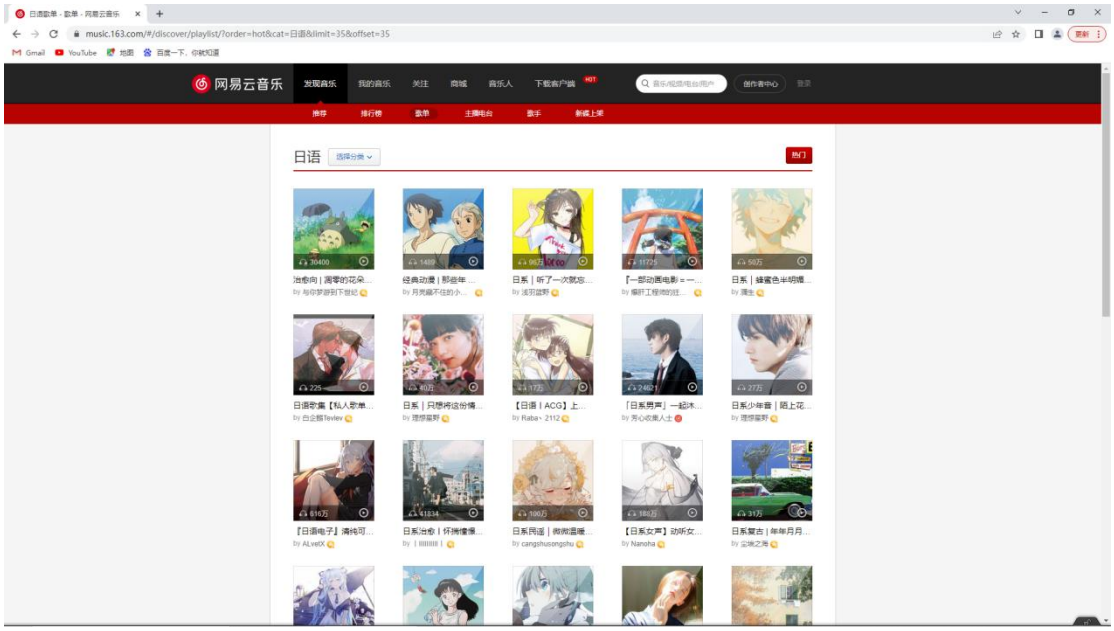
程设第十一次作业

20377383 樊思涵

准备工作：

使用 chrome 打开网易云指定分类歌单



观察日语分类第二页的 url 地址

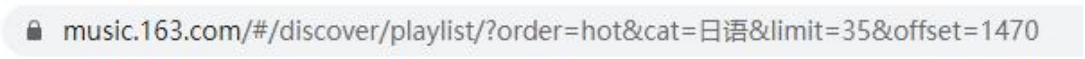
简单分析一下

order 为排序方式

cat 为种类

limit 为一页的歌曲数量

offset 为当前页数（35 是第二页）



总页数源码位置

```
<a href="/discover/playlist/?order=hot&cat=%E6%97%A5%E8%AF%AD&limit=35&offset=1470" class="zpgi">43</a> == $0
<a href="/discover/playlist/?order=hot&cat=%E6%97%A5%E8%AF%AD&limit=35&offset=35" class="zbtn znxt">下一页</a>
</div>
```

对任意歌单右键点击检查查看对应源码位置

可以在 Elements 中发现部分我们需要的信息

包括：

需要保存的封面图片，根据 img 节点获取 url 保存图片

歌单标题

```
<a title="[日系私人订制] 最懂你的日系推荐 每日更新35首" href="/playlist?id=2829896389" class="msk"></a> == $0
```

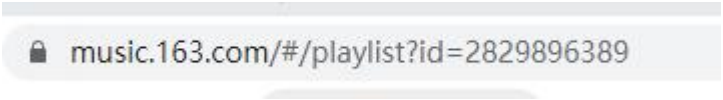
创建者 id 与昵称

```
<a title="云音乐官方歌单" href="/user/home?id=1463586082" class="nm nm-icn f-thide s-fc3">云音乐官方歌单</a> == $0
```

播放量

```
k"></a>
▼<div class="bottom">
  <a class="icon-play f-fr" title="播放" href="javascript:;" data-res-type="13" data-res-id="2829896389" data-res-action="play"></a>
  <span class="icon-headset"></span>
  <span class="nb">5631万</span>
```

剩下的信息需要我们打开新的对应歌单的 url
先看看歌单的 url 形式



发现只需要得知某歌单的形式就可以轻松获得对应 url
（后续发现不能加’/#’）
再来分析分析歌单 url 中的内容
收藏数

```
class="u-btni u-btni-fav " href="javascript:;">
    <i>(62万)</i> == $0
```

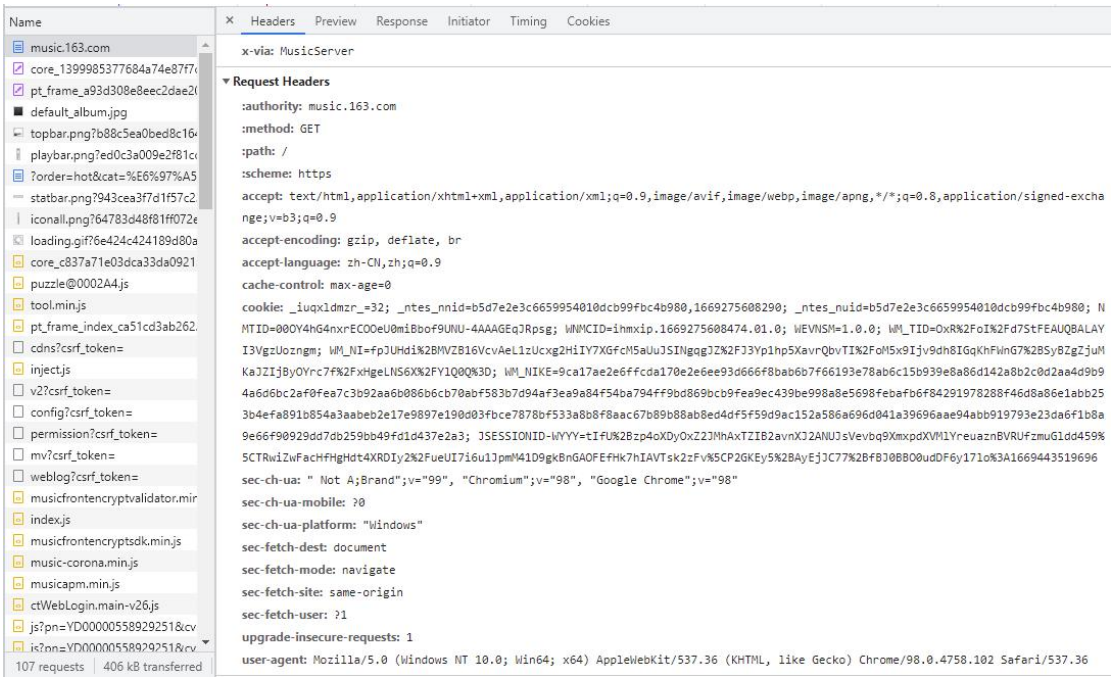
介绍

```
<b>介绍：</b>
" 发现更多你爱的日系好音乐"
<br>
" 就在你专属的日系私人订制"
<br>
" 每早更新，收藏订阅哦～"
<br>
```

转发、下载、评论数

```
▼<a data-res-id="2829896389" data-res-type="13" data-count="8177" data-res-action="share" data-res-name="[日系私人订制] 最懂你的日系推荐 每日更新35首" data-res-author="云音乐官方歌单" data-res-authors data-res-pic="https://p1.music.126.net/mgaaUpnJ3Bs_CwvjAlv3XA==/109951165545379929.jpg" class="u-btni u-btni-share " href="javascript:;">
   <i>(8177)</i>
 </a>
▼<a data-res-id="2829896389" data-res-type="13" data-res-action="download" class="u-btni u-btni-dl " href="javascript:;">
   <i>下载</i>
 </a>
▼<a data-res-action="comment" href="javascript:;" class="u-btni u-btni-cmmt ">
 ▼<i>
    "("
    <span id="cnt_comment_count">10117</span>
    ")"
  </i>
 </a>
```

准备工作全部完成，下面开始正式实现代码
在 Network 中获取 Headers 中的内容



使用 requests 库获得对应 html

```
def get_page(url):
    # 构造请求头部
    headers = {
        'Referer': 'http://music.163.com/',
        'Host': 'music.163.com',
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.75 Safari/537.36',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8'
        }
    # 发送请求，获得响应
    response = requests.get(url=url,headers=headers)
    response.raise_for_status
    response.encoding = response.apparent_encoding
    # 获取响应内容
    html = response.text
    return html
```

先获取一下一共有多少页

```
cat = urllib.parse.quote('日语')
#先在主页面获取总页数
home_page = 0
url='https://music.163.com/discover/playlist/?order=hot&cat=' + cat + '&limit=35'+'&offset='
html = get_page(url+str(35*home_page))
soup=BeautifulSoup(html,'html.parser')
txt_page = soup.find('div',{'class':"u-page"})#能够获取页数的信息
max_page = txt_page.text.split('\n')[-3] #获得该歌单一共有多少页
```

成功获取有用信息

```
['', '上一页', '1', '2', '3', '4', '5', '6', '7', '8', '...', '43', '下一页', '']
```

构建 get_inf1 函数

```
def get_inf1(page):
    """
    歌单总页面爬取内容
    最后将获取的所有歌单的
    标题、播放量、地址形式id、创建者昵称、id
    添加至全局变量列表
    并将封面图片保存在本地
    """
```

```
    url='https://music.163.com/discover/playlist/?order=hot&cat=' + cat + '&limit=35'+'&offset='
    html = get_html(url+str(35*(page-1)),headers1)
    soup=BeautifulSoup(html,'html.parser')
    c=soup.find_all('li')
    folder_path = 'C:\\Users\\LF\\Desktop\\网易云歌单封面\\第'+str(page)+'页封面图片'
    os.makedirs(folder_path, exist_ok=True)
    os.chdir(folder_path)  # 切换路径至上面创建的文件夹
    img_name = 1  # 用来给图片命名
    for item in c:
        try:
            nickname = item.find('a',{'class':"nm nm-icn f-thide s-fc3"})
            name_url=item.find('a',{'class':"tit f-thide s-fc0"})                  #URL信息
            number=int(item.find('span',{'class':'nb'}).text.replace('万','0000')) #播放量的信息
            list1=[
                   name_url['href'],                          #url形式的id信息
                   number,                                    #播放量
                   name_url['title'].replace(u'\xa0', u' '),  #歌单名称
                   nickname['title'],                         #创建者昵称
                   int(name_url['href'].split('=')[-1])]      #id
            inf_list.append(list1)
            img = item.find('img')
            if img is not None:
                url = img['src']
                if url.count('jpg') > 0:
                    save_img(url, str(img_name) + '.jpg')
                elif url.count('png') > 0:
                    save_img(url, str(img_name) + '.png')
                img_name += 1
        except:
            continue
```

从这里获取的信息可以再次从具体的歌单 url 中获得信息

```
def get_inf2(url_id):
    """
    进入具体的某个歌单的url获取信息
    返回id对应歌单的播放了、收藏数、
    创建者昵称、分享数、评论数、歌曲数量、
    歌单名称以及歌单介绍
    添加至全局变量列表
    """
```

再次获取 html
并构建 BeautifulSoup 类型

```
    #print(url_id)
    singleurl='https://music.163.com'+url_id
    singletext=get_html(singleurl,headers=headers2)
    #print(singletext)
    soup=BeautifulSoup(singletext,'html.parser')
```

```
    try:
        play_count=eval(soup.find('strong',{'class':'s-fc6'}).text)                          #播放量
        fav=soup.find('a',{'class':'u-btni u-btni-fav'}).i.text.strip('(').strip(')')        #收藏数
        nickname = soup.find('a',{'class':'s-fc7'}).text                                     #创建者昵称
        if('万') in fav:
            fav=eval(fav.replace('万','0000'))
        share=eval(soup.find('a',{'class':'u-btni u-btni-share'}).i.text.strip('(').strip(')'))   #分享数
        comment=eval(soup.find('a',{'data-res-action':'comment'}).i.span.text)               #评论数
        length=eval(soup.find('span',{'id':'playlist-track-count'}).text)                    #歌曲数量
        name=soup.find('h2',{"class":'f-ff2 f-brk'}).text.replace(u'\xa0', u' ')             #歌单名称
        tags=soup.find_all('a',{'class':'u-tag'})
        introduction = soup.find('p',{"class":'intr f-brk'}).text.split('\n')
        intr=''
        for i in range(1,len(introduction)):
            intr += introduction[i]
        #print(intr)
        p=len(tags)
        tag1='NA'
        tag2='NA'
        tag3='NA'
        if p>=1:
            tag1=tags[0].text.replace(u'\xa0', u' ')
        if p>=2:
            tag2=tags[1].text.replace(u'\xa0', u' ')
        if p==3:
            tag3=tags[2].text.replace(u'\xa0', u' ')
        list1=[name,nickname,play_count,fav,share,comment,length,tag1,tag2,tag3,intr]
        inf_plus_list.append(list1)
        print('解析歌单成功')
    except:
        print('解析歌单失败')
        return
```

尝试使用生产者-消费者模型
建立对应生产消费线程

```
con=Condition()

inf_plus_list=[]
q=list()
def check_q():
    return len(q)

def consume():
    with con:
        con.wait_for(check_q)
        id_lis = q.pop()
        for id in id_lis:
            get_inf2(id[0])

def produce(page,cat):
    with con:
        get_inf1(page,cat)
        con.notify_all()
```

建立线程

```
    for page in range(1,max_page+1):
        c=Thread(target=consume,name='c-'+str(page))
        #break   #在测试阶段只爬第一页
    plist=[]
    for page in range(1,max_page+1):
        p=Thread(target=produce,name='p-'+str(page),args=(page,cat))
        #break
    for c in clist:
        c.start()
    for p in plist:
        p.start()
```

运行后发现很容易出现莫名的阻塞现象，所以我改用 ThreadPoolExecutor 进行多线程处理

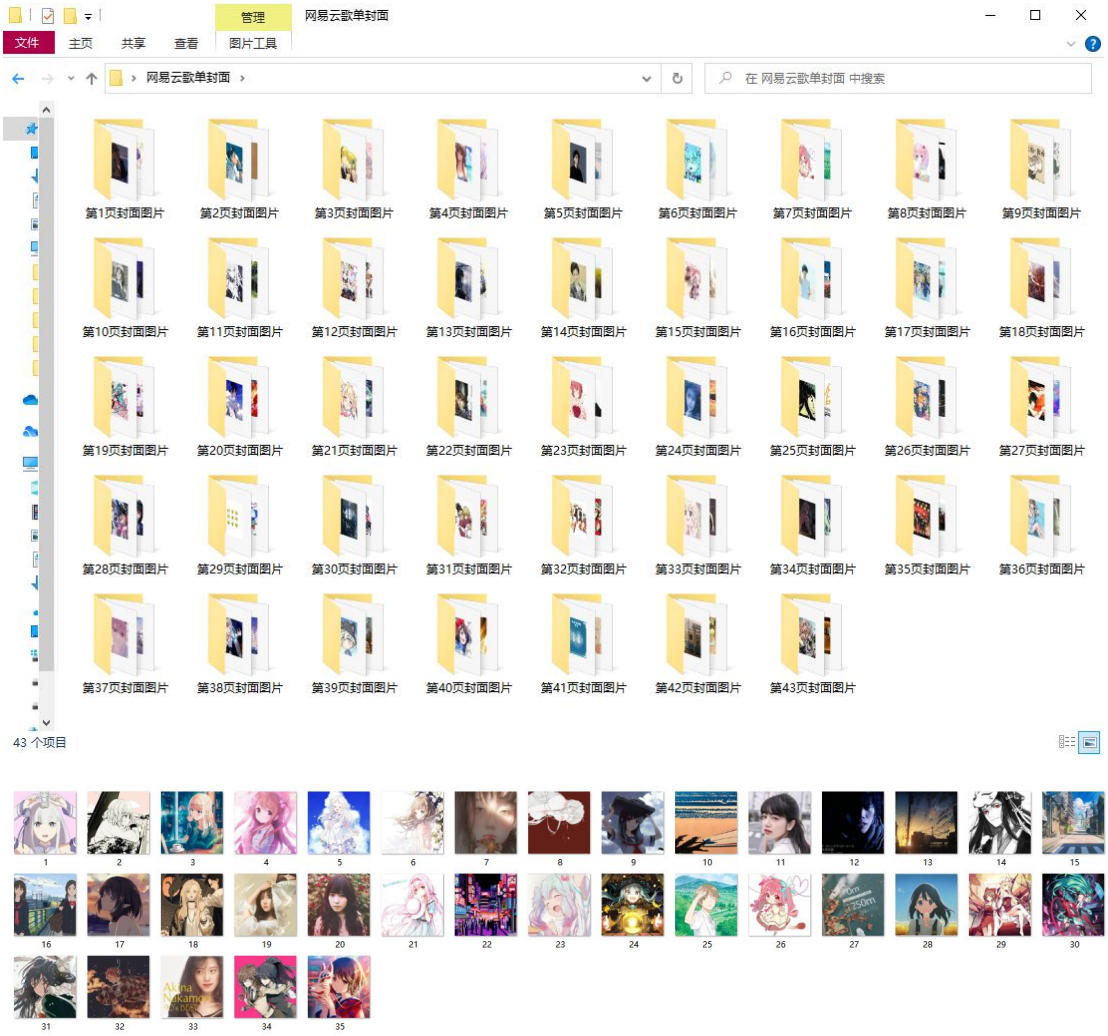
```
140     with concurrent.futures.ThreadPoolExecutor() as executor:
141         for page in range(1,max_page+1):
142             executor.submit(get_inf1,page)
143             break   #在测试阶段只爬第一页
144     pd_inf=pd.DataFrame(inf_list)
145     url_lis=list(pd_inf[0])
146     #print(url_lis)
147     #for i in url_lis:
148     #    get_inf2(i)
149     #map(get_inf2,url_lis)
150     with concurrent.futures.ThreadPoolExecutor() as executor: #多线程处理
151         for id in url_lis:
152             executor.submit(get_inf2,id)
```

最终信息处理与输出

```
159     pd_inf_plus=pd.DataFrame(inf_plus_list)
160     title_list=['名称','创建者昵称','播放量','收藏数','分享次数','评论数','歌单数量','tag1','tag2','tag3','介绍']
161     c=pd.Series(title_list)
162     pd_inf_plus.columns=c
163     pd_inf_plus.to_csv(r'C:\Users\LF\Desktop\{}.csv'.format(song_type))
```

最终结果展示

```
PS E:\code\py_code> python -u "e:\code\py_code\week12\week12.py"
1.jpg 文件保存成功!
2.jpg 文件保存成功!
3.jpg 文件保存成功!
4.jpg 文件保存成功!
5.jpg 文件保存成功!
6.jpg 文件保存成功!
7.jpg 文件保存成功!
8.jpg 文件保存成功!
9.jpg 文件保存成功!
10.jpg 文件保存成功!
11.jpg 文件保存成功!
12.jpg 文件保存成功!
13.jpg 文件保存成功!
14.jpg 文件保存成功!
15.jpg 文件保存成功!
16.jpg 文件保存成功!
17.jpg 文件保存成功!
18.jpg 文件保存成功!
19.jpg 文件保存成功!
20.jpg 文件保存成功!
21.jpg 文件保存成功!
22.jpg 文件保存成功!
23.jpg 文件保存成功!
```




```
解析歌单成功
解析歌单成功
解析歌单成功
解析歌单成功
解析歌单成功
解析歌单成功
解析歌单成功解析歌单成功
解析歌单成功
解析歌单失败

解析歌单成功
解析歌单成功
解析歌单成功解析歌单成功
解析歌单失败

解析歌单成功
解析歌单成功
解析歌单成功解析歌单成功

解析歌单成功解析歌单成功
解析歌单成功

解析歌单成功解析歌单成功
解析歌单成功

PS E:\code\py_code>
```

一共爬了 1292 个歌单！
顺便试了试不用多线程

```
#for i in url_lis:
#    get_inf2(i)
#map(get_inf2,url_lis)
```

发现运行速度明显变慢

Ref.
爬虫相关内容简介
https://www.cnblogs.com/wsmrzx/p/9450462.html