

DAY07

Day06回顾

多线程爬虫

■ 思路

```
1 1、将待爬取的URL地址存放到队列中
2 2、多个线程从队列中获取地址,进行数据抓取
3 3、注意获取地址过程中程序阻塞问题
4 while True:
5     if not q.empty():
6         url = q.get()
7         ... ..
8     else:
9         break
```

■ 将抓取数据保存到同一文件

```
1 # 注意多线程写入的线程锁问题
2 from threading import Lock
3 lock = Lock()
4 lock.acquire()
5 python代码块
6 lock.release()
```

■ 代码实现思路

```
1 # 1、在 __init__(self) 中创建文件对象,多线程操作此对象进行文件写入
2 self.f = open('xiaomi.csv','a',newline='')
3 self.writer = csv.writer(self.f)
4 self.lock = Lock()
5 # 2、每个线程抓取1页数据后将数据进行文件写入,写入文件时需要加锁
6 def parse_html(self):
7     app_list = []
8     for xxx in xxx:
9         app_list.append([name,link,typ])
10    self.lock.acquire()
11    self.wirter.writerow(app_list)
12    self.lock.release()
13 # 3、所有数据抓取完成关闭文件
14 def main(self):
```

```
15 | self.f.close()
```

解析模块汇总

re、lxml+xpath、json

```
1  # re
2  import re
3  pattern = re.compile(r'',re.S)
4  r_list = pattern.findall(html)
5
6  # lxml+xpath
7  from lxml import etree
8  parse_html = etree.HTML(html)
9  r_list = parse_html.xpath('')
10
11 # json
12 # 响应内容由json转为python
13 html = json.loads(res.text)
14 # 所抓数据保存到json文件
15 with open('xxx.json','a') as f:
16     json.dump(item_list,f,ensure_ascii=False)
17
18 # 或
19 f = open('xxx.json','a')
20 json.dump(item_list,f,ensure_ascii=False)
21 f.close()
```

Day07笔记

cookie模拟登录

适用网站及场景

```
1 | 抓取需要登录才能访问的页面
```

cookie和session机制

```
1 | # http协议为无连接协议
2 | cookie: 存放在客户端浏览器
3 | session: 存放在Web服务器
```

人人网登录案例

■ 方法一 - 登录网站手动抓取Cookie

```
1 1、先登录成功1次,获取到携带登录信息的Cookie
2   登录成功 - 个人主页 - F12抓包 - 刷新个人主页 - 找到主页的包(profile)
3 2、携带着cookie发请求
4   ** Cookie
5   ** User-Agent
```

```
1 # 1、将self.url改为 个人主页的URL地址
2 # 2、将Cookie的值改为 登录成功的Cookie值
3 import requests
4 from lxml import etree
5
6 class RenrenLogin(object):
7     def __init__(self):
8         self.url = 'xxxxxxx'
9         self.headers = {
10             # 自己抓到的cookie
11             'Cookie': 'xxxxxx',
12             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
13             Gecko) Chrome/76.0.3809.100 Safari/537.36'
14         }
15
16     def get_html(self):
17         html = requests.get(url=self.url, headers=self.headers).text
18         self.parse_html(html)
19
20     def parse_html(self, html):
21         parse_html = etree.HTML(html)
22         r_list = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')
23         print(r_list)
24
25 if __name__ == '__main__':
26     spider = RenrenLogin()
27     spider.get_html()
```

■ 方法二

原理

```
1 1、把抓取到的cookie处理为字典
2 2、使用requests.get()中的参数:cookies
```

处理cookie为字典

```
1 # 处理cookies为字典
2 cookies_dict = {}
3 cookies = 'xxxx'
4 for kv in cookies.split('; '):
5     cookies_dict[kv.split('=')[0]] = kv.split('=')[1]
```

代码实现

```
1 import requests
2 from lxml import etree
3
4 class RenrenLogin(object):
5     def __init__(self):
6         self.url = 'http://www.renren.com/967469305/profile'
7         self.headers = {
8             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
            Gecko) Chrome/76.0.3809.100 Safari/537.36'
9         }
10
11     # 获取字典形式cookie的函数
12     def get_cookie_dict(self):
13         cookie_dict = {}
14         # 自己抓到的cookie
15         cookies = 'xxxxxxx'
16         for kv in cookies.split('; '):
17             # kv: 'td_cookie=184xxx'
18             key = kv.split('=')[0]
19             value = kv.split('=')[1]
20             cookie_dict[key] = value
21
22         return cookie_dict
23
24     def get_html(self):
25         # 获取cookies
26         cookies = self.get_cookie_dict()
27         html = requests.get(
28             url=self.url,
29             headers=self.headers,
30             cookies=cookies,
31         ).text
32         self.parse_html(html)
33
34     def parse_html(self, html):
35         parse_html = etree.HTML(html)
36         r_list = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')
37         print(r_list)
38
39 if __name__ == '__main__':
40     spider = RenrenLogin()
41     spider.get_html()
```

■ 方法三 - requests模块处理Cookie

原理思路及实现

```
1 # 1. 思路
2 requests模块提供了session类,来实现客户端和服务端的会话保持
3
4 # 2. 原理
5 1、实例化session对象
6     session = requests.session()
7 2、让session对象发送get或者post请求
```

```

8     res = session.post(url=url,data=data,headers=headers)
9     res = session.get(url=url,headers=headers)
10
11 # 3. 思路梳理
12 浏览器原理：访问需要登录的页面会带着之前登录过的cookie
13 程序原理：同样带着之前登录的cookie去访问 - 由session对象完成
14 1、实例化session对象
15 2、登录网站：session对象发送请求,登录对应网站,把cookie保存在session对象中
16 3、访问页面：session对象请求需要登录才能访问的页面,session能够自动携带之前的这个cookie,进行请求

```

具体步骤

```

1 1、寻找Form表单提交地址 - 寻找登录时POST的地址
2   查看网页源码,查看form表单,找action对应的地址: http://www.renren.com/PLogin.do
3
4 2、发送用户名和密码信息到POST的地址
5   * 用户名和密码信息以什么方式发送? -- 字典
6   键 : <input>标签中name的值(email,password)
7   值 : 真实的用户名和密码
8   post_data = {'email':'','password':''}
9
10 session = requests.session()
11 session.post(url=url,data=data)

```

程序实现

```

1 # 把Formdata中的 email 和 password 的改为自己真实的用户名和密码
2 import requests
3 from lxml import etree
4
5 class RenrenSpider(object):
6     def __init__(self):
7         self.post_url = 'http://www.renren.com/PLogin.do'
8         self.get_url = 'http://www.renren.com/967469305/profile'
9         # 实例化session对象
10        self.session = requests.session()
11
12    def get_html(self):
13        # email和密码为<input>节点中name的属性值
14        form_data = {
15            'email' : 'xxxx',
16            'password' : 'xxxx'
17        }
18        # 先session.post()
19        self.session.post(url=self.post_url,data=form_data)
20        # 再session.get()
21        html = self.session.get(url=self.get_url).text
22        self.parse_html(html)
23
24    def parse_html(self,html):
25        parse_html = etree.HTML(html)
26        r_list = parse_html.xpath('//li[@class="school"]/span/text()')
27        print(r_list)
28
29 if __name__ == '__main__':

```

```
30 spider = RenrenSpider()
31 spider.get_html()
```

selenium+phantomjs/Chrome/Firefox

selenium

▪ 定义

- 1 1、Web自动化测试工具，可运行在浏览器，根据指令操作浏览器
- 2 2、只是工具，必须与第三方浏览器结合使用

▪ 安装

- 1 Linux: `sudo pip3 install selenium`
- 2 Windows: `python -m pip install selenium`

phantomjs浏览器

▪ 定义

- 1 无界面浏览器(又称无头浏览器)，在内存中进行页面加载,高效

▪ 安装(phantomjs、chromedriver、geckodriver)

Windows

- 1 1、下载对应版本的phantomjs、chromedriver、geckodriver
- 2 2、把chromedriver.exe拷贝到python安装目录的Scripts目录下(添加到系统环境变量)
查看python安装路径: `where python`
- 3 3、验证
cmd命令行: `chromedriver`
- 4
- 5
- 6
- 7 # 下载地址
- 8 1、chromedriver : 下载对应版本
<http://chromedriver.storage.googleapis.com/index.html>
- 9
- 10 2、geckodriver
<https://github.com/mozilla/geckodriver/releases>
- 11
- 12 3、phantomjs
<https://phantomjs.org/download.html>
- 13

Linux

```
1 1、下载后解压
2   tar -zxvf geckodriver.tar.gz
3 2、拷贝解压后文件到 /usr/bin/ (添加环境变量)
4   sudo cp geckodriver /usr/bin/
5 3、更改权限
6   sudo -i
7   cd /usr/bin/
8   chmod 777 geckodriver
```

■ 使用

示例代码一：使用 selenium+浏览器 打开百度

```
1 # 导入selenium的webdriver接口
2 from selenium import webdriver
3 import time
4
5 # 创建浏览器对象
6 browser = webdriver.PhantomJS()
7 browser.get('http://www.baidu.com/')
8
9 time.sleep(5)
10
11 # 关闭浏览器
12 browser.quit()
```

示例代码二：打开百度，搜索赵丽颖，点击搜索，查看

```
1 from selenium import webdriver
2 import time
3
4 # 1.创建浏览器对象 - 已经打开了浏览器
5 browser = webdriver.Chrome()
6 # 2.输入: http://www.baidu.com/
7 browser.get('http://www.baidu.com/')
8 # 3.找到搜索框,向这个节点发送文字: 赵丽颖
9 browser.find_element_by_xpath('//*[@id="kw"]').send_keys('赵丽颖')
10 # 4.找到 百度一下 按钮,点击一下
11 browser.find_element_by_xpath('//*[@id="su"]').click()
```

■ 浏览器对象(browser)方法

```
1 # from selenium import webdriver
2 1、 browser = webdriver.Chrome(executable_path='path')
3 2、 browser.get(url)
4 3、 browser.page_source # HTML结构源码
5 4、 browser.page_source.find('字符串')
6   # 从html源码中搜索指定字符串,没有找到返回: -1
7 5、 browser.quit() # 关闭浏览器
```

■ 定位节点

单元素查找(1个节点对象)

```
1 1、 browser.find_element_by_id('')
2 2、 browser.find_element_by_name('')
3 3、 browser.find_element_by_class_name('')
4 4、 browser.find_element_by_xpath('')
5 ... ..
```

多元素查找([节点对象列表])

```
1 1、 browser.find_elements_by_id('')
2 2、 browser.find_elements_by_name('')
3 3、 browser.find_elements_by_class_name('')
4 4、 browser.find_elements_by_xpath('')
5 ... ..
```

■ 节点对象操作

```
1 1、 ele.send_keys('') # 搜索框发送内容
2 2、 ele.click()
3 3、 ele.text # 获取文本内容, 包含子节点和后代节点的文本内容
4 4、 ele.get_attribute('src') # 获取属性值
```

京东爬虫案例

■ 目标

```
1 1、 目标网址 : https://www.jd.com/
2 2、 抓取目标 : 商品名称、 商品价格、 评价数量、 商品商家
```

■ 思路提醒

```
1 1、 打开京东, 到商品搜索页
2 2、 匹配所有商品节点对象列表
3 3、 把节点对象的文本内容取出来, 查看规律, 是否有更好的处理办法?
4 4、 提取完1页后, 判断如果不是最后1页, 则点击下一页
5 # 如何判断是否为最后1页? ? ?
```

■ 实现步骤

找节点

```
1 1、 首页搜索框 : //*[@id="key"]
2 2、 首页搜索按钮 : //*[@id="search"]/div/div[2]/button
3 3、 商品页的 商品信息节点对象列表 : //*[@id="J_goodsList"]/ul/li
4 4、 for循环遍历后
5 名称: ../div[@class="p-name p-name-type-2"]/a/em
6 价格: ../div[@class="p-price"]
7 评论: ../div[@class="p-commit"]/strong
8 商家: ../div[@class="p-shop"]
```


执行JS脚本，获取动态加载数据

```
1 browser.execute_script(  
2     'window.scrollTo(0,document.body.scrollHeight)'  
3 )
```

代码实现

```
1 from selenium import webdriver  
2 import time  
3  
4 class JdSpider(object):  
5     def __init__(self):  
6         self.url = 'https://www.jd.com/'  
7         # 创建浏览器对象  
8         self.browser = webdriver.Firefox()  
9         self.i = 0  
10        self.page = 1  
11  
12        # 跳转到商品详情页 - 爬虫书  
13        def get_html(self):  
14            # 找节点,send_keys() click()  
15            so = '//*[@id="key"]'  
16            button = '//*[@id="search"]/div/div[2]/button'  
17            self.browser.get(self.url)  
18            self.browser.find_element_by_xpath(so).send_keys('爬虫书')  
19            self.browser.find_element_by_xpath(button).click()  
20            # 必须的: 给页面留出加载时间  
21            time.sleep(3)  
22  
23            # 匹配每个商品信息的li节点对象列表, li.text  
24            def parse_html(self):  
25                # 拉到最下面,所有商品加载,再提取数据  
26                self.browser.execute_script(  
27                    'window.scrollTo(0,document.body.scrollHeight)'  
28                )  
29                time.sleep(3)  
30  
31                li_list = self.browser.find_elements_by_xpath('//*[@id="J_goodsList"]/ul/li')  
32                item = {}  
33                for li in li_list:  
34                    # 用 find_element_xxx  
35                    item['price'] = li.find_element_by_xpath('.//div[@class="p-price"]').text.strip()  
36                    item['name'] = li.find_element_by_xpath('.//div[@class="p-name p-name-type-  
37                    2"]/a/em').text.strip()  
38                    item['comment'] = li.find_element_by_xpath('.//div[@class="p-  
39                    commit"]/strong').text.strip()  
40                    item['market'] = li.find_element_by_xpath('.//div[@class="p-shop"]').text.strip()  
41                    print(item)  
42                    self.i += 1  
43                    print('第%d页抓取完成' % self.page)  
44                    self.page += 1  
45  
46            # 入口函数  
47            def run(self):
```

```
46         self.get_html()
47     for i in range(1):
48         self.parse_html()
49         # 判断是否为最后一页
50         if self.browser.page_source.find('pn-next disabled') == -1:
51             # -1说明没找到,不是最后一页,点击 下一页 按钮
52             self.browser.find_element_by_class_name('pn-next').click()
53             # 给新的一页元素加载预留时间
54             time.sleep(3)
55         # else:
56         #     break
57     print('数量:',self.i)
58
59 if __name__ == '__main__':
60     spider = JdSpider()
61     spider.run()
```