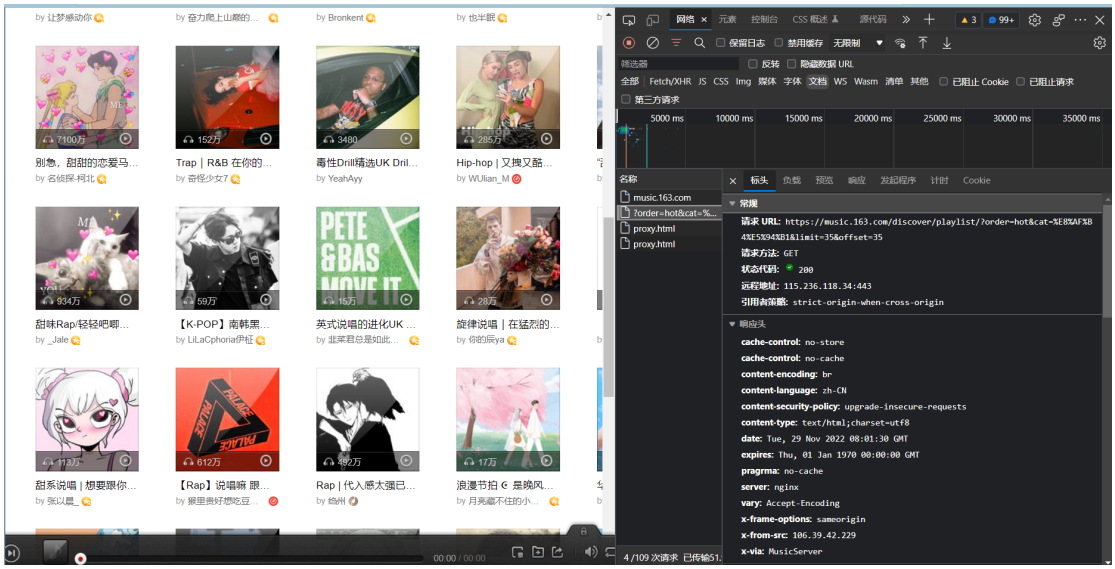
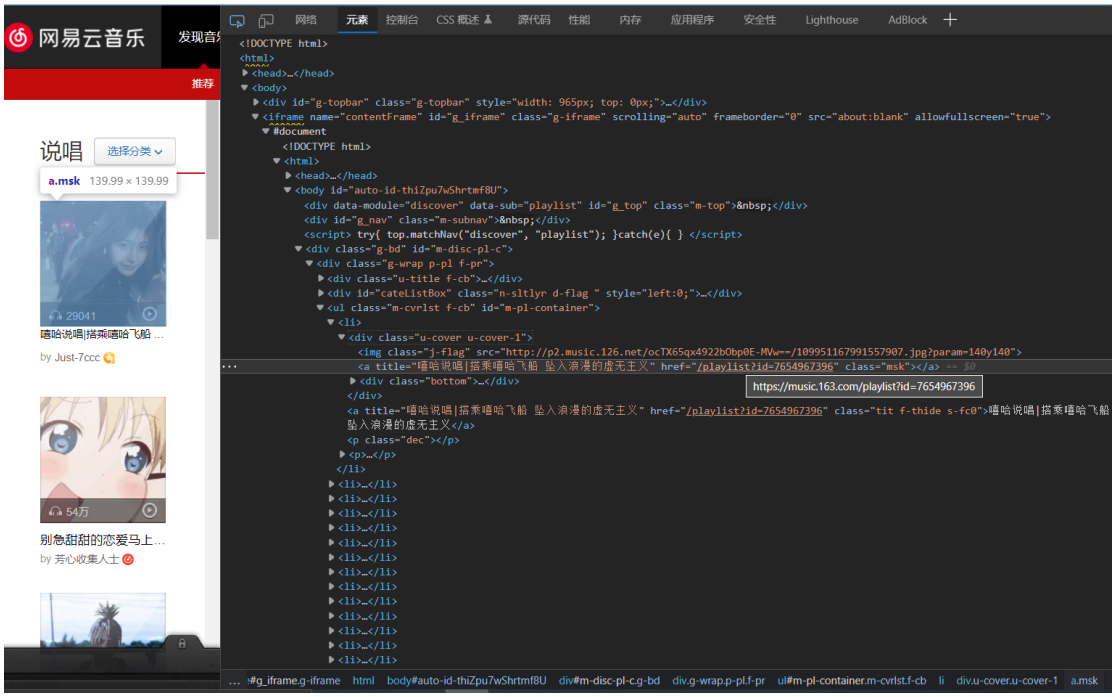
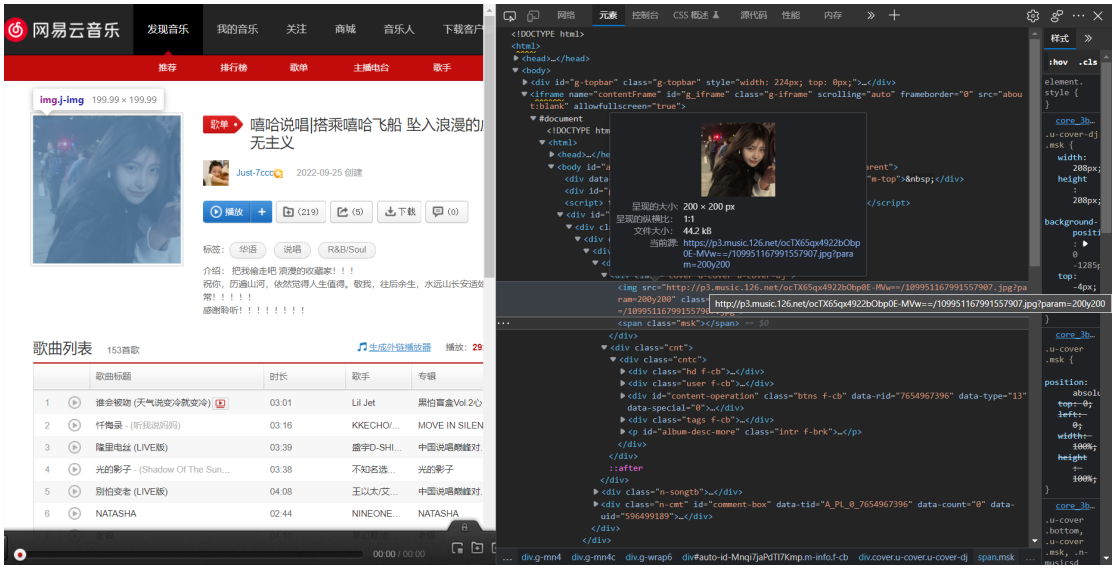
# 1 网页解析



解析网页，在【文档】类型中找到要爬取的目标 URL。



解析元素，找到目标歌单的 URL 所在位置




解析子 URL，找到想要爬取的内容的位置。

# 2 代码

## 2.1 准备工作

```python
import pandas as pd
import requests, time, csv, random, re, pickle
from lxml import etree
from threading import Thread, Lock
from queue import Queue


# 将网页参数设置为全局变量以便使用
agents = ["Mozilla/5.0 (Android; Mobile; rv:14.0) Gecko/14.0 Firefox/14.0",  # 设置多个Agent应对反爬
          "Mozilla/5.0 (Android; Tablet; rv:14.0) Gecko/14.0 Firefox/14.0",
          "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:21.0) Gecko/20100101 Firefox/21.0",
          "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:21.0) Gecko/20130331 Firefox/21.0",
          "Mozilla/5.0 (Windows NT 6.2; WOW64; rv:21.0) Gecko/20100101 Firefox/21.0",
          "Mozilla/5.0 (Linux; Android 4.1.1; Nexus 7 Build/JRO03D) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.166  Safari/53
          "Mozilla/5.0 (Linux; Android 4.0.4; Galaxy Nexus Build/IMM76B) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.133 Mobil
          "Mozilla/5.0 (Linux; Android 4.1.2; Nexus 7 Build/JZ054K) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.166 Safari/535
          "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/27.0.1453.93 Safari/537.36",
          "Mozilla/5.0 (compatible; WOW64; MSIE 10.0; Windows NT 6.2)",
          "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)",
          "Opera/9.80 (Windows NT 6.1; WOW64; U; en) Presto/2.10.229 Version/11.62"]
headers = {
        'Referer': 'https://music.163.com/',
        'User-Agent': random.choice(agents),
        }


def urls_extracting():
    pl_urls = []
    for page in range(0, 44):
        time.sleep(random.uniform(2, 5))  # 设置随机间隔时间反爬
        #print(f'---开始爬取第{page+1}页-->>>')
        url = f'https://music.163.com/discover/playlist/?order=hot&cat=%E8%AF%84%E5%94%B1&limit=35&offset={page*35}'
        response = requests.get(url=url, headers=headers).text  # 响应内容
        html = etree.HTML(response)  # 转换成XML
        playlist = html.xpath('//div[@class="u-cover u-cover-1"]/a[@class="msk"]/@href')  # 获取各歌单的子url列表
        for i in playlist:
            pl_url = 'https://music.163.com/' + i  # 歌单完整url
            pl_urls.append(pl_url)
        #print(f'>>>--第{page+1}页歌单url提取成功--√')
    with open('week12_pl_urls.csv', 'w', encoding = 'utf-8', newline = '') as f:
        for i in pl_urls:  # 保存pl_urls到本地方便使用
            csv.writer(f).writerow([i])

def write(row):  # 写入csv
    with open(r'week12_网易云歌单.csv', 'a', encoding = 'utf-8', newline = '') as f:
        csv.writer(f).writerow(row)
```

## 2.2 生产者尝试连接网页

```python
def produce(q):  # 生产者：网页请求
    with open('week12_pl_urls.csv', 'r', encoding = 'utf-8', newline = '') as f:
        pl_urls = [i[0] for i in csv.reader(f)]
    for i in range(0, len(pl_urls)):
        #time.sleep(random.uniform(2, 5))  # 设置随机间隔时间反爬
        print(f'---开始爬取第{i+1}个歌单-->>>')
        pl_response = requests.get(url=pl_urls[i], headers=headers).text
        q.put([i, pl_response])
```

## 2.3 消费者解析网页内容并写入 csv

```python
def consume(q, lock):  # 消费者：网页提取
    while True:
        item = q.get()
        if item is None:
            break
        i, pl_response = item
        pl_html = etree.HTML(pl_response)
        title = pl_html.xpath('//h2[@class="f-ff2 f-brk"]')[0].text
        cover = pl_html.xpath('//img[@class="j-img"]/@src')[0]
        with open('week12_网易云歌单封面/'+f'{i+1}'+'.jpg', 'wb') as image:
            image.write(requests.get(cover).content)
        creator_name = pl_html.xpath('//a[@class="s-fc7"]')[0].text
        creator_id = re.search('(?<=(?:)id=).*', pl_html.xpath('//a[@class="s-fc7"]/@href')[0])
        intro_result = pl_html.xpath('//p[@class="intr f-brk"]')
        if intro_result == []:
            introduction = ''
        else:
            introduction = intro_result[0].xpath('string(.)')[4:]  # 删去'介绍\n'
        num_songs = pl_html.xpath('//span[@id="playlist-track-count"]')[0].text
        num_play = pl_html.xpath('//strong[@id="play-count"]')[0].text
        num_add = pl_html.xpath('//a[@class="u-btni u-btni-fav "]/i/text()')[0][1:-1]
        num_share = pl_html.xpath('//a[@class="u-btni u-btni-share "]/i/text()')[0][1:-1]
        num_comment = pl_html.xpath('//span[@id="cnt_comment_count"]/text()')[0]
        row = [title, cover, creator_name, creator_id, introduction,
               num_songs, num_play, num_add, num_share, num_comment]
        lock.acquire()  # 加锁以免出错
        try:
            write(row)
        finally:
            lock.release()
        print(f'>>>--第{i+1}个歌单写入成功--√')
        q.task_done()
```

## 2.4 主函数

```python
if __name__=='__main__':
    #urls_extracting()
    with open(r'week12_网易云歌单.csv', 'a', encoding = 'utf-8', newline = '') as f:
        csv.writer(f).writerow(['title', 'cover', 'creator_name', 'creator_id', 'introduction',
                                'num_songs', 'num_play', 'num_add', 'num_share', 'num_comment'])
    n = 3  # 消费者线程数
    lock = Lock()
    q = Queue()
    threads = []
    for i in range(n):  # 创建并启动消费者进程
        t = Thread(target = consume, args = (q, lock))
        threads.append(t)
        t.start()
    produce(q)  # 启动生产者线程
    q.join()
    for i in range(n):  # 停止消费者进程
        q.put(None)
    for t in threads:
        t.join()
```

## 3 运行结果

```
---开始爬取第1511个歌单-->>>
---开始爬取第1512个歌单-->>>
>>>--第1510个歌单写入成功--√
>>>--第1511个歌单写入成功--√
---开始爬取第1513个歌单-->>>
>>>--第1512个歌单写入成功--√
---开始爬取第1514个歌单-->>>
---开始爬取第1515个歌单-->>>
>>>--第1513个歌单写入成功--√
>>>--第1514个歌单写入成功--√
---开始爬取第1516个歌单-->>>
>>>--第1515个歌单写入成功--√
>>>--第1516个歌单写入成功--√
```


4 爬虫程序往往需要稳定运行较长的时间，因此如果你的程序突然中断或异常（比如网络或被封），如何能够快速从断点重启？

记录程序运行点，比如记录当前爬虫的页数，在程序突然中断时，从中断的页数重新爬取。