第八关 看不见的数据：动态网页爬取

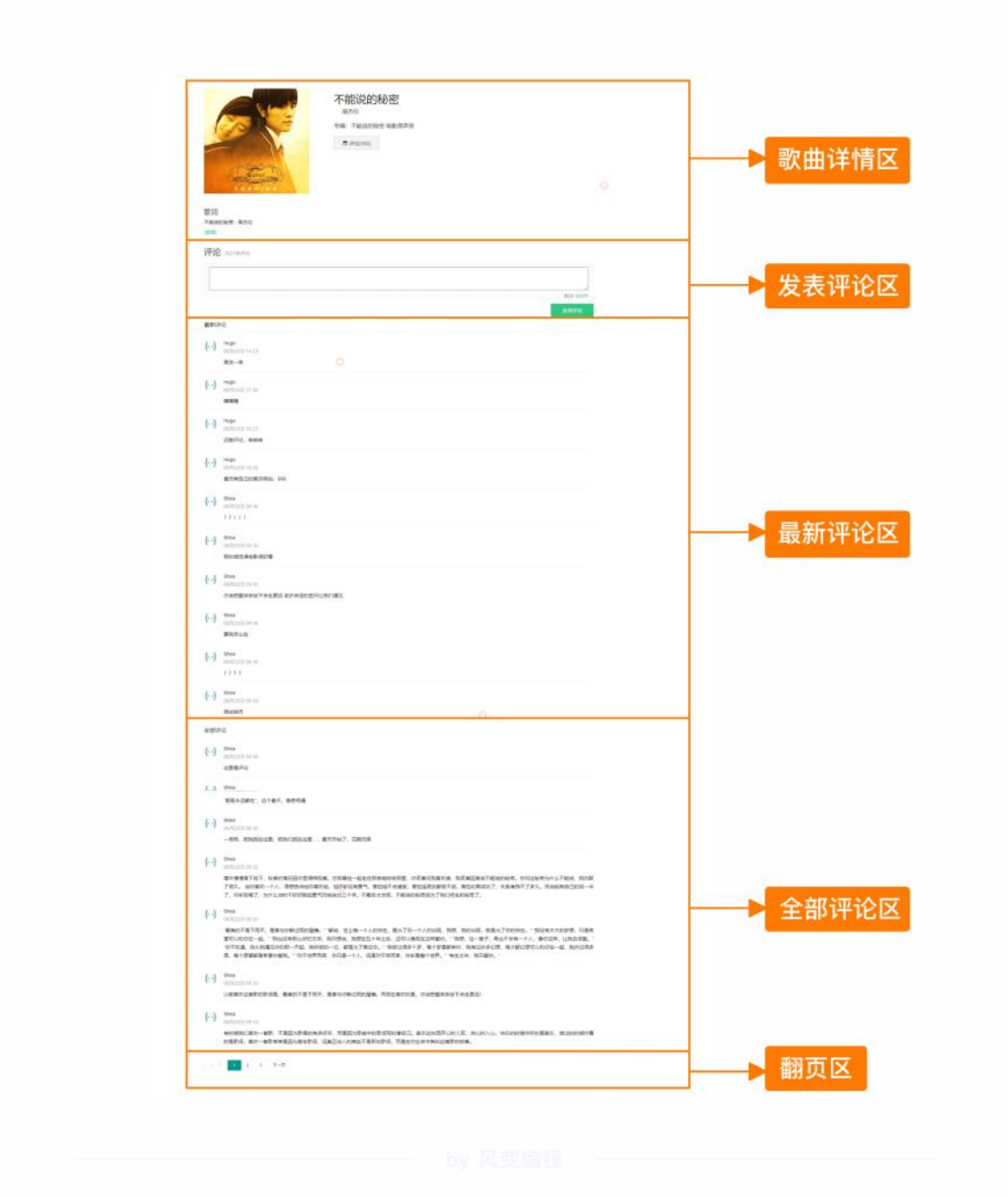
2022年12月14日 20:14

爬取《不能说的秘密》歌曲详情页：https://music.facode.cn/index.php/Home/Index/voice_details/id/403778 所有评论信息。

1.2 网页分析

打开《不能说的秘密》歌曲详情页：https://music.facode.cn/index.php/Home/Index/voice_details/id/403778

经过观察，可以看到整个页面主要分为 5 个区域，从上到下依次是：歌曲详情区、发表评论区、最新评论区、全部评论区以及翻页区。



由于要爬取评论信息，因此我们将目光聚焦在最新评论区、全部评论区以及翻页区。

请你探索一下歌曲详情页，然后回答下面的问题。

单选题

在翻页区点击【页码数】或者【下一页】，会对哪个区域进行翻页？

A.

最新评论区

B.

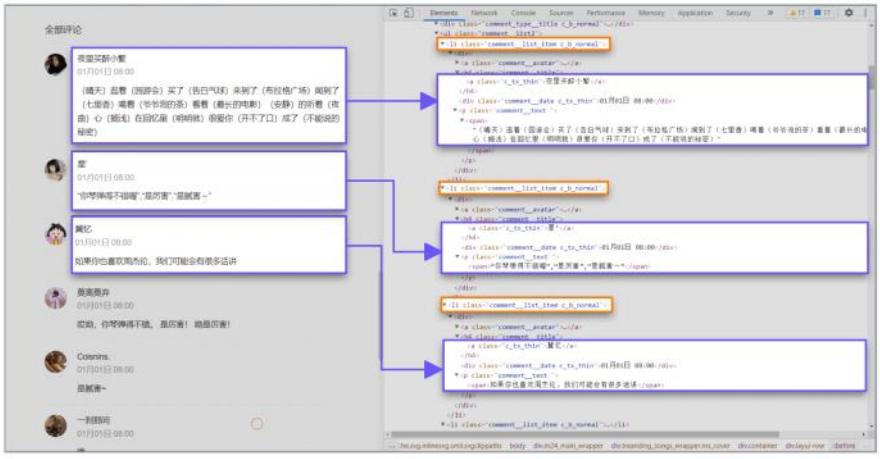
全部评论区

C.

最新评论区和全部评论区

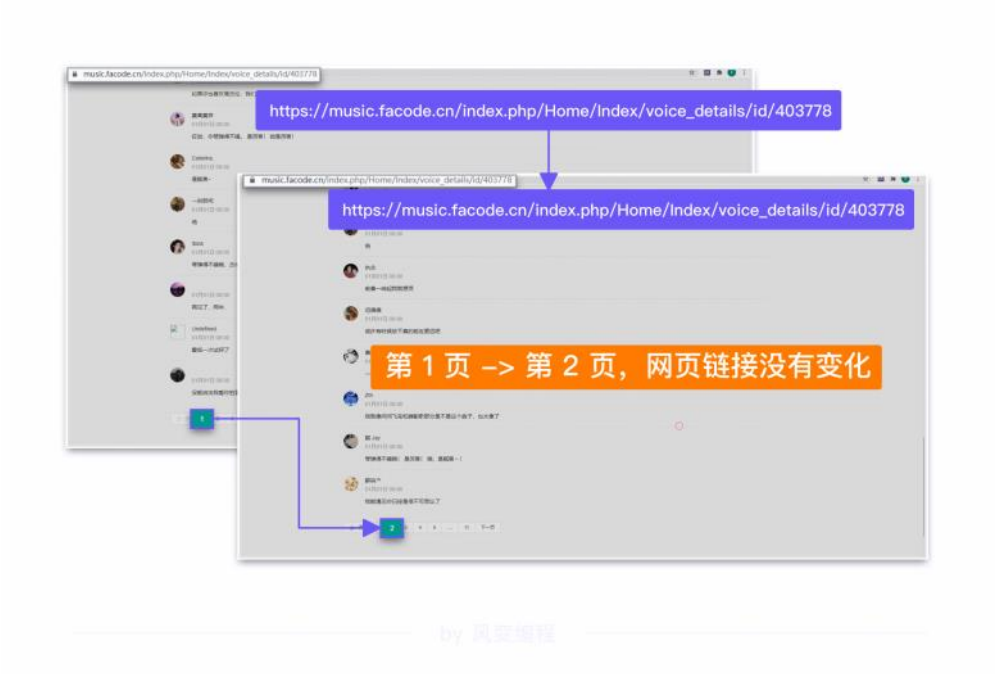
D.
歌曲详情页
回答正确
点击页码【2】或者【下一页】，可以看到全部评论区第 2 页内容（以下简称第 2 页评论）；继续点击【3】或者【下一页】，可以看到全部评论区第 3 页内容（以下简称第 3 页评论）。在此过程中，其他区域的内容不变，因此点击【页码数】或者【下一页】，只会对全部评论区进行翻页。
进一步探索，发现**最新评论区**会展示距离当前时间较近的评论，最多可以展示 10 条；**全部评论区**用来存放所有评论，可以展示多于 10 条的内容，但是当评论总数多于 10，需要分页展示。

比如评论总数为 27，全部评论区会这样分页：第 1 页 10 条、第 2 页 10 条、第 3 页 7 条。
既然全部评论区会存放所有评论，那么我们的关注点就进一步聚焦在**全部评论区**。
结合前面爬取闪光读书评论、豆瓣电影 Top250 前 100 部电影信息的经验，当一个网站有多页需要爬取时，我们的分析步骤一般为：
1）定位目标信息在 Elements 选项卡的位置；
2）探索不同页链接的规律。
下面我们就按照上面的步骤一步一步进行分析。
首先，滑动到全部评论区，用【指针工具】定位评论信息在 Elements 选项卡的位置，发现每条评论的评论信息均属于标签 `<li class="comment__list_item c_b_normal">`。



向上探索，发现 <ul class="comment__list2"> 是所有评论信息的父节点。

然后，探索全部评论区**不同页链接**是否有一定规律。
我们点击下一页，看看网页链接有什么变化。
已经跳转到第 2 页了，但是网页链接并没有变化。



再跳转至第 3 页。

有点慌，网页链接还是没有变化。看来，没有找到我们想要的规律。

这里做一个大胆的猜测，**闪光音乐的歌曲详情页**和以往接触的网页，在加载方式存在差异。

为了理清这中间发生了什么，我们打开 Network 选项卡，查看网页中加载的所有请求。

先看 HTML 文档所在的请求，即 document 类型的请求。

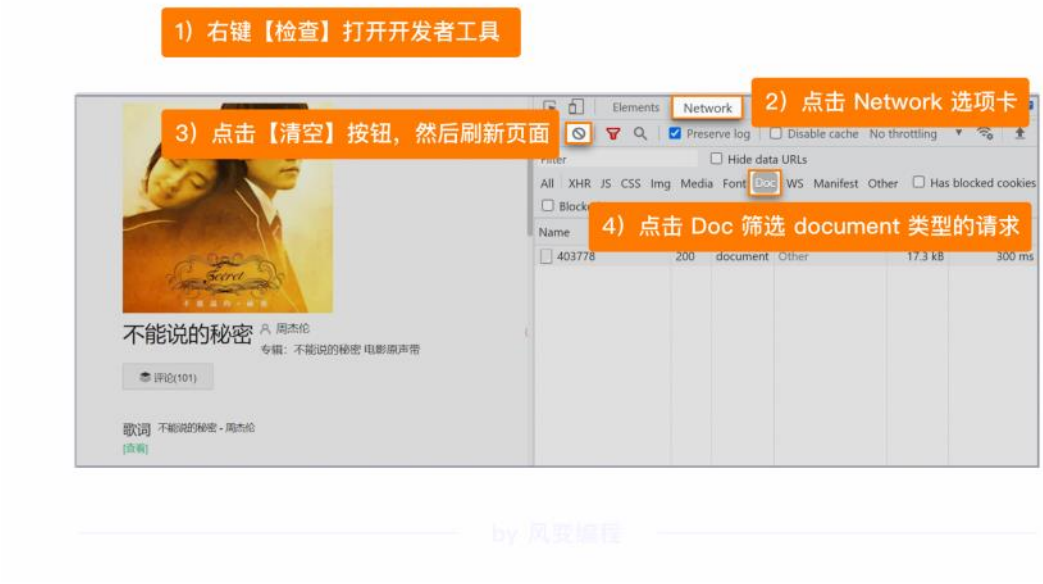
你可以参照下面的步骤操作：

1）右键【检查】打开开发者工具；

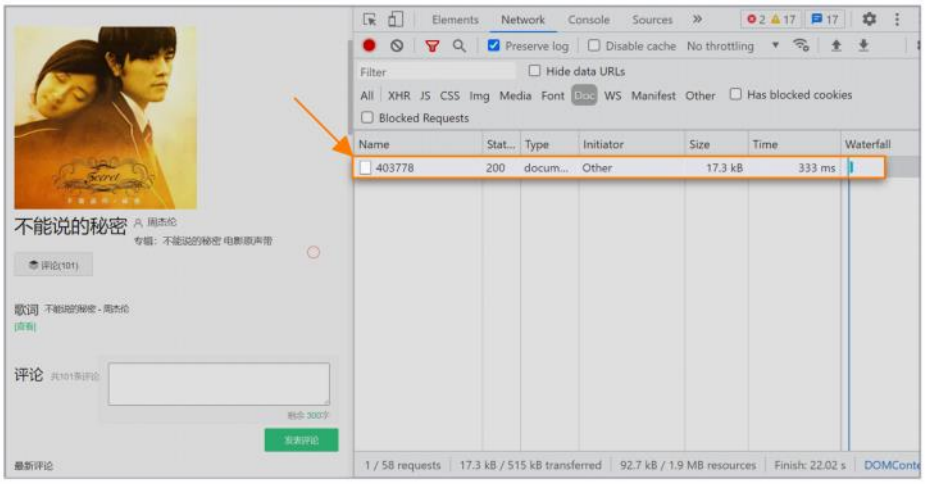
2）点击 Network 选项卡；

3）点击【清空】按钮，然后刷新页面；

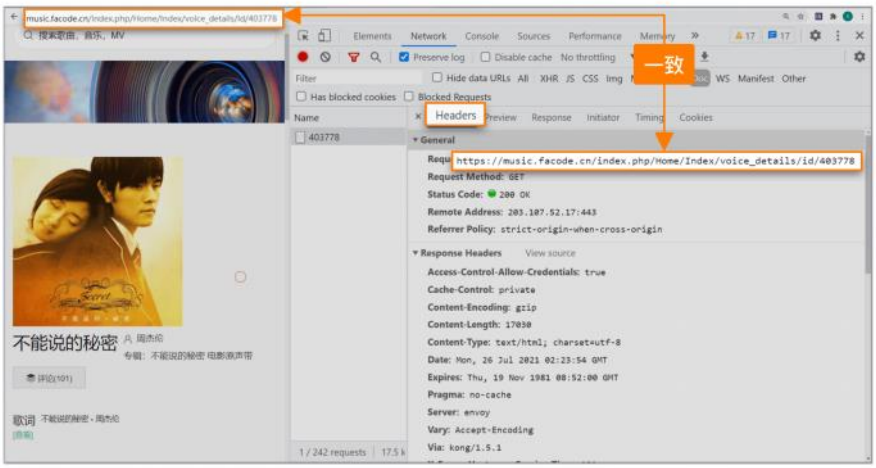
4）点击 Doc 筛选 document 类型的请求。



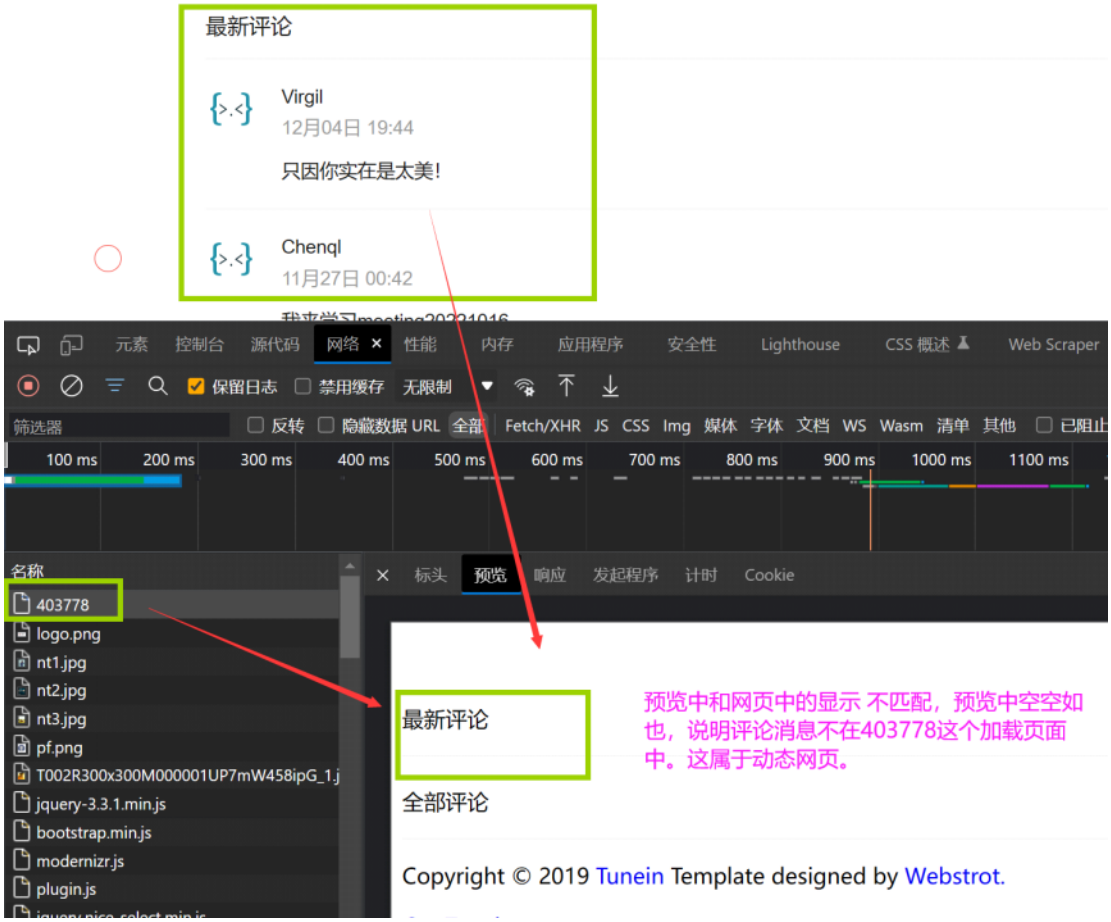
可以看到，当前网页只有一个 document 类型的请求 —— **403778**。



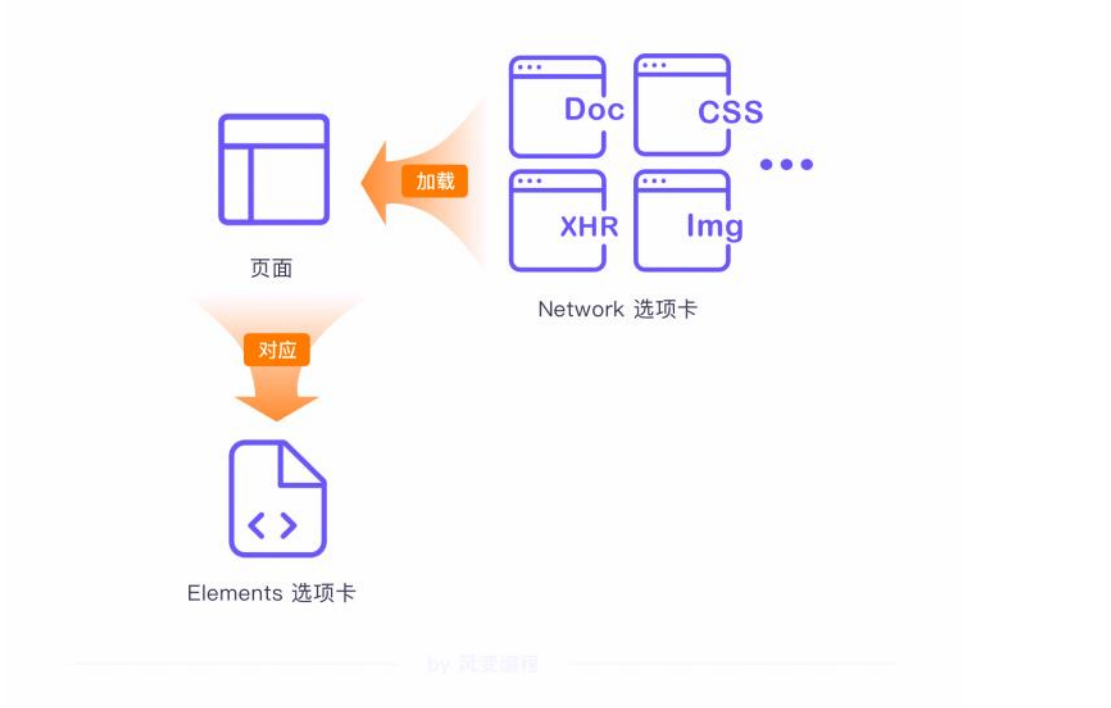
我们打开它，点击 Headers 观察其请求信息，发现请求链接（Request URL）和网页链接一致，说明 403778 确实是页面 HTML 文档所在的请求。



切换到 Preview，并向下滑动，可以看到最新评论区和全部评论区对应的内容空空如也。看来我们想要的评论区内容并不在 HTML 文档所在的请求中。

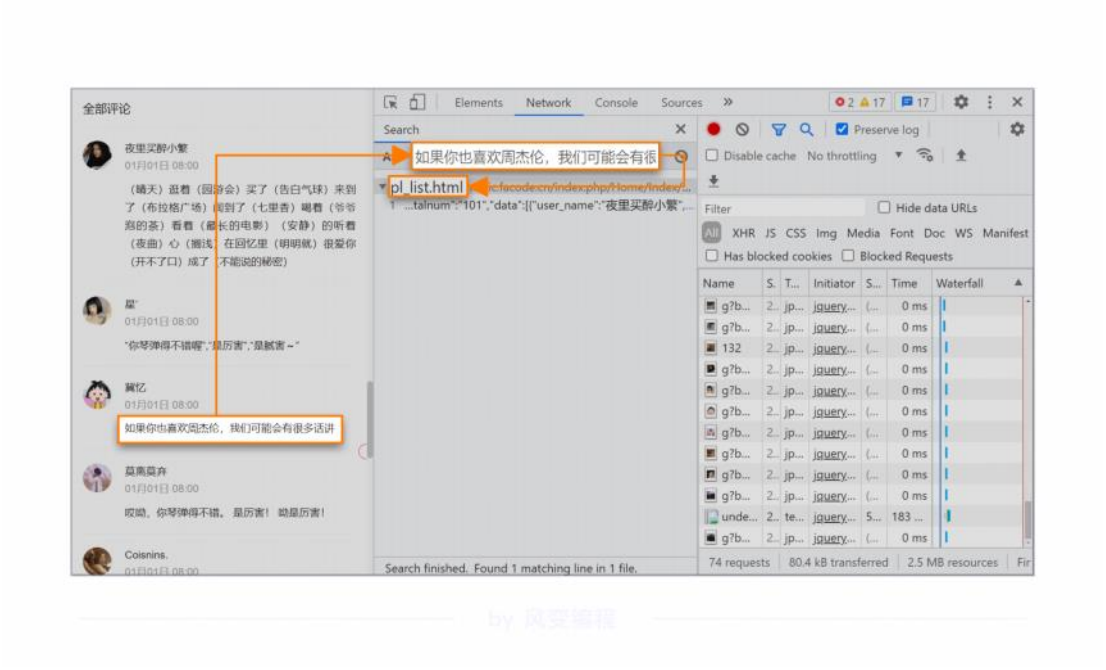


这时你可能会问，我们刚刚不是已经在 Elements 中成功定位了全部评论区的内容吗?
准确来讲，Elements 选项卡呈现了最终页面效果对应的 HTML 代码，而页面是由 Network 选项卡中各类请求共同加载而成的。

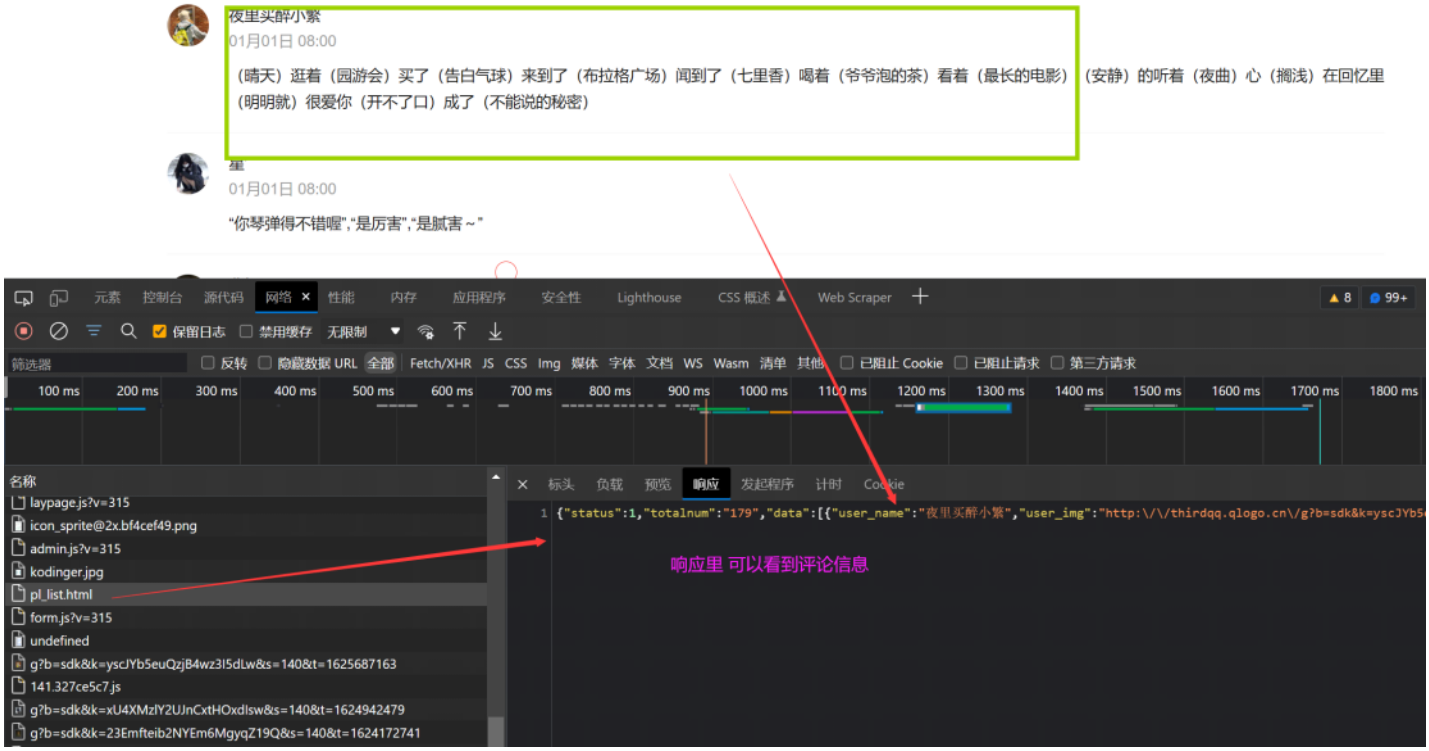


也就是说，在 Elements 选项卡中的某个内容，并不一定在 HTML 文档所在的请求（document）中。
而且实际上我们要找的目标内容刚好在别的请求中。
请你打开【全局搜索】，直接根据内容关键词定位请求。

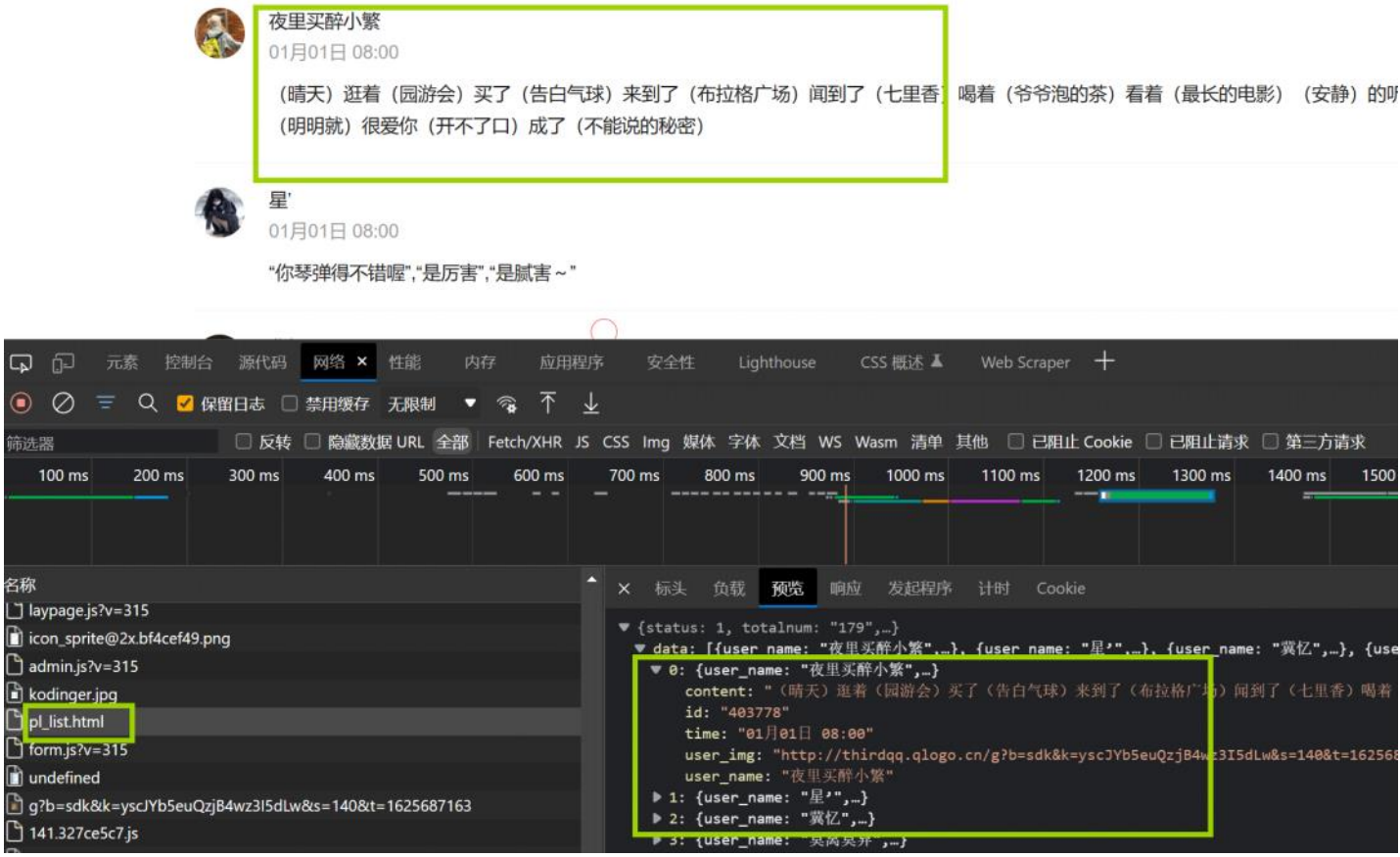
我选择了全部评论区的一条评论内容进行搜索：如果你也喜欢周杰伦，我们可能会有很多话讲。
你也可以试试别的。
找到了！这条评论内容在名称为 `pl_list.html` 的请求中。



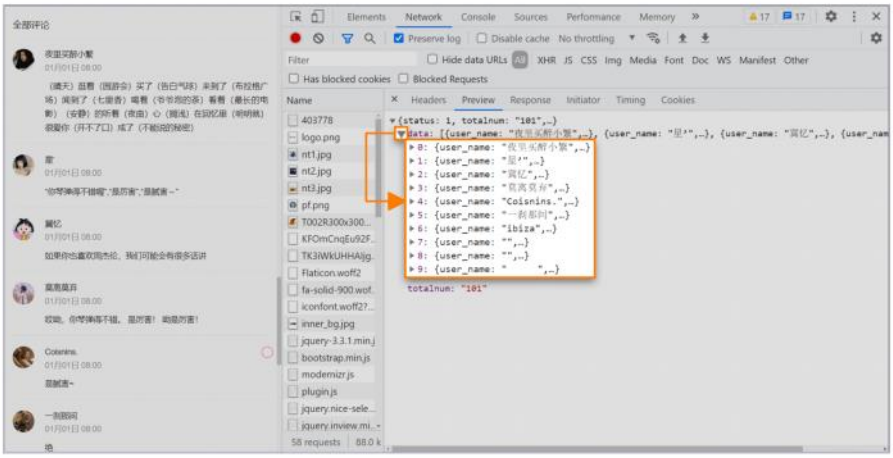
现在，直接点击搜索结果，可以切换到该请求的 Response，即网页的响应内容。



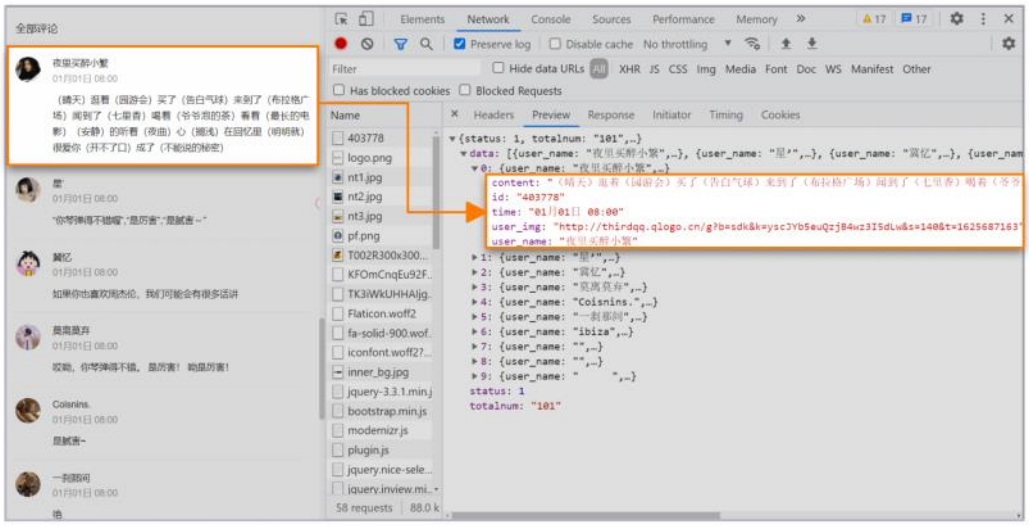
映入眼帘的只有一行数据，而且这行数据是嵌套字典的结构。我们可以向右拖动滑块，查看它的完整内容。
但说实话，可读性有点差。我们还是切换到 Preview 吧。因为浏览器开发者工具对 Response 中的响应内容做了专门的处理，并将处理后的内容放在 Preview，方便
我们进行查看。



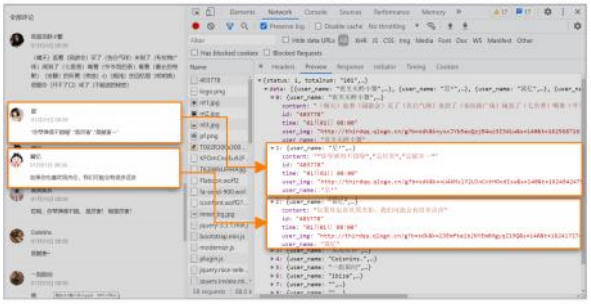
类似于 Elements 选项卡界面，部分数据默认是折叠的，点击左侧的小三角可以展开数据。
现在，请你点击 data 左侧的小三角，展开 data 中的数据。



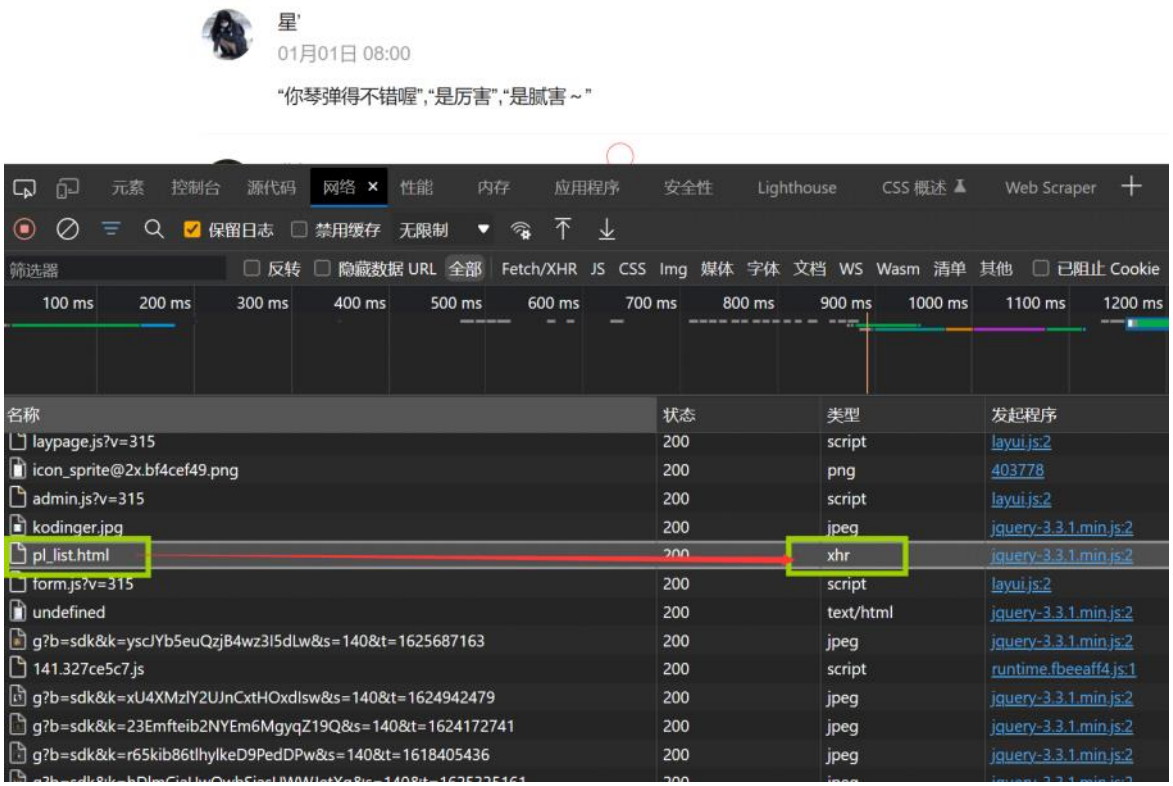
展开后可以看到 0 ~ 9，共 10 条数据，这 10 条数据可能就是当前全部评论区的 10 条评论信息。
继续点击 0 左侧的小三角，展开其中的数据。
果然！我们找到了第 1 条评论的信息，包括评论内容（content）、歌曲 id（id）、评论时间（time）、用户头像链接（user_img）、用户名（user_name）。



我们的目标是爬取评论的用户名、评论时间以及评论内容，看来这里的内容已经能满足我们的需求。
同理，展开 1 可以找到全部评论区的第 2 条评论，展开 2 可以找到全部评论区的第 3 条评论。



依此可得，全部评论区第 1 页（以下简称**第 1 页评论**）的 10 条评论信息都存放在 **pl_list.html** 请求的响应内容中。
那么，我们就完成对**第 1 页评论**的定位。
关闭当前的 **pl_list.html** 请求（点击 Headers 左边的关闭按钮），发现它是 xhr 类型。

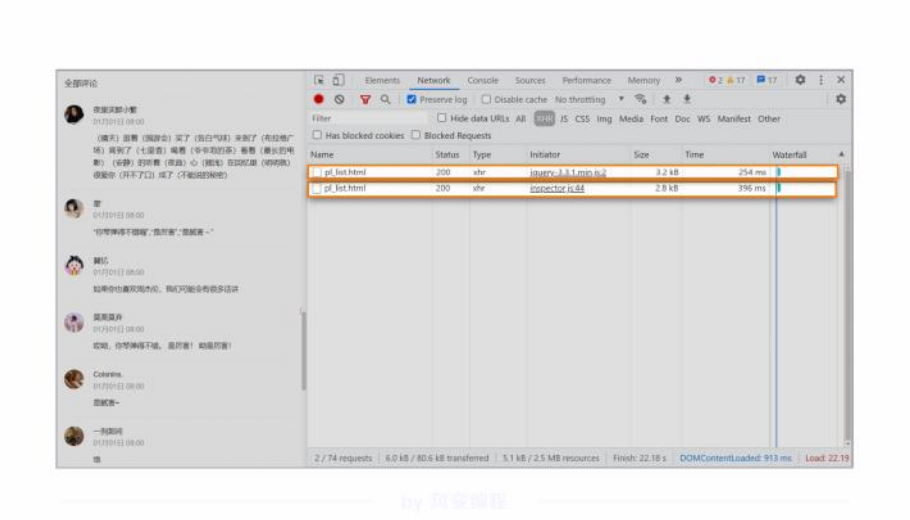


最新评论和其他页评论是否也在 xhr 请求中呢？我们来验证一下。
请你点击 XHR 筛选 xhr 请求（如果勾选了【Preserve log】，还要点击【清除】按钮），然后刷新页面，就能找到最新的需求。

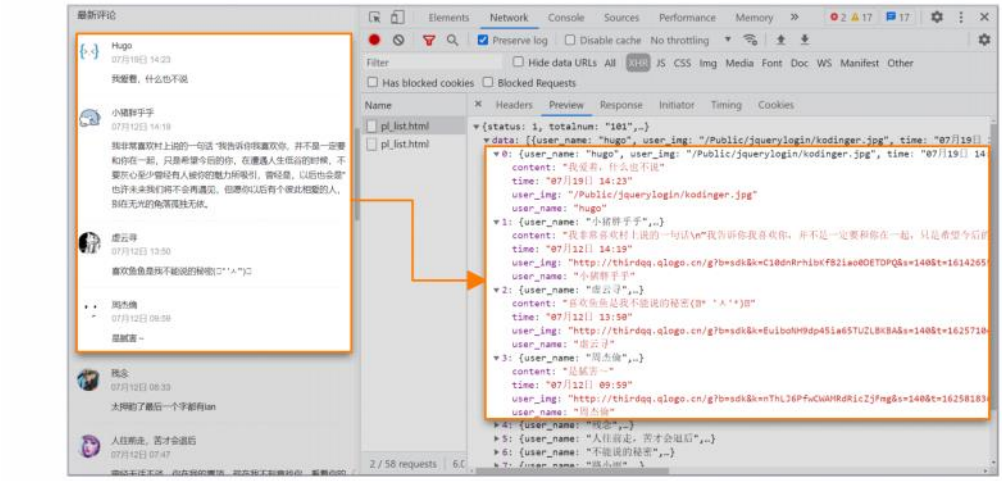
你可以打开下面的视频，观看具体的操作演示。

https://static.pandateacher.com/rc-upload-1627285488167-5%5E%E7%AD%9B%E9%80%89xhr%E8%AF%B7%E6%B1%82v3@hdv.mp4

筛选之后，请求界面只剩下两个名称为 **pl_list.html** 的 xhr 请求。



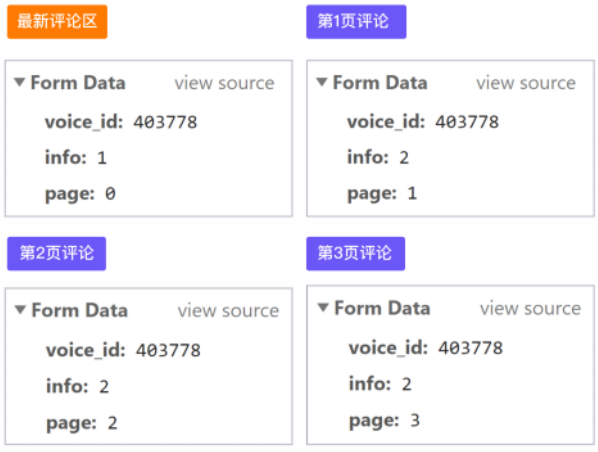
打开并查看 Preview，其中一个是**第 1 页评论**对应的请求，另一个则是**最新评论**区对应的请求。（本课所有图片中出现的数据为截图时的数据，数据会更新，以网页中观察到的为准）



不要关闭开发者工具，我们继续定位**第 2 页评论**和**第 3 页评论**。
打开第 2 页评论，会弹出一个新的 **xhr** 请求，查看 Preview，刚好是第 2 页的内容。

https://static.pandateacher.com/rc-upload-1626840438098-9%5E%E7%82%B9%E5%87%BB%E4%B8%8B%E4%B8%80%E9%A1%B5-%E5%AD%97%E5%B9%95%E7%89%88v2@hdv.mp4

同样，打开第 3 页评论，弹出的 **xhr** 请求也会对应第 3 页的内容，你可以动手试试。
那么到这里，我们就完成对**评论区所有内容**的定位。
最新评论区和全部评论区的每一页评论都以 **xhr** 请求进行传递，每个请求的名称相同，但是其 Preview 却不同。
每次都通过 Preview 区分不同请求稍显麻烦，有没有更简洁的特征呢？
我们对比一下各个请求在 Headers 中的请求信息，看看有什么区别。
经过一番摸索，可以发现它们的请求体【Form Data】不同，并且有一定的规律。



全部评论区 info 的值均为 2，并且 page 和页数对应。最新评论区则比较特殊，info 的值为 1 ，page 的值为 0。
这样，我们就能通过【Form Data】区分不同的评论区请求。
小结一下，**闪光音乐歌曲详情页**和之前的网页在加载方式上确实存在差异。
差异一：评论区内容不在网页链接对应的 HTML 文档中，而在 xhr 类型的请求中。
差异二：翻页时整个页面不会刷新，只对全部评论区内容进行加载。

1.3 代码体验

在网页分析阶段，我们知道评论区内容不在 HTML 文档中，而是在 pl_list.html 请求中。
这是什么原因造成的呢？我们又应该如何爬取呢？
在这之前，先来体验一下我写的项目代码，再进行下一步的学习吧。

```python
# 代码体验
# 爬取网页https://music.facode.cn/index.php/Home/Index/voice_details/id/403778 里的所有评论信息

import requests
import csv

# 设置歌曲详情页链接
song_url = 'https://music.facode.cn/index.php/Home/Index/voice_details/id/403778'
# 设置评论请求链接
comment_url = 'https://music.facode.cn/index.php/Home/Index/pl_list.html'

# 创建空列表存储字典数据
comment_list = []
```

```python
for page in range(1,19):

# 设置评论表单递交参数
    payload = {
    'voice_id': 403778,
    'info': 2,
    'page': page
    }

    # 使用requests访问评论页面
    comment_res = requests.post(comment_url, data=payload)
```

```python
    # 上面得到的comment_res是一个json ，下面需要把json转换成python的字典
    comment_dict = comment_res.json()

    # 循环获取评论信息
    for comment_data in comment_dict['data']:
        # 将评论信息写入字典
        comment_info = {'用户名':comment_data['user_name'],
                        '评论内容':comment_data['content'],
                        '评论时间':comment_data['time']
                        }
        comment_list.append(comment_info)

    print(f'第{page}页爬取完成')
```

```
# 写入csv文件
with open('D:\PythonTest\风变python学习资料\Python爬虫\comment.csv', 'w', newline='', encoding='utf-8-sig')
 as f:
    comment_csv = csv.DictWriter(f,fieldnames=['用户名','评论内容','评论时间'])
    comment_csv.writeheader()
    comment_csv.writerows(comment_list)
    print('存储完毕')
```

```
第11页爬取完成
第12页爬取完成
第13页爬取完成
第14页爬取完成
第15页爬取完成
第16页爬取完成
第17页爬取完成
第18页爬取完成
存储完毕
```

```
# 代码体验
# 爬取网页https://music.facode.cn/index.php/Home/Index/voice_details/id/403778 里的所有评论信息

import requests
import csv

# 设置歌曲详情页链接
song_url = 'https://music.facode.cn/index.php/Home/Index/voice_details/id/403778'
# 设置评论请求链接
comment_url = 'https://music.facode.cn/index.php/Home/Index/pl_list.html'

# 创建空列表存储字典数据
comment_list = []

for page in range(1,19):

# 设置评论表单递交参数
  payload = {
  'voice_id': 403778,
  'info': 2,
  'page': page
  }

  # 使用requests访问评论页面
  comment_res = requests.post(comment_url, data=payload)

  # 上面得到的comment_res是一个json， 下面需要把json转换成python的字典
  comment_dict = comment_res.json()

  # 循环获取评论信息
  for comment_data in comment_dict['data']:
    # 将评论信息写入字典
    comment_info = {'用户名':comment_data['user_name'],
            '评论内容':comment_data['content'],
            '评论时间':comment_data['time']
            }
    comment_list.append(comment_info)

  print(f'第{page}页爬取完成')

# 写入csv文件
with open('D:\PythonTest\风变python学习资料\Python爬虫\comment.csv', 'w', newline='', encoding='utf-8-sig') as f:
  comment_csv = csv.DictWriter(f,fieldnames=['用户名','评论内容','评论时间'])
  comment_csv.writeheader()
  comment_csv.writerows(comment_list)
```

```
print('存储完毕')
```

3. 爬取单页歌曲评论

3.1 静态网页和动态网页

之前一直畅通无阻，是因为目标是**静态网页**，我们可以直接通过 HTML 文档所在的请求获取所有内容。
比如闪光读书网的书籍评论，豆瓣电影 Top250 的电影信息。
而现在，目标变了！闪光音乐歌曲详情页是**动态网页**，它的部分内容在HTML 文档，部分内容在 xhr 请求的响应内容中，加载过程是分步完成的。
类似的例子还有很多，下面我就通过视频给你展示一下：
1）在某搜索引擎输入不同的关键词，会出现不同的搜索提示；

https://static.pandateacher.com/rc-upload-1627285488167-3%5E%E6%90%9C%E7%B4%A2%E5%BC%95%E6%93%8E%E4%B8%8D%E5%90%8C%E6%8F%90%E7%A4%BA@hdv.mp4

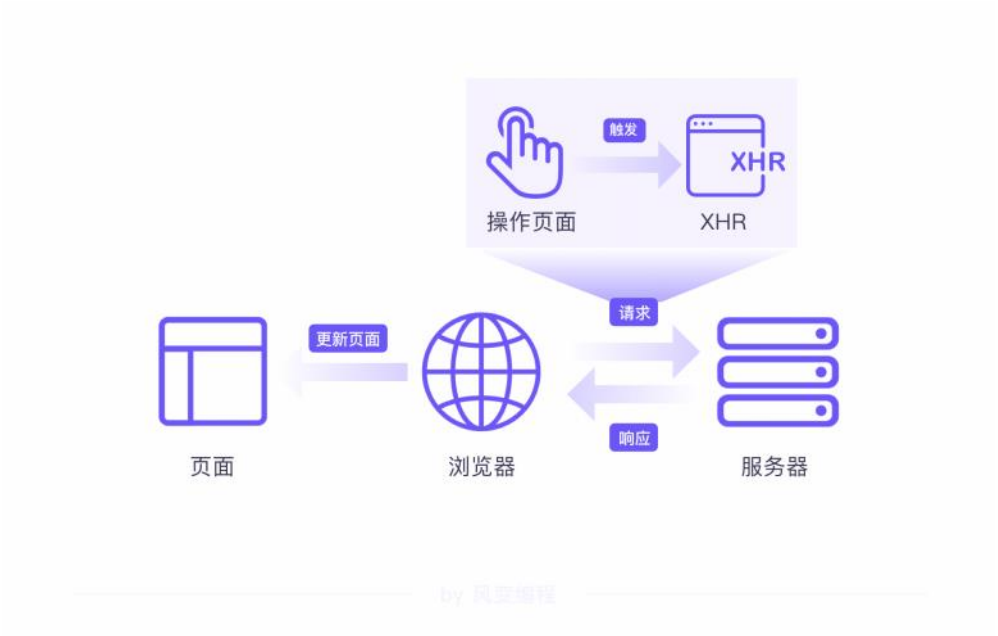
2）在某博热门资讯页向下滚动页面，可以不断地刷出新的内容；

https://static.pandateacher.com/rc-upload-1626078789496-5%5E%E5%BE%AE%E5%8D%9A%E5%90%91%E4%B8%8B%E6%BB%9A%E5%8A%A8%E9%A1%B5%E9%9D%A2@hdv.mp4

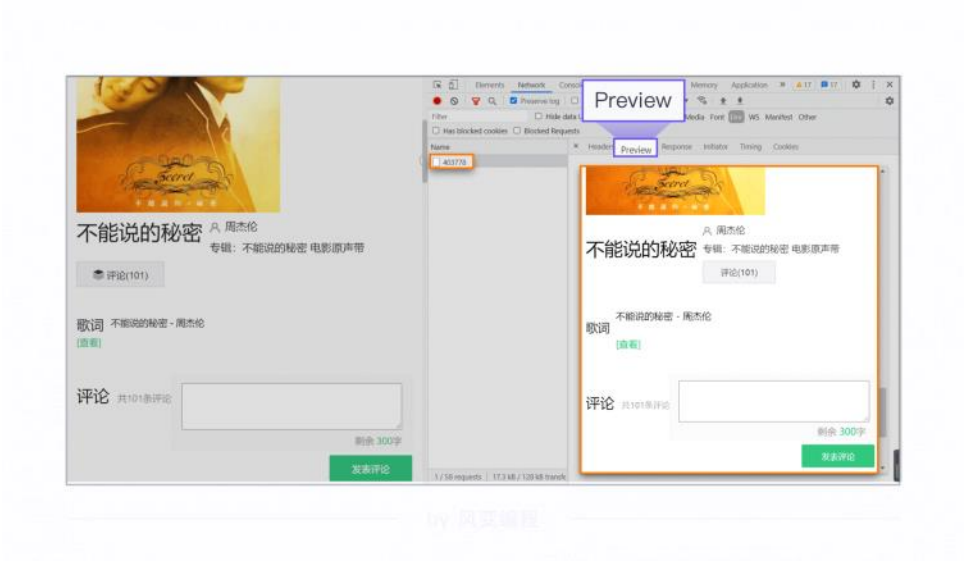
这些网页都采用了动态加载，我们不能再通过 HTML 文档所在的请求获取所有内容。
那该咋办嘛？总不能束手就擒吧！
方法总归是有的，我们可以通过 xhr 类型的请求找到我们想要的内容。

3.2 AJAX 和 xhr

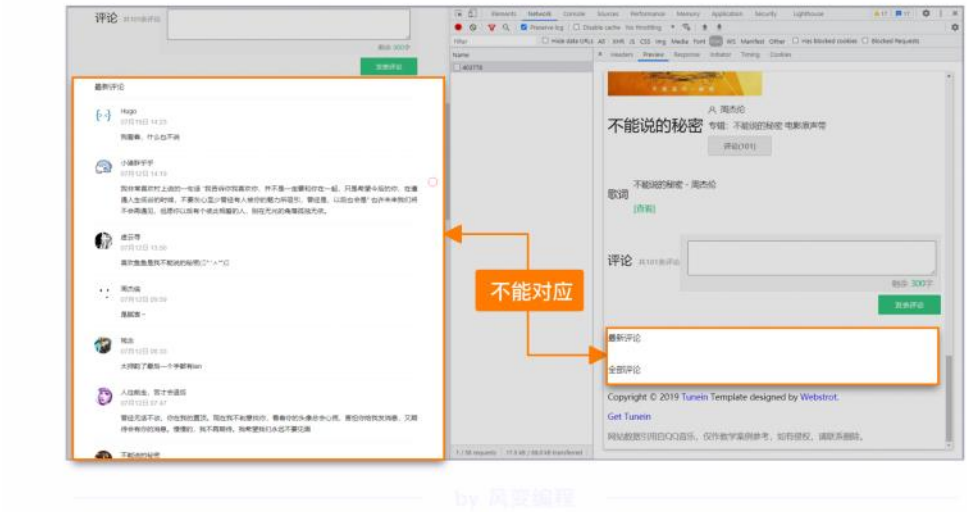
AJAX 是一套综合了多项技术的网页开发技术，它的核心是 XMLHttpRequest 对象（以下简称 XHR）。
其最大的优势就是在**不刷新页面**的情况下，通过 XHR 向服务器请求数据，然后插入到页面中进行呈现。



那么，它是如何应用的呢？
我们依然以歌曲《不能说的秘密》的详情页为例，分析一下它的加载过程。
当我们打开歌曲详情页的时候，浏览器会发送请求，获取 HTML 文档所在的 **403778** 请求并进行加载。你可以在它的 Preview 中看到加载结果。



在加载结果中，评论区内容是空的，那么最新评论区和全部评论区的内容是怎么来的呢？



这是因为加载过程中触发了 xhr 请求，也就是 **pl_list.html**。每一次翻页同样也会触发 xhr 请求，并对全部评论区内容进行替换。**403778** 和 **pl_list.html** 请求的内容不同，因此触发 xhr 请求不会影响网页链接，即**不刷新页面**的情况下就能更新全部评论区内容。接下来，我们就来详细看看 **pl_list.html** 的请求内容和响应内容，并尝试获取评论区内容。

```python
# 导入模块
import requests

# 设置评论区请求链接
comment_url = 'https://music.facode.cn/index.php/Home/Index/pl_list.html'

# 设置第 1 页表单提交数据
data = {'voice_id': 403778,
        'info': 2,
        'page': 1}

# 发起请求，获取网页内容
comment_res = requests.post(comment_url, data=data)

# 打印网页返回的状态码
print(comment_res.status_code)

# 打印字符串类型的评论信息
print(comment_res.text)
```

```
200
{"status":1,"totalnum":"179","data":[{"user_name":"夜里买醉小繁","user_img":"http:\/\/thirdqq.qlogo
.cn\/g?b=sdk&k=yscJYb5euQzjB4wz3I5dLw&s=140&t=1625687163","time":"01月01日 08:00","content":"（晴天）逛
着（园游会）买了（告白气球）来到了（布拉格广场）闻到了（七里香）喝着（爷爷泡的茶）看着（最长的电影）（安静）的听着（夜曲）
心（搁浅）在回忆里（明明就）很爱你（开不了口）成了（不能说的秘密）","id":"403778"},{"user_name":"星’’,
"user_img":"http:\/\/thirdqq.qlogo.cn\/g?b=sdk&k=xU4XMzlY2UJnCxtHOxdIsw&s=140&t=1624942479",
"time":"01月01日 08:00","content":"“你琴弹得不错喔”,“是厉害”,“是腻害～”","id":"403778"},{"user_name":"冀忆
","user_img":"http:\/\/thirdqq.qlogo.cn\/g?b=sdk&k=23Emfteib2NYEm6MgyqZ19Q&s=140&t=1624172741",
```

```
#获取单页评论区内容
#导入模块
importrequests

#设置评论区请求链接
comment_url='https://music.facode.cn/index.php/Home/Index/pl_list.html'

#设置第1页表单提交数据
data={'voice_id':403778,
'info':2,
'page':1}

#发起请求，获取网页内容
comment_res=requests.post(comment_url,data=data)

#打印网页返回的状态码
print(comment_res.status_code)

#打印字符串类型的评论信息
print(comment_res.text)
```

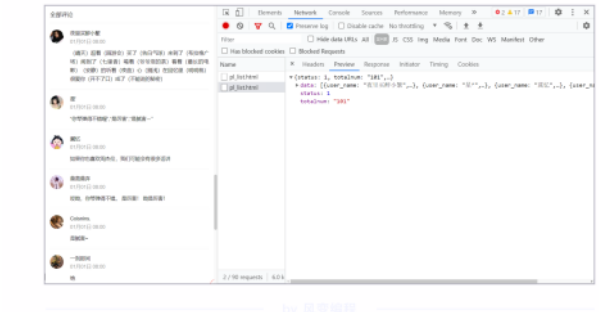
发现这些数据是这样的

```
终端

200
{"status":1,"totalnum":"100","data":[{"user_name":null,"user_img":null,"time":"01月01日 08:00","content":"（晴天
）逛着（园游会）买了（告白气球）来到了（布拉格广场）闻到了（七里香）喝着（爷爷泡的茶）看着（最长的电影）（安静）
的听着（夜曲）心（搁浅）在回忆里（明明就）很爱你（开不了口）成了（不能说的秘密）","id":"403778","img":"http:\/\/t
hirdqq.qlogo.cn\/g?b=sdk&k=yscJYb5euQzjB4wz3I5dLw&s=140&t=1625687163","nickname":"夜里买醉小繁","like_num":"0"},{
"user_name":null,"user_img":null,"time":"01月01日 08:00","content":"“你琴弹得不错喔”,“是厉害”,“是腻害～”","id":"4
03778","img":"http:\/\/thirdqq.qlogo.cn\/g?b=sdk&k=xU4XMzlY2UJnCxtHOxdIsw&s=140&t=1624942479","nickname":"星’","l
ike_num":"0"},{"user_name":null,"user_img":null,"time":"01月01日 08:00","content":"如果你也喜欢周杰伦，我们可能会
有很多话讲","id":"403778","img":"http:\/\/thirdqq.qlogo.cn\/g?b=sdk&k=23Emfteib2NYEm6MgyqZ19Q&s=140&t=1624172741"
,"nickname":"冀忆","like_num":"0"},{"user_name":null,"user_img":null,"time":"01月01日 08:00","content":"哎呦，你
琴弹得不错。\n是厉害！\n呦是厉害！","id":"403778","img":"http:\/\/thirdqq.qlogo.cn\/g?b=sdk&k=r65kib86tlhylkeD9Pe
dDPw&s=140&t=1618405436","nickname":"莫离莫弃","like_num":"0"},{"user_name":null,"user_img":null,"time":"01月01日
 08:00","content":"是腻害~","id":"403778","img":"http:\/\/thirdqq.qlogo.cn\/g?b=sdk&k=hDlmCiaUwQwbSiasUWWJetXg&s=
140&t=1625325161","nickname":"Coisnins.","like_num":"0"},{"user_name":null,"user_img":null,"time":"01月01日 08:00
","content":"绝","id":"403778","img":"https:\/\/thirdwx.qlogo.cn\/mmopen\/vi_32\/fpfzRHjhmW5bjliaBqYEmOWRW10B5kxS
VMaZPABcic8M83fGSwoRUAhb4ymzJrwmLmBjfjq6XFN7Mcob6dHqx5Qg\/132","nickname":"一刹那间","like_num":"0"},{"user_name"
:null,"user_img":null,"time":"01月01日 08:00","content":"琴弹得不错哦，杰伦","id":"403778","img":"http:\/\/thirdq
q.qlogo.cn\/g?b=sdk&k=yG6ngtp3d1ZumfiacJ013jA&s=140&t=1624691500","nickname":"ibiza","like_num":"0"},{"user_name"
:null,"user_img":null,"time":"01月01日 08:00","content":"再见了，两年..","id":"403778","img":"http:\/\/thirdqq.ql
ogo.cn\/g?b=sdk&k=6bulKMUwpGwXyxe83sc5cA&s=140&t=1557434834","nickname":"","like_num":"0"},{"user_name":null,"use
r_img":null,"time":"01月01日 08:00","content":"最后一次试探了","id":"403778","img":"http:\/\/thirdqq.qlogo.cn\/g?
b=sdk&k=S46FJb5urHHRn8gIiafqskQ&s=140&t=1555549008","nickname":"","like_num":"0"},{"user_name":null,"user_img":nu
ll,"time":"01月01日 08:00","content":"没能说出我爱你怕更失望","id":"403778","img":"http:\/\/thirdqq.qlogo.cn\/g?b
=sdk&k=e7JyHtTXaWhUCZFZQJk7HA&s=140&t=1625329904","nickname":"       ","like_num":"0"}]}
```

探索数据真面目

请你切换到 Preview，查看**第 1 页评论**对应的 **pl_list.html** 请求的预览。



如果已经关闭了开发者工具，可以参照下面的步骤重新打开。

1）右键【检查】打开开发者工具；

2）点击 Network 选项卡；

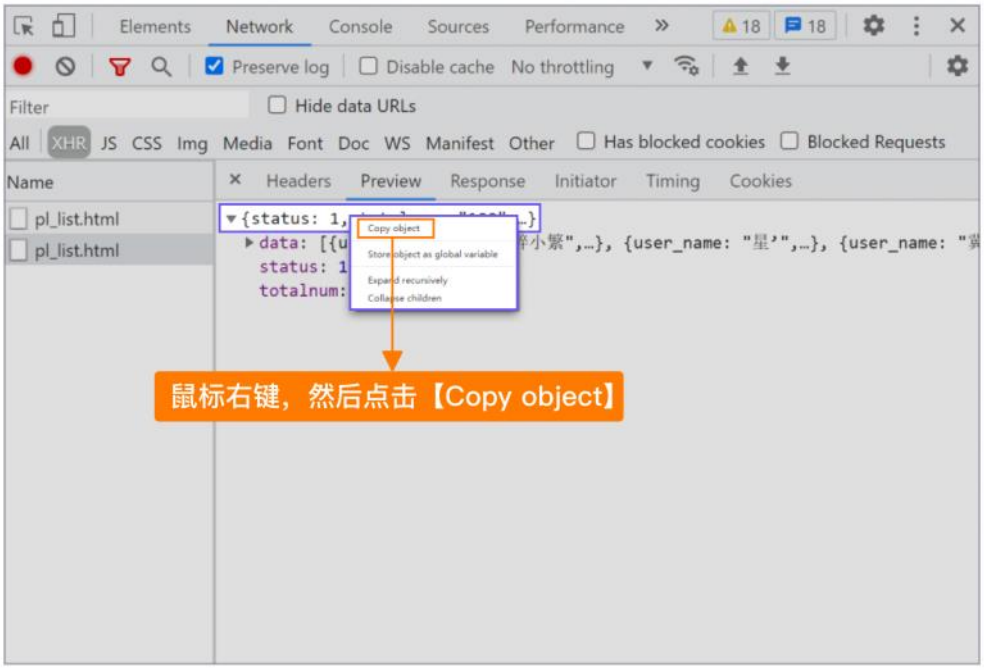
3）点击 XHR 筛选 xhr 请求；

4）点击【清空】按钮，然后刷新页面；

5）打开 xhr 请求，查看 Preview。

将鼠标放在第一行 {"status": 1,"totalnum": "100",…}，右键并选择【Copy object】，复制到本地。

（该数据为截图时的数据，数据会更新，以网页中观察到的为准）



然后就能看到**第 1 页评论**信息的数据结构 。

（本课所有图片中出现的数据为截图时的数据，数据会更新，以网页中观察到的为准）

经过观察，可以看到数据中大体有两种结构：

一种是字典结构，用大括号 { } 围起来，以其中一个结构为例进行高亮 。

另一种是列表结构，用方括号 [ ] 围起来（键 "data" 的值） 。

因为列表结构有很多行，建议你向下滑动查看。

因此，可以得出这样的结论，我们想要的评论区信息是通过类似于字典嵌套列表，列表再嵌套字典的结构进行保存的。

注：整个文本内容很多，为了更清楚地展示其数据结构，只选择三条评论信息进行展示

但是你也看到了，我一直在用字典结构、列表结构，难道刚刚分析的数据不是嵌套字典/列表吗？
试着看一下响应内容的数据类型吧。请你直接运行下面的代码，然后查看终端结果。

```
# 查看响应内容的数据类型
# 导入模块
import requests

# 设置评论区请求链接
comment_url = 'https://music.facode.cn/index.php/Home/Index/pl_list.html'

# 设置第 1 页表单提交数据
data = {
    'voice_id': 403778,
    'info': 1,
    'page': 0
}
# 发起请求，获取网页内容
comment_res = requests.post(comment_url, data)

# 查看响应内容的格式
print(type(comment_res.text))
print(comment_res.text)
```

```
<class 'str'>
{"status":1,"totalnum":"179","data":[{"user_name":"Virgil",
"user_img":"\/Public\/jquerylogin\/kodinger.jpg","time":"12月04日 19:44","content":"只因
","img":null,"nickname":null},{"user_name":"chenql","user_img":"\/Public\/jquerylogin\/
"time":"11月27日 00:42","content":"我来学习meeting20221016","img":null,"nickname":null},
{"user_name":"fgh","user_img":"\/Public\/jquerylogin\/kodinger.jpg","time":"11月27日 00:
"content":"你说把爱渐渐放下会走更远\n\n又何必去改变 已错过的时间\n\n你用你的指尖阻止我说再见\n\n想象
去之前\n\n你说把爱渐渐放下会走更远\n\n或许命运的签 只让我们遇见\n\n只让我们相恋这一季的秋天\n\n飘落后
```

```
#查看响应内容的数据类型
#导入模块
importrequests

#设置评论区请求链接
comment_url='https://music.facode.cn/index.php/Home/Index/pl_list.html'

#设置第1页表单提交数据
data={
'voice_id':403778,
'info':1,
'page':0
}
#发起请求，获取网页内容
comment_res=requests.post(comment_url,data)

#查看响应内容的格式
print(type(comment_res.text))
print(comment_res.text)
```

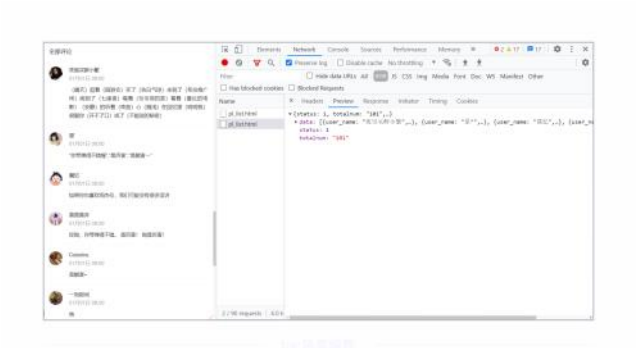
打印 type() 返回 <class 'str'>，说明响应内容是字符串，而非字典。

再严格一点说，该响应内容其实是 **JSON 格式的文本。**

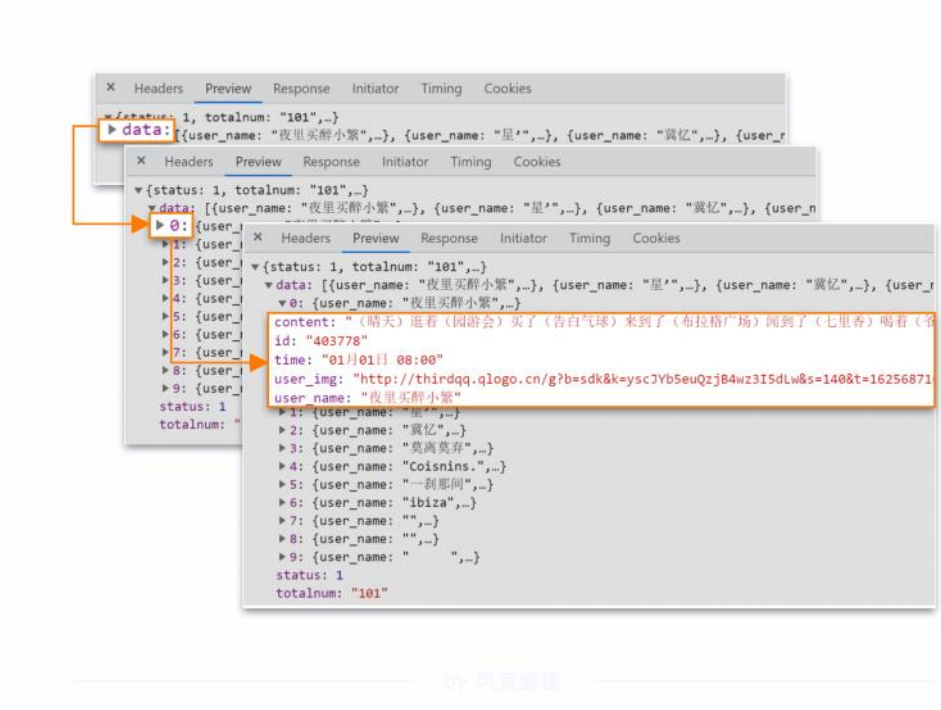
3.3 JSON

JSON 是一种轻量级的数据交换方式，占用字符数量极少，特别适合用在互联网传递中。
目前很多编程语言都支持 JSON 格式的文本，当然也包括我们课程中用到的 Python 。
经过前面的探索，我们可以简单归纳一下 JSON 文本的数据结构：
1）大括号{ }和其中的数据组成 {key1: value1, key2: value2, key3: value3...} 的字典结构；
2）方括号[ ]和其中的数据组成 [元素 1, 元素 2, 元素 3...] 的列表结构；
JSON 由字典结构和列表结构自由组合而成，多次嵌套后，依然结构清晰，是数据交换的极佳方式。
正因如此，当我们从 JSON 文本中提取数据时，需要重点关注它的嵌套方式。
我们一般通过 Preview 观察 JSON 文本的嵌套方式，下面就来看看评论区信息有怎样的结构。

请你再次回到**第 1 页评论**对应 **pl_list.html** 请求的 Preview。

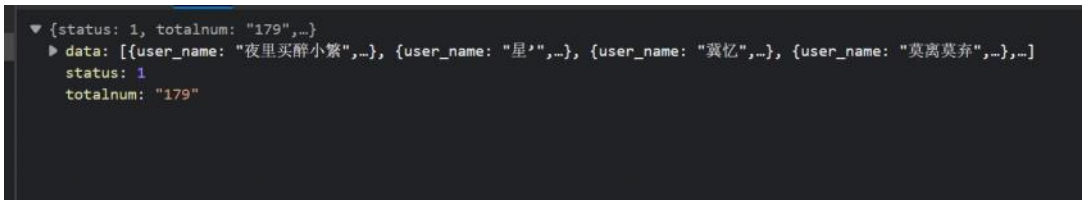


可以参照下面的步骤操作：
1）右键【检查】打开开发者工具；
2）点击 Network 选项卡；
3）点击 XHR 筛选 xhr 请求；
4）点击【清空】按钮，然后刷新页面；
5）打开 xhr 请求，查看 Preview。
点击箭头可以展开/收起相应内容，请你逐个展开 data、0，找到第一条评论的信息 。

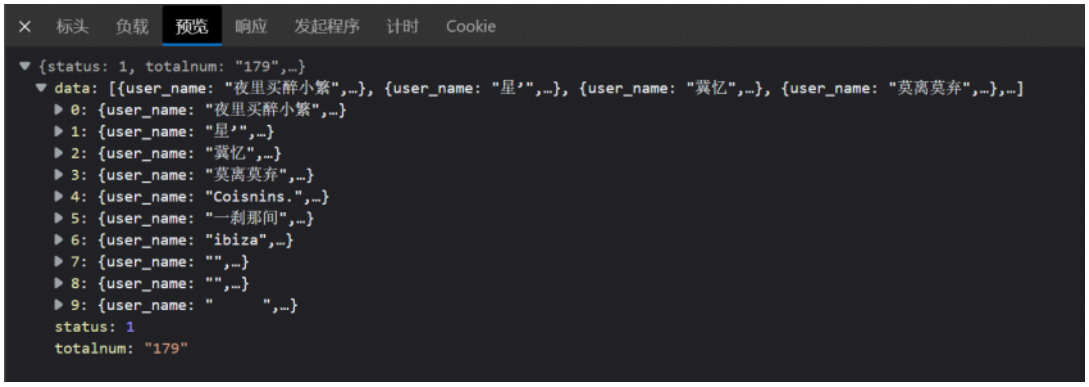


展开数据的过程中，会看到部分内容是紫色的，部分内容是红色的，可能还有一些内容是蓝色的 。
这些内容是浏览器对原始的 JSON 格式文本处理后的结果：
如果数据是**字典结构**，浏览器会将每一个键值对分行展示，并按照键的英文首字母重新排列；
如果数据是**列表结构**，浏览器会将各个列表元素按照索引的顺序分行展示，并把元素对应的索引设置为键，元素设置为值。

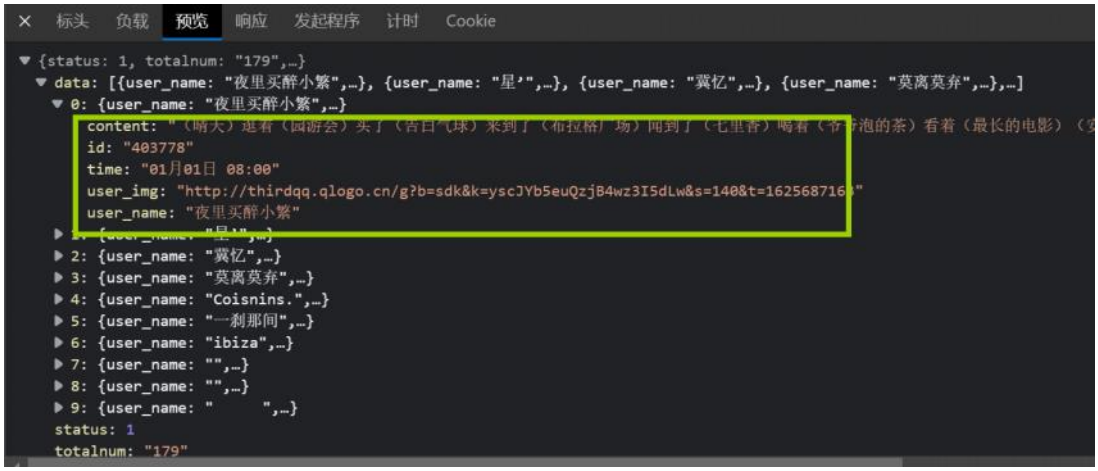
这么说有点抽象，我们来看看实际情况是怎样的。
（本课所有图片中出现的数据为截图时的数据，数据会更新，以网页中观察到的为准）
先来看第一行，整个数据的最外层结构：{"status": 1,"totalnum": "101",…}，（省略号处省略了很多数据）。
由于最外层是**字典结构**，所以浏览器将每一个键值对分行展示，并按照键的英文首字母重新排列，从上到下依次是 data、status、totalnum。



键 "data" 对应的值是一个**列表结构**，由 10 个字典结构的元素组成，因此浏览器就将这 10 个元素按照索引的顺序分行展示，并把元素的对应的索引设置为键，元素设置为值。



可以看到，每个**字典结构**的元素是一条评论信息：{'user_name': '', 'user_img': '', 'time': '', 'content': '', 'id': ''}，键是评论信息的字段，值是字段对应的内容。
以我们看到的第 1 条评论为例，浏览器就处理为这样的结构，从上到下依次是content、id、time、user_img、user_name：



其他的评论也会被处理为同样的结构，你可以展开看看。
之后，只要面对 JSON 格式文本，我们就可以直接在 Preview 中观察和分析。
知道了这些之后，相信你更关注的还是如何从 **JSON 格式文本中提取数据**。
JSON 格式文本不能直接提取，需要转换为字典类型才行。
我们可以用 Response.json() 方法将响应的 JSON 格式文本转换为字典类型。
请你直接运行 **【knowledge2.py】** 中的代码 。
看看返回的数据类型是否为字典。
打印 type() 返回 <class 'dict'>，说明数据成功被转换为字典 。
我们想要的数据已经一步一步处理为熟悉的样子——字典。

下面对翻页分析

请你切换到全部评论区第 2 页对应的 **pl_list.html** 请求，并找到它的请求体数据【Form Data】。

可以和第 1 页的【Form Data】对比一下。

仔细对比之后，发现 page 的值不同，第 1 页 page 的值是 1，第 2 页 page 的值是 2。
我们试着修改一下键 'page' 的值，看看能否得到全部评论区第 2 页内容。
请你移步到 **【main2.py】** 的第 11 行，修改 'page' 的值为 2，看看终端结果有什么变化 。
如此一来，我们就得到了全部评论区第 2 页的内容。
（本课所有图片中出现的数据为截图时的数据，数据会更新，以网页中观察到的为准）

也就是说，修改请求体中 page 的值，就能得到不同页的评论。那么，修改 voice_id、info 的值是否也能得到其他的结果呢？
答案是肯定的，你可以自行探索，这里就不详细展开了。

4. 爬取所有歌曲评论

通过观察请求体【Form Data】的规律，相信你已经能爬取任意一页的评论信息。

下面请完成爬取

方法2

```
# 提取 所有评论页数据 方法2

import csv
import requests
from bs4 import BeautifulSoup

# 设置歌曲详情页请求链接
page_url = 'https://music.facode.cn/index.php/Home/Index/voice_details/id/403778'
# 设置评论区请求链接
comment_url = 'https://music.facode.cn/index.php/Home/Index/pl_list.html'

# 发起请求，获取网页内容
show_res = requests.get(page_url)
# 解析数据
soup = BeautifulSoup(show_res.text, 'html.parser')
```

```
# 提取评论总数所在的标签
comment_tag = soup.find('span', class_='c_tx_thin part__tit_desc')
# 提取评论总数
comment_num = int(comment_tag.text[1:-3])

# 评论总数除以 10，取商并赋值给页数
page_num = comment_num // 10

# 如果评论总数除以 10，余数不为零，则页数 +1
if comment_num % 10 != 0:
    page_num += 1

# 设置列表，用以存储每条评论的信息
data_list = []

# 循环评论总页数的次数，从第1页开始
for page in range(1, page_num + 1):

    # 设置表单提交数据
    data = {
        'voice_id': '403778',
        'info': 2,
        'page': page
    }

    # 发起请求，获取网页内容
    comment_res = requests.post(comment_url, data=data)

    # 将json格式的文本转换为字典
    json_data = comment_res.json()

    # 循环获取每一条评论信息
    for comment in json_data['data']:
        # 将评论的信息添加到字典中
        comment_dict = {
            '用户名': comment['user_name'],
            '评论时间': comment['time'],
            '评论内容': comment['content']
        }
        print(comment_dict)

        # 存储每条评论的信息
        data_list.append(comment_dict)

# 新建 csv 文件，用以存储评论的信息
with open('../所有评论.csv', 'w', encoding='utf-8-sig') as f:
    # 将文件对象转换成 DictWriter 对象
    f_csv = csv.DictWriter(f, fieldnames=['用户名', '评论时间', '评论内容'])
    # 写入表头与数据
    f_csv.writeheader()
    f_csv.writerows(data_list)
```

或者体验下 开始的 体验代码

5.2 知识归纳与总结

本节课主要学习了以下几个知识点：

1）静态网页和动态网页

静态网页，所见即所爬；**动态网页**，所见非所爬。

随着我们面对的网页越来越多样，我们定位内容的方法也需要更加严谨。

你可以先在 Elements 选项卡中定位元素，然后切换到 Network 选项卡，在 HTML 文档所在请求的 Preview 中验证：

a. 如果能找到，则可以直接根据网页链接爬取目标内容；

b. 否则，可以尝试用 XHR 筛选 xhr 请求，并进一步在其 Preview / Response 中进行定位，然后再进行爬取。

2）AJAX 和 xhr

AJAX 技术是一套综合了多项技术的网页开发技术，其请求类型是 xhr。

爬取动态内容，需要熟练查看 xhr 请求，以下步骤供你参考：

a. 右键【检查】打开开发者工具；

b. 点击 Network 选项卡；

c. 点击 XHR 筛选 xhr 请求；

d. 点击【清空】按钮，然后刷新页面；

e. 打开 xhr 请求，查看 Headers、Preview、Response。

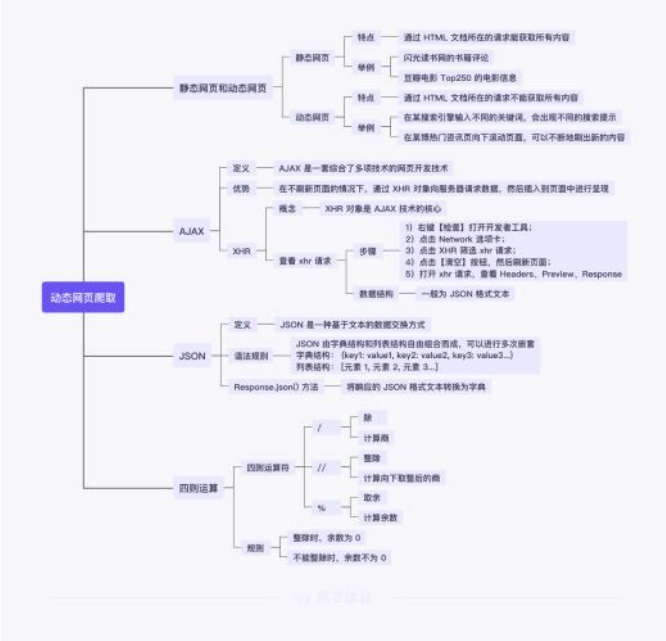
3）JSON

AJAX 技术一般通过 JSON 格式的文本传递数据。

使用 Response.json() 方法 可以将响应的 JSON 格式文本转换为字典。

我们一般在 Preview 中查看 JSON 文本的结构，然后根据我们分析的字典/列表结构提取数据。

以下是我们的知识点总结图：



今天的课程到这里就结束了，我们下节课见！

作业：

爬取最新评论区的内容