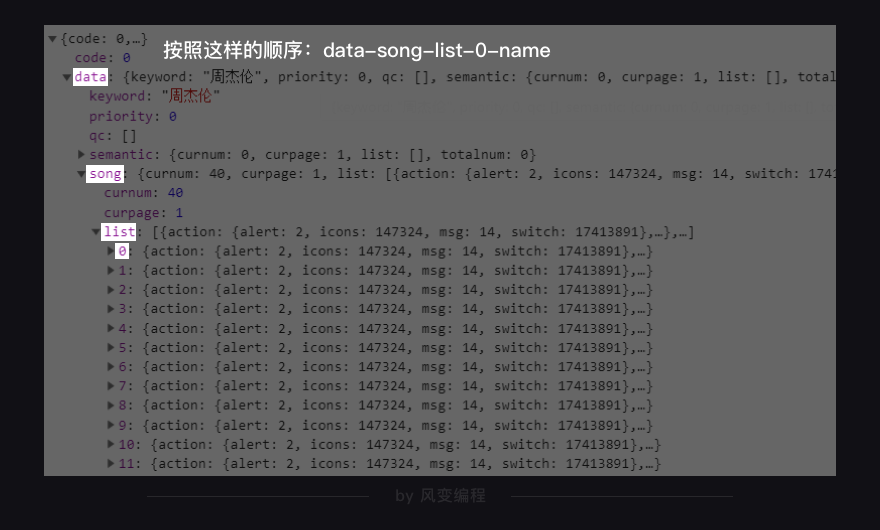
我们把后面的三个，留待后续关卡详细解释。今天，你只是看看它们就好，然后将注意力放在第0个General上面。点开它，你会看到：



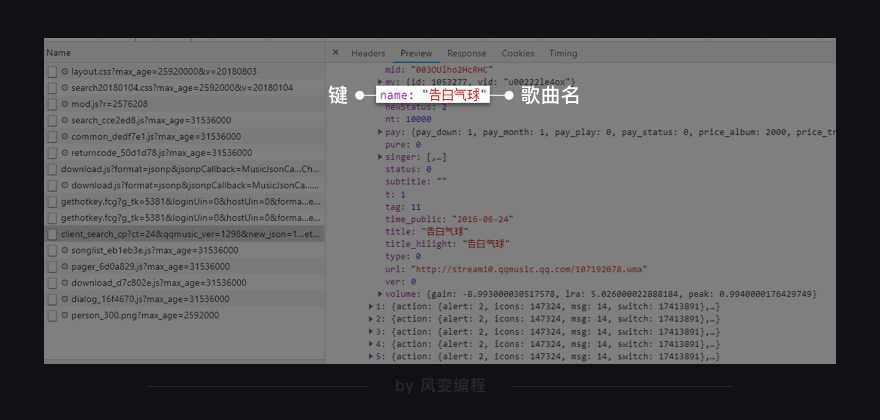
看到了吗？General里的Requests URL就是我们应该去访问的链接。如果在浏览器中打开这个链接，你会看到一个让人绝望的结构：最外层是一个字典，然后里面又是字典，往里面又有列表和字典……



它就和你在Response里看到的一个样。还是放弃挣扎吧，回到原网址，直接用Preview来看就好。列表和字典在此都会有非常清晰的结构，层层展开。



如上，我们一层一层地点开，按照这样的顺序：data-song-list-0-name，看到：



歌曲名就在这里，它的键是name。理解这句话：这个XHR是一个字典，键data对应的值也是一个字典；在该字典里，键song对应的值也是一个字典；在该字典里，键list对应的值是一个列表；在该列表里，一共有20个元素；每一个元素都是一个字典；在每个字典里，键name的值，对应的是歌曲名。

此刻的你有了一个大胆的想法：利用requests.get()访问这个链接，把这个字典下载到本地。然后去一层一层地读取，拿到歌曲名。

到此，我们的代码可以写成这样，你可以尝试运行看看：

import requests

# 引用requests库

res = 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

print(res.text)

# 把它打印出来

在这里，我们又遇到一个障碍：使用res.text取到的，是字符串。它不是我们想要的列表/字典，数据取不出来。老虎吃天，没处下嘴。

json是什么？

或许你会问：吴枫老师，我们已经学过如何把response对象转成字符串，那有没有什么属性或者方法，能把response对象转成列表/字典呢？

办法自然有，但我要先讲给你一个新的知识点——**json**。

json是什么呢？粗暴地来解释，在Python语言当中，json是一种特殊的字符串，这种字符串特殊在它的写法——它是用列表/字典的语法写成的。

a = '1,2,3,4'

# 这是字符串

b = [1,2,3,4]

# 这是列表

c = '[1,2,3,4]'

# 这是字符串，但它是用json格式写的字符串

这种特殊的写法决定了，json能够有组织地存储信息。

我们在生活当中，总是在接触林林总总的数据。如果它们直接以堆砌的形式出现在你面前，你很难阅读它。比如：想象一个乱序排布的字典，一个堆满文件的电脑桌面，一本不分段落章节的小说……

数据需要被有规律地组织起来，我们才能去查找、阅读、分析、理解。比如：汉语字典应该按照拼音排序，文件应该按照一定规律放进不同的文件夹，小说要有章节目录——大标题、中标题、小标题。

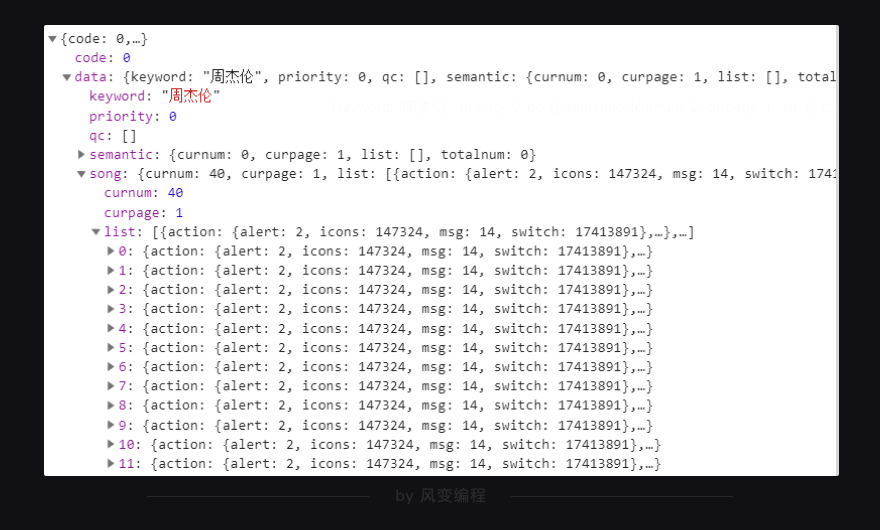
可以发现，组织数据的方式也有规律，规律有三条：



一般来说，这三条占得越多，数据的结构越清晰；占得越少，数据的结构越混沌。

生活如此，网络之间的数据传输也是如此。在之前，我们已经学习过html，它通过标签、属性来实现分层和对应。

json则是另一种组织数据的格式，长得和Python中的列表/字典非常相像。它和html一样，常用来做网络数据传输。刚刚我们在XHR里查看到的列表/字典，严格来说其实它不是列表/字典，它是json。



或许你会有疑问：那直接写成列表/字典不就好了，为什么要把它表示成字符串？答案很简单，因为不是所有的编程语言都能读懂Python里的数据类型（如，列表/字符串），但是所有的编程语言，都支持文本（比如在Python中，用字符串这种数据类型来表示文本）这种最朴素的数据类型。

如此，json数据才能实现，跨平台，跨语言工作。

而json和XHR之间的关系：XHR用于传输数据，它能传输很多种数据，json是被传输的一种数据格式。就是这样而已。

我们总是可以将json格式的数据，转换成正常的列表/字典，也可以将列表/字典，转换成json。

json数据如何解析？

说回到我们的案例，当我们请求得到了json数据，应该如何读取呢？我们可以在requests库的官方文档中，找到答案。我们打开浏览器，搜索“requests 官方文档”，会来到这个界面：



点开链接，进入文档，你会看到一个非常傲娇的作者。



使用浏览器的ctrl+f功能，在网页内搜索关键词json，能够非常快捷地找到这里：



点击进入，你将看到requests库处理json数据的方法。

方法很简单，请求到数据之后，使用json()方法即可成功读取。接下来的操作，就和列表/字典相一致。



下面来体验一下，运行下方代码：

import requests

# 引用requests库

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')

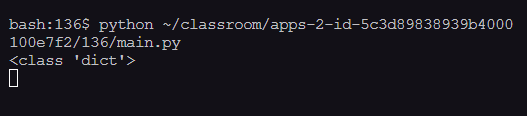
# 调用get方法，下载这个字典

json\_music = res\_music.json()

# 使用json()方法，将response对象，转为列表/字典

print(type(json\_music))

# 打印json\_music的数据类型



实操：完成代码实现

现在，我们至少可以写代码，提取出20个周杰伦的歌曲名。你可以尝试续写这个代码，稍后我会提供参考答案。

import requests

# 引用requests库

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

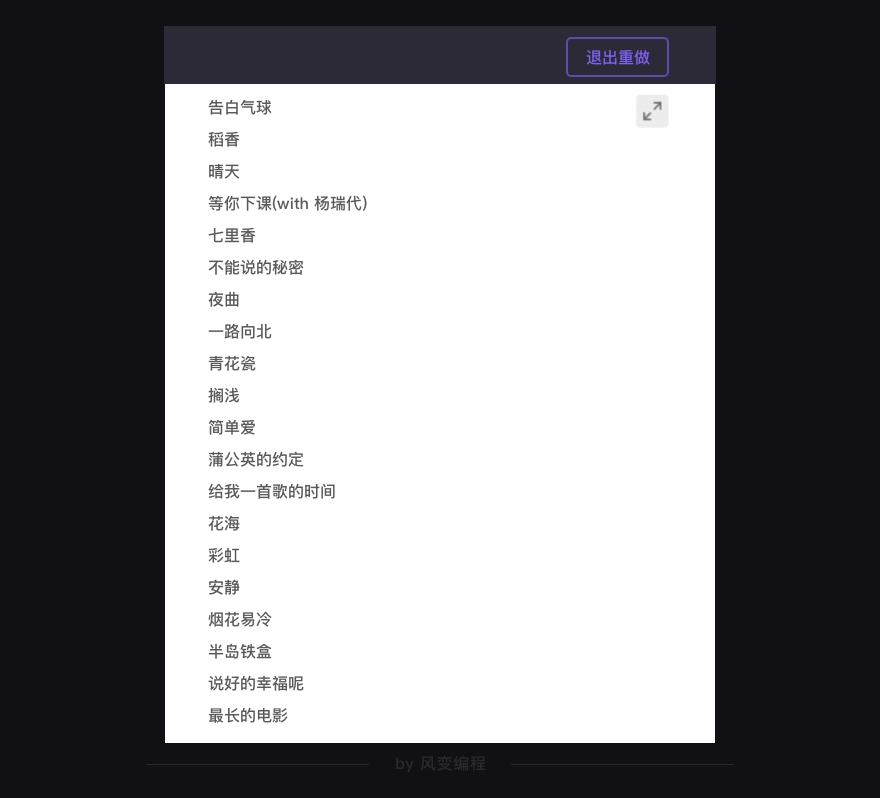
for music in list\_music:

# 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

你应该能看到类似这样的结果（反正我写这个教程的时候是这样，不知道现在会不会变）：



成功！撒花！

就是这样一个代码，它能拿到周杰伦在QQ音乐上，前20个歌曲的名单。

事实上，如果对这个程序稍加延展，它就能拿到：歌曲名、所属专辑、播放时长，以及播放链接。因为这些信息都在那个XHR里，认真观察分析，如果有必要的话还可以配合翻译软件。最终，你可以用同样的方法把它们提取出来。就像这样：

import requests

# 引用requests库

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

一个总结

截至当前，我们已经部分完成了初定目标：爬取周杰伦的歌曲清单。

为什么说部分？一方面是我们只拿到20首歌曲的信息，远不能满足一个狂热粉丝的需要。另一方面，只拿到歌名/专辑/时长……这些数据还不够酷，狂热粉丝还想拿到所有的歌词，甚至还有歌曲的评论。



这会是一个浩荡的工程，因为有相当量的数据要爬取。但拿到这些数据，它就有了数据分析价值：周杰伦的歌最常出现哪些关键词？用户都在评论些什么内容？他们都喜欢在什么时间听？

同理，你可以拿到任何一个歌手的这些信息。如果你是一个音乐行业的从业者，那么它们将对于你产生价值。如果你不是，那么这个爬虫技术，可以帮助你在自己行业创造价值——换自己领域的网站去爬就好。

想拿到这么多数据，你需要学习下一关的知识：狂热粉丝——带参数请求数据。

拿到这么多数据，想要有规律地存储，你要学习第6关的知识：爬到的数据存哪里？——csv&excel文件

这么多的数据，爬起来太慢想要对它进行加速怎么办？你就需要学习11、12关的知识……

如是种种，学无止境，说的就是这样一回事。但事情的最开始，这所有一切的底层原理，一定还是这寥寥几行代码。

截止到这一关，你已经能够看懂绝大多数的网络数据请求组，并且尝试用Python去模拟这些请求，再往后，都是基于此的延伸。

一个复习

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。

而如果你想在Python语言中，实现列表/字典转json，json转列表/字典，则需要借助json模块。json模块不在我们的教学范围之内，所以不做展开。你可阅读它的官方文档来了解，地址在这里：[https://docs.python.org/3/library/json.html](https://docs.python.org/3/library/json.html" \t "_blank)

一个简单的应用示例，是这样：

import json

# 引入json模块

a = [1,2,3,4]

# 创建一个列表a。

b = json.dumps(a)

# 使用dumps()函数，将列表a转换为json格式的字符串，赋值给b。

print(b)

# 打印b。

print(type(b))

# 打印b的数据类型。

c = json.loads(b)

# 使用loads()函数，将json格式的字符串b转为列表，赋值给c。

print(c)

# 打印c。

print(type(c))

# 打印c的数据类型。