爬虫第0-5关知识复习梳理

来啦来啦，同学们在嘛~

首先跟大家讲一下，我们的班群和复习群会在下午进行解散

如果现在没法参与讲解的同学们，助教会将复习课的内容放在毕业大礼包里发给你们。

群解散后，只要不手动将群从微信聊天列表里删除，大家还是可以通过爬楼的方式查看群消息记录的

准备开始爬虫课程的复习啦，复习课主要是知识点梳理，通过这个复习课，帮助大家将爬虫0-5关的课程内容重新巩固一下

有疑问的同学，在讲解结束后，助教会留出时间来给大家提问，希望大家遵守一下提问秩序哈~

山脚

第0关

第0关，我们初识爬虫

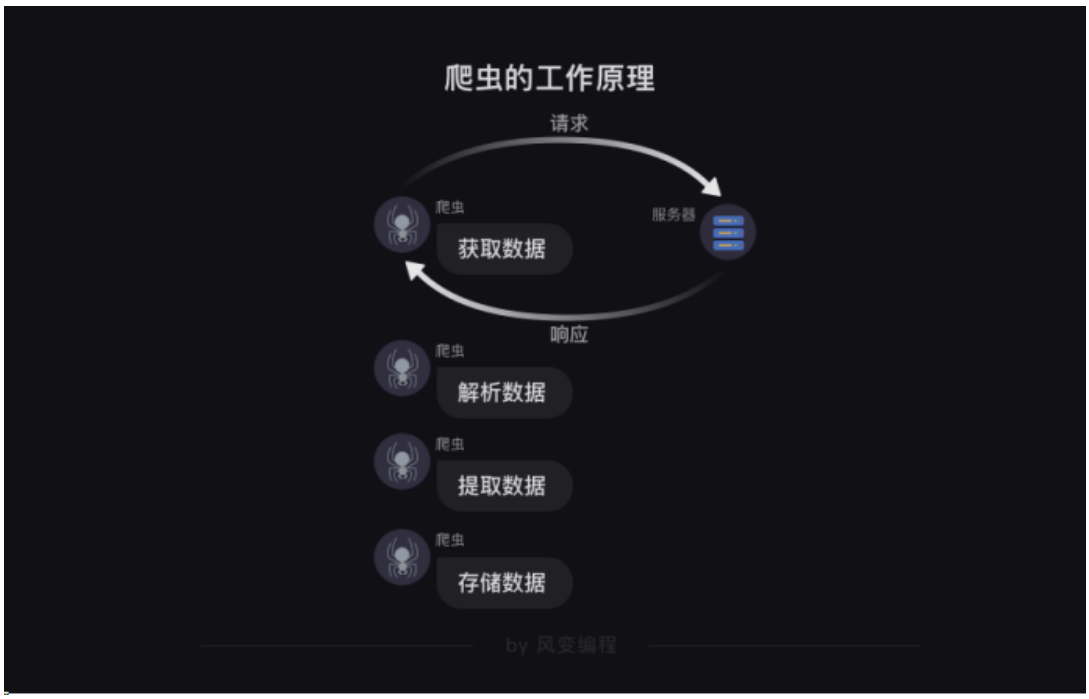
爬虫，从本质上来说，就是利用程序在网上拿到对我们有价值的数据

所以，我们首先要了解浏览器的工作原理，以及爬虫的工作原理

过程是：发出请求——响应请求——解析数据——提取数据——存储数据

因为我们不是学习网络的，所以我们不需要去研究每个过程是怎么实现的

我们大概了解是这么个过程即可



这里，爬虫的原理分为4个步骤，我们做每个爬虫项目都会依照这个步骤来进行

第0步：获取数据。爬虫程序会根据我们提供的网址，向服务器发起请求，然后返回数据。

第1步：解析数据。爬虫程序会把服务器返回的数据解析成我们能读懂的格式。

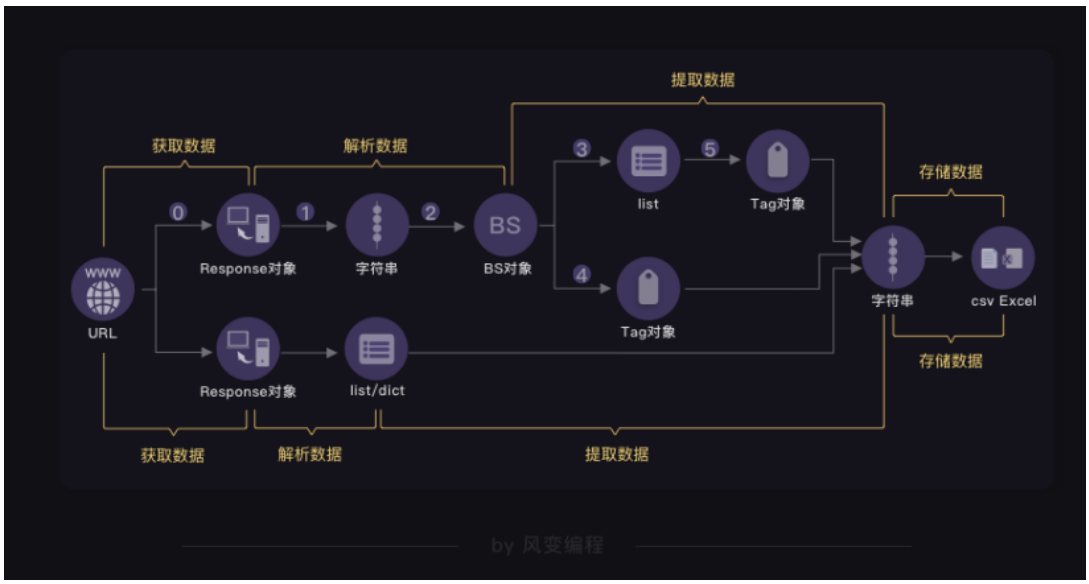
第2步：提取数据。爬虫程序再从中提取出我们需要的数据。

第3步：储存数据。爬虫程序把这些有用的数据保存起来，便于你日后的使用和分析。

请大家牢记住这4个步骤，我们所有的爬虫项目，都是需要套进这个步骤里面去实现

所以我们不需要去硬背代码，要学会的是套用格式

我们把这些步骤，详细点表示出来是这样的



要获取数据，我们需要使用到requests模块

requests 库可以帮我们下载网页源代码、文本、图片，甚至是音频。其实，“下载”本质上是向服务器发送请求并得到响应。

而解析和提取数据，最基本的我们会使用到BeautifulSoup

解析数据，就是把服务器返回来的HTML源代码翻译为能看懂的样子

提取数据，把我们需要的数据从众多数据中挑选出来

但解析和提取数据的前提是，我们需要先学会看网页上的html信息

于是我们来到了第1关的学习

第1关

HTML（Hyper Text Markup Language）是用来描述网页的一种语言，也叫超文本标记语言

我们通常借助开发者工具栏来进行查看：在网页的空白处点击右键，然后选择“检查”（快捷方式是ctrl+shift+i，或者f12）

最简单的html结构是下面这样的

知识一：html的组成

1.架构

```
<html>
    <head>
            <meta charset="utf-8">  #定义了HTML文档的字符编码。
  <title>我是网页的名字</title>
    </head>
    <body>
        <h1>我是一级标题</h1>
        <h2>我是二级标题</h2>
        <h3>我是三级标题</h3>
         <p>我是一个段落啦。一级标题、二级标题和我，我们三个一起组成了body。
          </p>
     </body>
</html>
```

我们需要知道的是，所有<>都是叫做标签，而标签通常是成对出现的

开始标签+中间的所有内容+结束标签，它们在一起就组成了【元素】

常见的html元素有以下这些

常见HTML元素

开始标签	元素内容	结束标签	用法
<h1>	一级标题	</h1>	一级标题
<h2>	二级标题	</h2>	二级标题
	段落文本		段落
<a>	描述链接的文本		超链接
	其它元素或文本		块

by 风变编程

在<>标签里面，我们通常会加上属性，其中最常见的属性有以下几种

标签+属性，对于我们在爬取网页的时候定位至关重要

所以在我们的爬虫项目中，也就是利用网页的标签和属性来定位

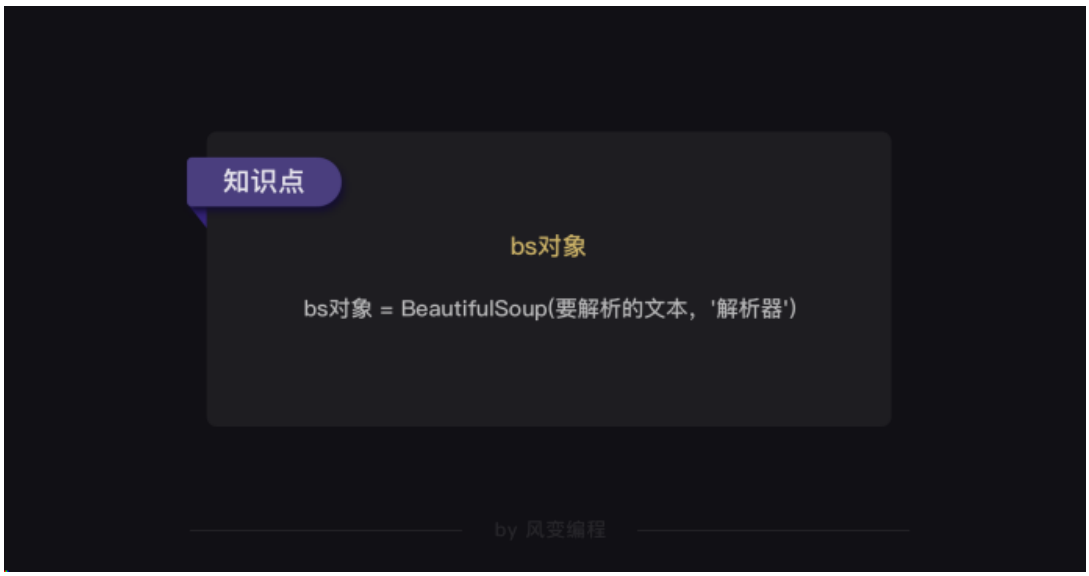
但是我们也不需要去记住html语言到底为什么这么写，到底是什么意思，学会看懂即可

第2关

学会看懂网页代码后，我们在第2关就开始学习用BeautifulSoup了

首先我们要安装模块哦，pip install BeautifulSoup4
（Mac电脑需要输入pip3 install BeautifulSoup4）

对于抓取到的内容，我们先进行解析，格式是这样的



这里需要两个参数

第0个参数是要被解析的文本，必须是字符串

第1个参数用来标识解析器，我们要用的是一个Python内置库：html.parser（它不是唯一的解析器，但是比较简单的）

我们通常写成以下的形式

实例:

```
import requests
from bs4 import BeautifulSoup  #引入BS库

res = requests.get('https://localprod.pandateacher.com/python-manuscript/crawler-html/spider-men5.0.htm
html = res.text
soup = BeautifulSoup(html,'html.parser') #把网页解析为BeautifulSoup对象
```

解析后，我们需要提取出我们要的数据，使用到的方法是find() 和 find_all()

这两者的区别是：
1、find()只提取首个满足要求的数据，而find_all()提取出的是所有满足要求的数据
2、find()提取后返回的是“字符串”类型；
而find_all()提取的是所有满足要求的数据，所以是以“列表”的类型返回的
3、因此，find()后面可以直接接find_all()等提取数据的格式；
而find_all()后面需要把元素提取出来后再接find()等提取数据的格式；

当我们使用find_all()后，在提取数据时，把元素从列表里面提取到的东西，是一个Tag对象

Tag对象可以再进一步提取出内容

至此，我们用一张图看懂爬虫前三步：获取、解析、提取

学会这三步后，我们在第3关进行了一次实操。如果对前面这部分内容还不太理解的同学，可以再多看几遍第3关的实操课哦

第3关

在第3关中，同学们问的最多的就是对提取出来数据的切片处理问题

在第3关【17：-13】这，这里切片切的是字符串，其实和列表的切片也是一样的

我们双击右边的文字

然后会发现前后蓝色部分都是被选中了，因为前后都是空格

接下来，我们可以把鼠标点进去

然后我们可以通过这个光标来数出切片的范围

这是一种最直接的方式，当然同学会说好麻烦啊，有没有其他方法?

有

我们可以通过不断尝试去找到正确的切片范围，不一定要一次到位

也可以把代码导出到一个txt文件里，数<>两侧的空格

再或者通过遍历的方式来数空格

遍历的方法怎么数？4行代码搞定

```
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
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

    temp = 1
    for i in tag_a.text:
        print(temp,i)
        temp+=1
```

当然你也可以用replace(" ","")来去空格；

或者strip函数，name = tag_a.text.strip()，方法有很多种

那么第3关学习完成后，我们已经实现了从一个比较大的网站去爬取我们想要的内容啦

但是这还是皮毛，很多数据不是这么简简单单就可以找到并爬取下来的

第4关

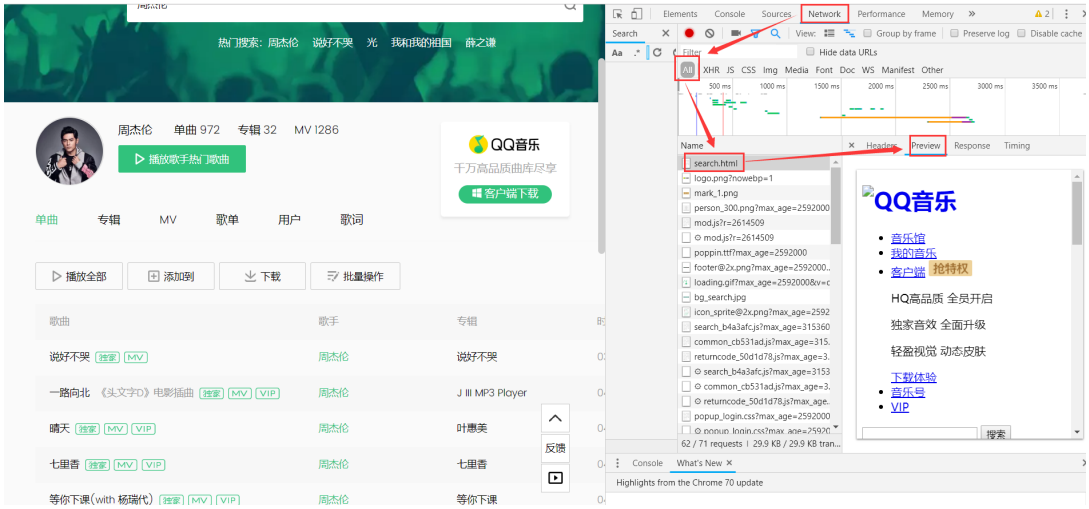
接着我们来到了第4关的学习，在这一关，我们学习的是Network

Network的功能是：记录在当前页面上发生的所有请求

在Network中，含有这些内容

ALL	查看全部
XHR	一种不借助刷新网页即可传输数据的对象
Doc	Document，第0个请求一般在这里
Img	仅查看图片
Media	仅查看媒体文件
Other	其他
JS和CSS	前端代码，负责发起请求和页面实现
Font	字体
WS和Manifest	网络编程相关知识，无需了解

by 风变编程

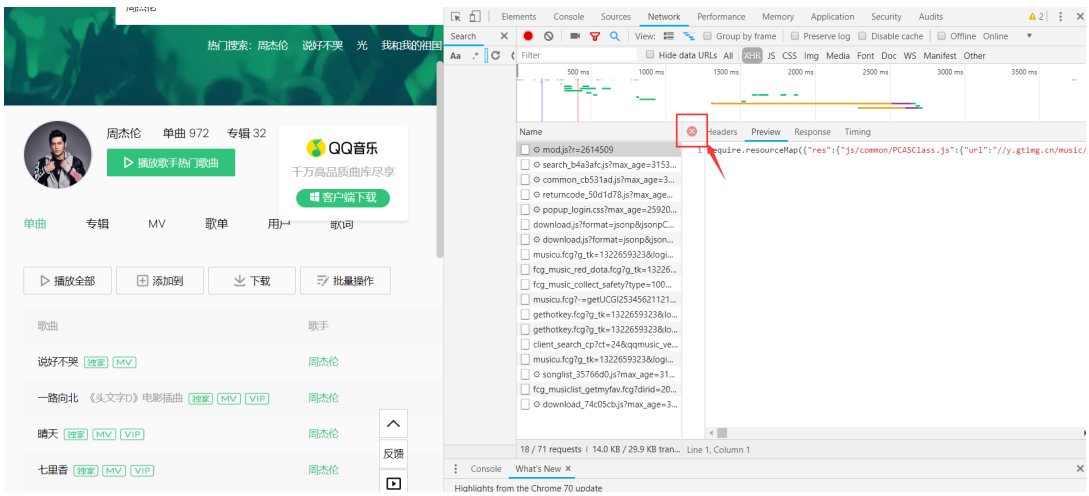


通常，第一个请求为html请求

一般，我会先看一下第一个请求，如果在preview或response中没有我需要的数据，那么说明要爬取的内容需要通过到XHR里面寻找

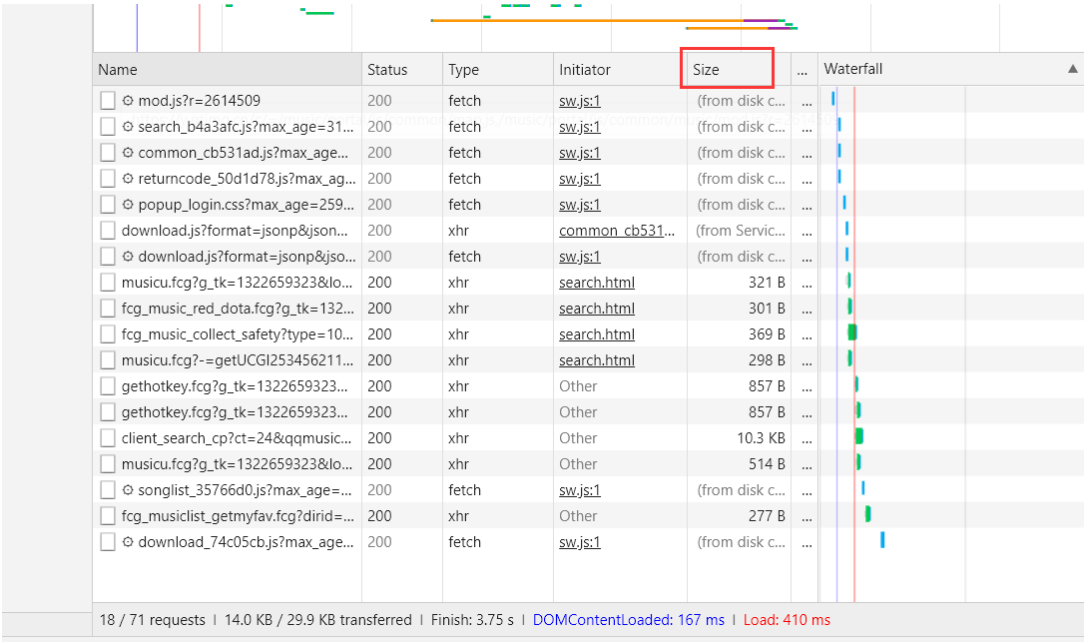
XHR对象，是服务器和浏览器之间传输数据

这里也教同学一种比较快速找到我们需要的XHR的方法



当页面刷新后，所有请求信息加载完毕，我们可以点这个X，把页面关掉

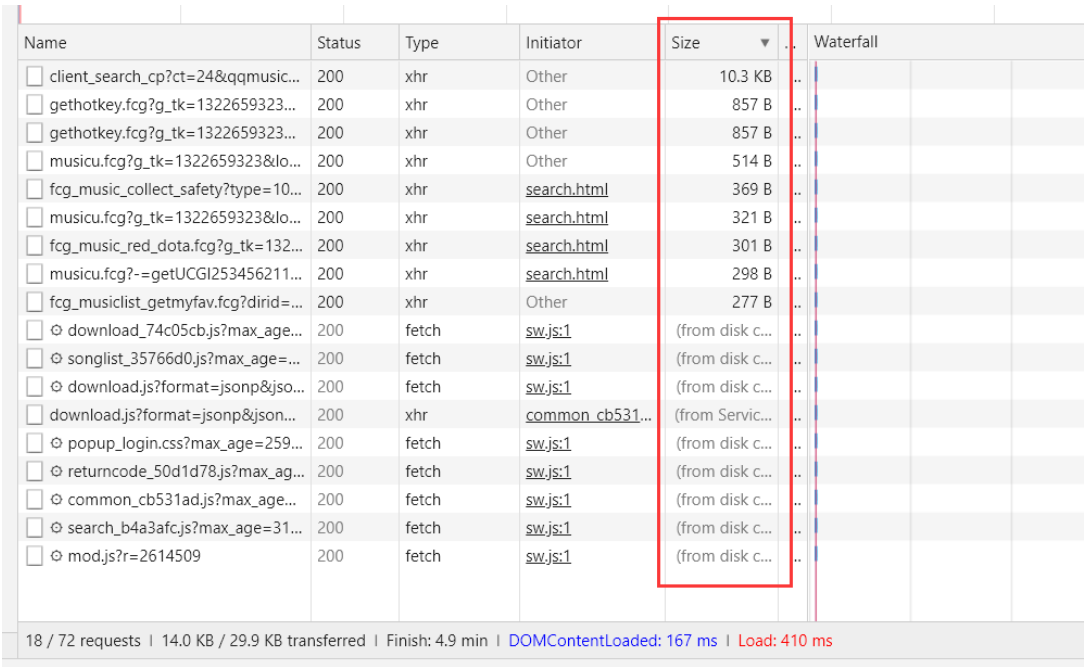
点掉后是这样的



Name	Status	Type	Initiator	Size	...	Waterfall
mod.js?r=2614509	200	fetch	sw.js:1	(from disk c...	...	
search_b4a3afc.js?max_age=31...	200	fetch	sw.js:1	(from disk c...	...	
common_cb531ad.js?max_age...	200	fetch	sw.js:1	(from disk c...	...	
returncode_50d1d78.js?max_ag...	200	fetch	sw.js:1	(from disk c...	...	
popup_login.css?max_age=259...	200	fetch	sw.js:1	(from disk c...	...	
download.js?format=jsonp&json...	200	xhr	common_cb531...	(from Servic...	...	
download.js?format=jsonp&jso...	200	fetch	sw.js:1	(from disk c...	...	
musicu.fcg?g_tk=1322659323&lo...	200	xhr	search.html	321 B	...	
fcg_music_red_dota.fcg?g_tk=132...	200	xhr	search.html	301 B	...	
fcg_music_collect_safety?type=10...	200	xhr	search.html	369 B	...	
musicu.fcg?-=getUCGI253456211...	200	xhr	search.html	298 B	...	
gethotkey.fcg?g_tk=1322659323...	200	xhr	Other	857 B	...	
gethotkey.fcg?g_tk=1322659323...	200	xhr	Other	857 B	...	
client_search_cp?ct=24&qqmusic...	200	xhr	Other	10.3 KB	...	
musicu.fcg?g_tk=1322659323&lo...	200	xhr	Other	514 B	...	
songlist_35766d0.js?max_age=...	200	fetch	sw.js:1	(from disk c...	...	
fcg_musiclist_getmyfav.fcg?dirid=...	200	xhr	Other	277 B	...	
download_74c05cb.js?max_age...	200	fetch	sw.js:1	(from disk c...	...	

18 / 71 requests | 14.0 KB / 29.9 KB transferred | Finish: 3.75 s | DOMContentLoaded: 167 ms | Load: 410 ms

我们需要点两下size



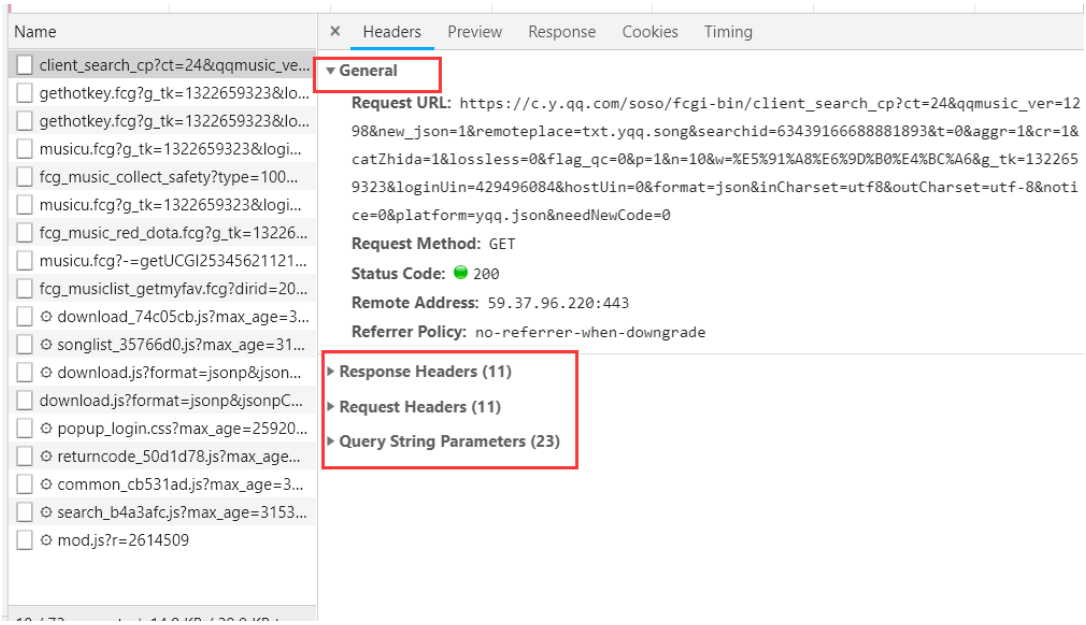
Name	Status	Type	Initiator	Size ▼	...	Waterfall
client_search_cp?ct=24&qqmusic...	200	xhr	Other	10.3 KB	...	
gethotkey.fcg?g_tk=1322659323...	200	xhr	Other	857 B	...	
gethotkey.fcg?g_tk=1322659323...	200	xhr	Other	857 B	...	
musicu.fcg?g_tk=1322659323&lo...	200	xhr	Other	514 B	...	
fcg_music_collect_safety?type=10...	200	xhr	search.html	369 B	...	
musicu.fcg?g_tk=1322659323&lo...	200	xhr	search.html	321 B	...	
fcg_music_red_dota.fcg?g_tk=132...	200	xhr	search.html	301 B	...	
musicu.fcg?-=getUCGI253456211...	200	xhr	search.html	298 B	...	
fcg_musiclist_getmyfav.fcg?dirid=...	200	xhr	Other	277 B	...	
download_74c05cb.js?max_age...	200	fetch	sw.js:1	(from disk c...	...	
songlist_35766d0.js?max_age=...	200	fetch	sw.js:1	(from disk c...	...	
download.js?format=jsonp&jso...	200	fetch	sw.js:1	(from disk c...	...	
download.js?format=jsonp&json...	200	xhr	common_cb531...	(from Servic...	...	
popup_login.css?max_age=259...	200	fetch	sw.js:1	(from disk c...	...	
returncode_50d1d78.js?max_ag...	200	fetch	sw.js:1	(from disk c...	...	
common_cb531ad.js?max_age...	200	fetch	sw.js:1	(from disk c...	...	
search_b4a3afc.js?max_age=31...	200	fetch	sw.js:1	(from disk c...	...	
mod.js?r=2614509	200	fetch	sw.js:1	(from disk c...	...	

18 / 72 requests | 14.0 KB / 29.9 KB transferred | Finish: 4.9 min | DOMContentLoaded: 167 ms | Load: 410 ms

然后请求会根据size进行一个排序

通常第一个很大可能就是我们需要的数据所在的请求啦，如果第一个请求没有，再往第二个请求找

在XHR中，我们需要跟着preview去查看是否有我们需要的内容，找到请求之后，再回到Headers



Headers下的四个内容，我们在后面几关的课程都会学习到

在第一个General里的Requests URL就是我们应该去访问的链接

但是，requests.get()XHR里的URL后，使用tag.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。

于是，我们需要将普通字符串类似的数据，转换为json格式的数据转，也就是列表/字典的字符串

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

这种特殊的写法决定了，json能够有组织地存储信息。

我们可以将json格式的数据，转换成正常的列表/字典，这样就可以愉快的用列表和字典提取元素格式去提取我们需要的数据了

格式如：res.json()

这里也提一个，细心的同学可能发现，后面的课程例子中，还有json.loads()这种用法

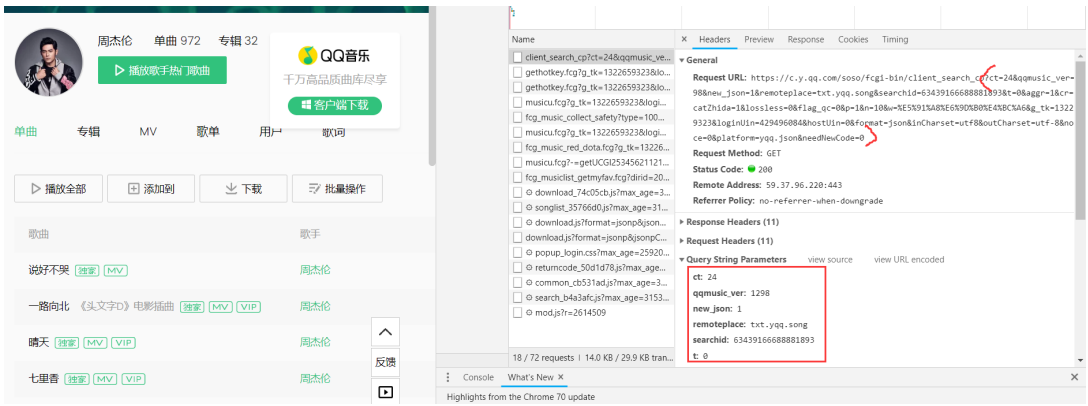
其实这两者，在实现的功能上没什么区别都是解析json库，只是传递的值不一样

使用 json.loads(res.text) 需要导入json模块，而res.json() 是requests的方法

第5关

到了第5关，我们将开始用参数去请求数据

这个参数，指XHR里， Headers里的四种类型的参数，比如Params



看到这张图，其实Params部分这些参数，就是上面URL的尾部信息

观察URL
前半部分大多形如：https://xx.xx.xxx/xxx/xxx
后半部分，多形如：xx=xx&xx=xxx&xxxxx=xx&……
两部分使用?来连接。

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，我们想要什么样的数据。

在写代码的时候，我们会将Params复制下来并封装为一个字典

这里提供一种方法，直接把给params加引号
https://blog.csdn.net/mouday/article/details/80460612

当我们要爬取连续的网页时，通常要去仔细的研究params的参数变化

找到每一页的变化规律，作为变量

并通过循环的方式去改变变量，这样来达到爬取连续网页的效果

除了Params后，我们还会加上请求头，也就是Request Headers的内容

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

#封装请求头

```
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) Chr
    # 标记了请求从什么设备，什么浏览器上发出
    }
```

这里建议在添加请求头的时候，把上图的三个都加上

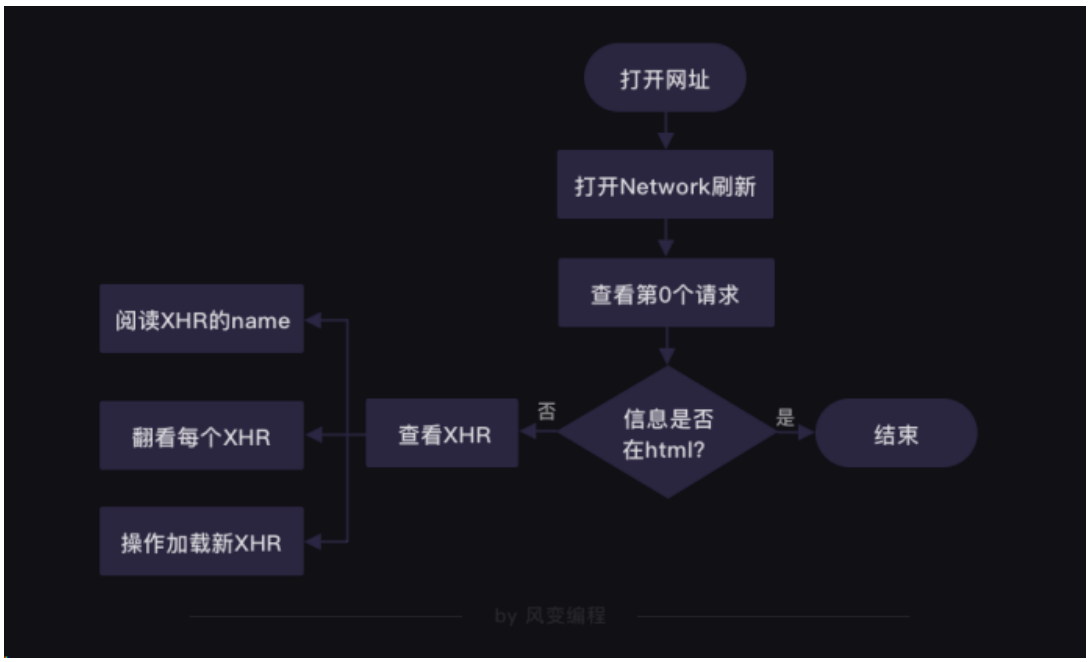
origin、referer、user-agent

封装好Params和Header之后，那我们就可以愉快的发起请求了

#作为参数传递

```
res_music = requests.get(url,headers=headers,params=params)
# 发起请求，填入请求头和参数
```

梳理一下整个流程，是这样的



至此，第5关结束，前面学习了请求、解析、提取的知识

还记得我们上面讲的爬虫4个步骤吗

我们在这里讲了

获取数据
解析数据
提取数据

而到了第6关的学习，我们则会开始学习储存数据的操作

所以到了山腰，才算是真真正正爬虫的入门哦

最后，用一张思维导图来总结今晚所讲的内容