第3章

爬取豆瓣电影简介

本章案例将介绍如何爬取豆瓣电影简介，以此帮助读者学习如何通过编写爬虫程序来批量地从互联网中获取信息。本案例中将借助两个第三方库——Requests 库和 BeautifulSoup 库。通过 Requests 库获取相关的网页信息，通过 BeautifulSoup 库解析大体框架信息的内容，并且将局部信息中最关键的内容提取出来。通过使用第三方库，读者可以实现定向网络爬取和网页解析的基本目标。

视频讲解

3.1 确定信息源

首先，需要明确要爬取哪些内容，并精确到需要爬取的网页。在本章中需要爬取电影的简介，因此选择信息较为全面的豆瓣电影。

进入“豆瓣电影”首页，如图 3-1 所示，可以看到各种推荐的电影。选择其中任意一部电影，进入电影简介页面，如图 3-2 所示。可以发现每部电影的简介页面都具有相似的结构，这为编写爬虫程序提供了极大的方便。这意味着只需要以某部电影的电影简介页面为模板编写爬虫程序，此后便可以运用到豆瓣网站中其他所有电影简介的页面。

图 3-1　豆瓣电影首页

图 3-2　豆瓣电影简介页面

3.2　获取网页信息

确定了需要爬取的网页后，需要如何获取这些信息呢？要知道网页是一个包含 HTML 标签的纯文本文件。可以通过使用 Requests 库的 request()方法向指定网址发送请求获得的文本文件。注意，在使用 Python 第三方库前需要在本机上安装第三方库，可以使用 pip 方式或者其他方式安装。

1. Requests 库的两个主要方法

下面先来简单介绍本实验需要用到的 Requests 库的两个主要方法。

(1) requests.request()方法。该方法的作用是构造一个请求，是支持其他方法的基础方法。

(2) requests.get()方法。该方法是本案例需要用到的一个获取网页的方法，会构造一个向服务器请求资源的 Request 对象，然后返回一个包含服务器资源的 Response 对象。其作用是获取 HTML 网页，对应于 HTTP 的 GET。调用该方法需要一些参数。

① requests.get(url, params=None, ** kwargs)。

② url：需要获取页面的 URL 链接。

③ params：可选参数，URL 中的额外参数，字典或字节流格式。

④ ** kwargs：12 个控制访问的参数。

2. Response 对象机器属性

Response 对象包含服务器放回的所有信息，也包含请求的 Request 信息。该对象具如下属性。

① status_code：HTTP 请求的返回状态，200 表示链接成功。只有链接成功的对象内容才是正确有效的，而对于链接失败的行为则产生异常的结果。这里可以使用 raise_for_status()方法。该方法会判断 status_code 是否等于 200，如果不等于则产生异常 requests.HTTPError。

② text：HTTP 相应内容的字符串形式，也即 URL 对应的页面内容。这部分也是爬虫

需要获得的最重要部分。

③ encoding：从 HTTP header 中猜测的响应内容的编码方式。选择正确的编码方式才能得到可读的正确信息内容。

④ apparent_encoding：从内容中分析出响应内容的编码方式。

学习完上面的基础知识，就可以利用 Requests 库编写程序来获取“豆瓣电影”首页的信息。获取“豆瓣电影”首页信息的程序如图 3-3 所示。

```
import requests

#豆瓣电影主页网址
url = "https://movie.douban.com"
#headers属于**kwargs可选参数之一
headers = {'user-agent':'Mozilla/5.0'}
try:
    #向豆瓣电影主页发出请求，返回的Response对象储存为r
    r = requests.get(url, headers = headers)
    #检查statu_code状态，若链接失败则抛出异常
    r.raise_for_status()
    #确定编码方式
    r.encoding = r.apparent_encoding
    #输出爬取的HTML文本
    print(r.text)
except:
    #爬取失败的处理
    print("爬取失败")
```

图 3-3　获取“豆瓣电影”首页信息的程序

在控制台里就能够看见“豆瓣电影”首页 HTTP 相应内容的字符串形式，如图 3-4 所示。

```
Console 1/A

    <div id="reviews" class="s" data-dstat-areaid="77" data-dstat-mode="click,expose">
        <div class="reviews-hd">
            <h2>最受欢迎的影评<span><a href="/review/best/">更多热门影评&raquo;</a></span><span><a href="/review/latest/">新片影评
&raquo;</a></span></h2>
        </div>
        <div class="reviews-bd">

    <div class="review ">
        <div class="review-hd">
            <a href="https://movie.douban.com/subject/4301043/?from=reviews">
                    <img class="lazy"
                        data-original="https://img9.doubanio.com/view/photo/s_ratio_poster/public/p1213203384.jpg"
                        src="https://img3.doubanio.com/f/movie/54a69da17972770faffce1f758c820ecb8e440af/pics/movie/grey.gif"
                        alt="传染病" />
            </a>
        </div>
        <div class="review-bd">
            <h3><a href="https://movie.douban.com/review/12228570/" class="">你原本只想回望八年前的非典，却无奈成了八年后新冠病毒的预言</
a></h3>
            <div class="review-meta">
                <a href="https://www.douban.com/people/210579110/">connie301</a> 评论
                <a href="https://movie.douban.com/subject/4301043/?from=reviews">《传染病》</a>

                    <span class="allstar50"></span>
            </div>
            <div class="review-content">
                影片公映于2011年，虽然取材于2003年中国的非典疫情，但这一年距非典结束已经八年，八年里经历过非典的那群人长大的长大、成家的成
家、衰老的衰老……曾经的举国之痛在各自的喜怒哀乐面前似乎无足轻重，这样一部...
                    <a href="https://movie.douban.com/review/12228570/">(全文)</a>
            </div>
```

图 3-4　控制台输出的部分内容

【试一试】 改变上述代码中的 URL 链接，尝试爬取你喜欢网站中的信息，看看返回的内容有什么区别。

3.3　解析信息内容

浏览器通过解析 HTML 文件，展示丰富的网页内容。HTML 是一种标识性的语言。HTML 文件是由一组尖括号形成的标签组织起来的，每对括号形成一对标签，标签之间存在一定的关系，形成标签树。可以利用 BeautifulSoup 库解析、遍历、维护这样的“标签树”。

【注意】 对于没有 HTML 基础的读者，简单学习 HTML 后将有助于对本书后续内容的理解。

1. BeautifulSoup 库中的一些基本元素

Tag：标签，最基本的信息组织单元，用< >标明开头，</>标明结尾。任何标签都可以用 soup.< tag >访问获得，如果 HTML 文档中存在多个相同< tag >对应内容时，soup.< tag >返回第一个该类型标签的内容。

Name：标签的名字，即尖括号中的字符串。如 … 的 name 是'p'。每个<tag>都有自己的名字，可通过<tag>.name 获取。

Attributes：标签的属性，字典形式组织。一个<tag>可以有零个或多个属性。可以通过<tag>.attrs 获取。

NavigableString：标签内非属性字符串，即<>和</>之间的字符串。可以通过<tag>.string 调用。

Comment：标签内字符串的注释部分，一种特殊的 Comment 类型。

2. BeautifulSoup 库的三种遍历方式

由于 HTML 格式是一个树状结构，BeautifulSoup 库提供了三种遍历方式，分别为下行遍历、上行遍历和平行遍历。利用下列属性可以轻松地对其进行遍历操作。

(1) 下行遍历。

.contents：子节点的列表，将<tag>所有的儿子节点存入列表。

.children：子节点的迭代类型，与.contents 相似。

.descendants：子孙节点的迭代类型，包含所有子孙节点。

(2) 上行遍历。

.parent：节点的父亲标签。

.parents：节点先辈标签的迭代类型。

(3) 平行遍历。

.next_sibling：返回按照 HTML 文本顺序的下一个平行节点标签。

.previous_sibling：返回按照 HTML 文本顺序的上一个平行节点标签。

.next_siblings：迭代类型，返回按照 HTML 文本顺序的所有平行节点标签。

.previous_siblings：迭代类型，返回按照 HTML 文本顺序的前续所有平行节点标签。

3. 查找内容

该如何对所需要的内容进行查找呢？这就需要用到一个方法<>.find_all(name, attrs, recursive, string, ** kwargs)。其中，name：对标签名称的检索字符串，attrs：对标签属性值的检索字符串，recursive：是否对子孙全部检索，默认为 True，string：<>…</>中字符串区域的检索字符串。因此，可以利用该方法对所需要的成员进行特征检索。该方法还有许多拓展方法，在此就不一一介绍了，感兴趣的读者可以自行阅读相关文件。

4. 定位文本内容

如果想定位所需要的文本内容，则要在需要爬取的页面按 F12 键，便调出开发者界面，即可直接查看该网页的 HTML 文本。Chrome 浏览器开发者模式如图 3-5 所示。

可以单击图 3-6 中的按钮，此后只需将光标悬停在需要查找的网页内容上，便可自动定位其在 HTML 文本中的标签位置。

例如，可以在图 3-5 所示的页面中快速定位该影片导演的信息所在的位置，如图 3-7 所示。

通过观察可以发现，“导演”标签的平行标签中 NavigableString 便是需要的导演名字。以此方法，可以快速定位电影编剧、主演、类型等信息。通过比较多个网页发现标签中的共同元素，便可以通过编写程序快速解析出需要的信息，并运用于所有的豆瓣电影简介页面。

图 3-5　Chrome 浏览器开发者模式

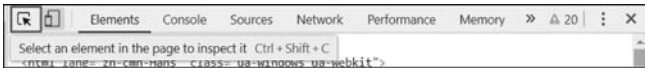

图 3-6　按钮

```
▼<div id="info">
  ▼<span>
      <span class="pl">导演</span> == $0
      ": "
    ▼<span class="attrs">
        <a href="/celebrity/1048021/" rel="v:directedBy">萨姆·门德斯</a>
      </span>
    </span>
    <br>
```

图 3-7　导演信息的 HTML 文本

下面编写程序来实验一下，如图 3-8 所示。

```
import requests
#导入Beautiful Soup库
import bs4

#将url链接更换为示例的链接
url = "https://movie.douban.com/subject/30252495/?tag=%E7%83%AD%E9%97%A8&from=gaia"
headers = {'user-agent':'Mozilla/5.0'}

try:
    r = requests.get(url, headers = headers)
    r.raise_for_status()
    r.encoding = r.apparent_encoding

    #用BeautifulSoup方法来解析得到的HTML文本
    soup = bs4.BeautifulSoup(r.text, "html.parser")
    #find与find_all类似，但只返回第一个匹配的对象
    director = soup.find("span",string = "导演").next_sibling.next_sibling
    #输出导演名字
    print(director.string)
except:
    print("爬取失败")
```

图 3-8　获取电影《1917》导演的爬虫程序

在控制台就可以看到输出的导演名字“萨姆·门德斯”。类似地，根据其他标签的特征，也可以爬取其他信息。

【试一试】 模仿上述代码，爬取一个你喜欢网页的信息。

3.4　批量爬取网页信息

学会了如何爬取一个网页，但是如何做到批量爬取呢？这里提供两个可行的方法作为参考。

（1）通过豆瓣的排行榜获取不同电影简介页面的网址，将其存入本地列表中，不断爬取。

（2）从每个豆瓣电影简介页面的“喜欢这部电影的人也喜欢”内容块中，获取相关电影简介页面的网址。

每爬取一个电影就能得到 10 个相关电影简介页面的网址，理论上的程序可以不断地获取新的电影简介页面的网址，并将其存入队列。只需要借助一个 while 循环便可以轻松地完成操作。

下面来编写一个完整的爬取豆瓣电影简介的程序，由于目标网站反爬措施升级，以下列出的部分程序可能会运行失效。列出如下代码的目的是交流学习，读者可以在掌握相关库的基本用法及思路的基础上，灵活做出调整。

```
import requests
import bs4
import queue


def get_HTMLtext(url, headers):
    try:
        r = requests.get(url, headers = headers)
        r.raise_for_status
        return r.text
    except:
        return None


def get_soup(text):
    return bs4.BeautifulSoup(text, "html.parser")


def get_name(soup):
    return soup.find("span", {"property": "v:itemreviewed"}).string


def get_director(soup):
    director = soup.find("span", string = "导演").next_sibling.next_sibling
    return director.string


def get_actors(soup):
    actors = ""
    act = soup.find("span", {"class": "actor"}).find("span", {"class": "attrs"})
    for actor in act:
        actors += actor.string
    return actors


def get_types(soup):
    types = ""
    for t in soup.find_all("span", {"property": "v:genre"}):
        types += t.string
        types += " "
    return types
```

```
# 获取更多 url 链接
def get_more_url(soup, queue):
    for t in soup.find_all("dd"):
        queue.put(t.a.get("href"))
    return


url = "https://movie.douban.com/subject/30252495/?tag - %E7%83%AD%E9%97%A8"
headers = {
    'User - Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/109.0.0.0 Safari/537.36'
}
my_queue = queue.Queue()
# 将起始网页存入队列
my_queue.put(item = url)
# 用于存储信息的列表,可根据需要换成其他数据结构存储
movie_list = []
# 已爬取网页数
num = 0
# 爬取程序,可指定爬取网页数
while (num < 10):
    # 创建字典存储信息
    movie = {}
    # 从队列中取出一个 url 链接
    url = my_queue.get()
    # 获取 HTML 文本
    text = get_HTMLtext(url, headers)
    # 判断是否失败
    if (text == None):
        continue
    # 利用 BeautifulSoup 库解析
    soup = get_soup(text)
    # 在字典中存入电影名
    movie["电影名"] = get_name(soup)
    # 在字典中存入导演
    movie["导演"] = get_director(soup)
    # 在字典中存入主演
    movie["主演"] = get_actors(soup)
    # 在字典中存入电影类型
    movie["类型"] = get_types(soup)
    # 将更多网页加入队列
    get_more_url(soup, my_queue)
    # 将本电影信息加入列表
    movie_list.append(movie)
    # 已经爬取页面数加 1
    num += 1

print(movie_list)
```

3.5　本章小结

本章通过介绍爬取豆瓣电影简介的案例详细分析了一个简单的爬虫程序。将爬虫分为四个步骤,便于读者理解。同时介绍了两个第三方库——Requests 库和 BeautifulSoup 库的使用方法,以及一些关于 HTML 的小知识。这能帮助读者更好地理解爬虫。