

Day05回顾

动态加载网站数据抓取

- 1 1、F12打开控制台，页面动作抓取网络数据包
- 2 2、抓取json文件URL地址
- 3 # 控制台中 XHR : 异步加载的数据包
- 4 # XHR -> Query String(查询参数)

cookie模拟登陆

- 1 1、适用网站类型：爬取网站页面时需要登录后才能访问，否则获取不到页面的实际响应数据
- 2 2、方法1（利用cookie）
 - 3 1、先登录成功1次,获取到携带登陆信息的Cookie（处理headers）
 - 4 2、利用处理的headers向URL地址发请求
- 5 3、方法2（利用session会话保持）
 - 6 1、登陆，找到POST地址：form -> action对应地址
 - 7 2、定义字典，创建session实例发送请求
 - 8 # 字典key : <input>标签中name的值(email,password)
 - 9 # post_data = {'email':'','password':''}

selenium+phantomjs/chrome/firefox

■ 特点

- 1 1、简单，无需去详细抓取分析网络数据包，使用真实浏览器
- 2 2、需要等待页面元素加载，需要时间，效率低

■ 使用流程

```
1 from selenium import webdriver
2
3 # 创建浏览器对象
4 browser = webdriver.Firefox()
5 browser.get('https://www.jd.com/')
6
7 # 查找节点
8 node = browser.find_element_by_xpath('')
9 node.send_keys('')
```

```
10 node.click()
11
12 # 获取节点文本内容
13 content = node.text
14
15 # 关闭浏览器
16 browser.quit()
```

■ 设置断点调试

- 1 通过search搜索关键字找到相关js文件
- 2 点击该文件并点击 {} 格式化输出js代码
- 3 单击前面的行号,重新发请求,会执行JS到断点处,显示每行代码的详细信息

Day06笔记

哔哩哔哩小视频抓取案例

■ 目标

- 1 url : <http://vc.bilibili.com/p/eden/rank#/?tab=全部>
- 2 抓取目标 : 所有异步加载的小视频

■ 实现步骤

1. F12抓包

```
1 json地址 : http://api.vc.bilibili.com/board/v1/ranking/top?
2 查询参数 :
3     page_size: 10
4     next_offset: 1 11 21 31 41 51 61
5     tag: 今日热门
6     platform: pc
```

■ 代码实现

```
1 '''哔哩哔哩小视频下载, 百度搜索: 哔哩哔哩小视频, 找到地址: '''
2
3 import requests
4 import time
5 import random
6 import string
7
```

```

8 class BilibiliVideoSpider(object):
9     def __init__(self):
10         self.url = 'http://api.vc.bilibili.com/board/v1/ranking/top?'
11         self.headers = {'User-Agent': 'Mozilla/5.0'}
12         self.all_chars = string.punctuation + string.whitespace
13
14     # 获取页面
15     def get_json(self):
16         for offset in range(1,22,10):
17             params = {
18                 'page_size': '10',
19                 'next_offset': offset,
20                 'tag' : '今日热门',
21                 'platform' : 'pc'
22             }
23             html = requests.get(url=self.url,params=params,headers=self.headers).json()
24             self.downloader(html)
25
26
27     # 下载视频
28     def downloader(self,html):
29         for video in html['data']['items']:
30             video_url = video['item']['video_playurl']
31             video_content = requests.get(video_url,headers=self.headers).content
32             filename = video['item']['description']
33             # 如何处理文件名中的特殊字符
34             for char in filename:
35                 if char in self.all_chars:
36                     filename = filename.replace(char, ' ')
37             filename = filename + '.mp4'
38
39             with open(filename.replace('\n',''), 'wb') as f:
40                 f.write(video_content)
41                 print('%s下载成功' % filename)
42             # 每爬取1个视频随机休眠几秒钟
43             time.sleep(random.randint(3,8))
44
45 if __name__ == '__main__':
46     spider = BilibiliVideoSpider()
47     spider.get_json()

```

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

1. 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
```

2. 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 )
```

3. 代码实现

```
1 from selenium import webdriver
2 import time
3
4 class JdSpider(object):
5     def __init__(self):
6         self.browser = webdriver.Chrome()
7         self.url = 'https://www.jd.com/'
8         self.i = 0
9
10    # 获取商品页面
11    def get_page(self):
12        self.browser.get(self.url)
13        # 找2个节点
14        self.browser.find_element_by_xpath('//*[@id="key"]').send_keys('爬虫书籍')
15        self.browser.find_element_by_xpath('//*[@id="search"]/div/div[2]/button').click()
16        time.sleep(2)
17
18    # 解析页面
19    def parse_page(self):
20        # 把下拉菜单拉到底部，执行JS脚本
21        self.browser.execute_script(
22            'window.scrollTo(0,document.body.scrollHeight)'
23        )
24        time.sleep(2)
25        # 匹配所有商品节点对象列表
26        li_list = self.browser.find_elements_by_xpath('//*[@id="J_goodsList"]/ul/li')
27        for li in li_list:
28            li_info = li.text.split('\n')
29            if li_info[0][0:2] == '每满':
30                price = li_info[1]
31                name = li_info[2]
32                commit = li_info[3]
```

```

33         market = li_info[4]
34     else:
35         price = li_info[0]
36         name = li_info[1]
37         commit = li_info[2]
38         market = li_info[3]
39     print('\033[31m*****\033[0m')
40     print(price)
41     print(commit)
42     print(market)
43     print(name)
44     self.i += 1
45
46     # 主函数
47     def main(self):
48         self.get_page()
49         while True:
50             self.parse_page()
51             # 判断是否该点击下一页,没有找到说明不是最后一页
52             if self.browser.page_source.find('pn-next disabled') == -1:
53                 self.browser.find_element_by_class_name('pn-next').click()
54                 time.sleep(2)
55             else:
56                 break
57         print(self.i)
58
59 if __name__ == '__main__':
60     spider = JdSpider()
61     spider.main()

```

chromedriver设置无界面模式

```

1  from selenium import webdriver
2
3  options = webdriver.ChromeOptions()
4  # 添加无界面参数
5  options.add_argument('--headless')
6  browser = webdriver.Chrome(options=options)
7  browser.get('http://www.baidu.com/')
8  browser.save_screenshot('baidu.png')

```

Fiddler抓包工具

- 配置Fiddler

```
1 # 添加证书信任
2 1、Tools - Options - HTTPS
3 勾选 Decrypt Https Traffic 后弹出窗口，一路确认
4 # 设置只抓取浏览器的数据包
5 2、...from browsers only
6 # 设置监听端口（默认为8888）
7 3、Tools - Options - Connections
8 # 配置完成后重启Fiddler（重要）
9 4、关闭Fiddler,再打开Fiddler
```

■ 配置浏览器代理

```
1 1、安装Proxy SwitchyOmega插件
2 2、浏览器右上角：SwitchyOmega->选项->新建情景模式->AID1901(名字)->创建
3 输入：HTTP:// 127.0.0.1 8888
4 点击：应用选项
5 3、点击右上角SwitchyOmega可切换代理
```

■ Fiddler常用菜单

```
1 1、Inspector：查看数据包详细内容
2 整体分为请求和响应两部分
3 2、常用菜单
4 Headers：请求头信息
5 WebForms：POST请求Form表单数据：<body>
6 GET请求查询参数：<QueryString>
7 Raw
8 将整个请求显示为纯文本
```

多线程爬虫

应用场景

```
1 1、多进程：CPU密集程序
2 2、多线程：爬虫(网络I/O)、本地磁盘I/O
```

知识点回顾

■ 队列

```

1  # 导入模块
2  from queue import Queue
3  # 使用
4  q = Queue()
5  q.put(url)
6  q.get() # 当队列为空时, 阻塞
7  q.empty() # 判断队列是否为空, True/False

```

■ 线程模块

```

1  # 导入模块
2  from threading import Thread
3
4  # 使用流程
5  t = Thread(target=函数名) # 创建线程对象
6  t.start() # 创建并启动线程
7  t.join() # 阻塞等待回收线程
8
9  # 如何创建多线程, 如下方法你觉得怎么样????
10 for i in range(5):
11     t = Thread(target=函数名)
12     t.start()
13     t.join()

```

小米应用商店抓取(多线程)

■ 目标

```

1  1、网址 : 百度搜 - 小米应用商店, 进入官网
2  2、目标 : 应用分类 - 聊天社交
3      应用名称
4      应用链接

```

■ 实现步骤

1. 确认是否为动态加载

```

1  1、页面局部刷新
2  2、右键查看网页源代码, 搜索关键字未搜到
3  # 此网站为动态加载网站, 需要抓取网络数据包分析

```

2. F12抓取网络数据包

```

1 1、抓取返回json数据的URL地址 (Headers中的Request URL)
2   http://app.mi.com/categotyAlllistApi?page={}&categoryId=2&pageSize=30
3
4 2、查看并分析查询参数 (headers中的Query String Parameters)
5   page: 1
6   categoryId: 2
7   pageSize: 30
8   # 只有page再变, 0 1 2 3 ... , 这样我们就可以通过控制page的直拼接多个返回json数据的URL地址

```

■ 代码实现

```

1  import requests
2  from threading import Thread
3  from queue import Queue
4  import json
5  import time
6
7  class XiaomiSpider(object):
8      def __init__(self):
9          self.headers = {'User-Agent': 'Mozilla/5.0'}
10         self.url = 'http://app.mi.com/categotyAlllistApi?page={}&categoryId=2&pageSize=30'
11         # 定义队列, 用来存放URL地址
12         self.url_queue = Queue()
13
14         # URL入队列
15         def url_in(self):
16             # 拼接多个URL地址, 然后put()到队列中
17             for i in range(67):
18                 self.url.format((str(i)))
19                 self.url_queue.put(self.url)
20
21         # 线程事件函数(请求, 解析提取数据)
22         def get_page(self):
23             # 先get()URL地址, 发请求
24             # json模块做解析
25             while True:
26                 # 当队列不为空时, 获取url地址
27                 if not self.url_queue.empty():
28                     url = self.url_queue.get()
29                     html = requests.get(url, headers=self.headers).text
30                     self.parse_page(html)
31                 else:
32                     break
33
34         # 解析函数
35         def parse_page(self, html):
36             app_json = json.loads(html)
37             for app in app_json['data']:
38                 # 应用名称
39                 name = app['displayName']
40                 # 应用链接
41                 link = 'http://app.mi.com/details?id={}'.format(app['packageName'])
42                 d = { '名称' : name, '链接' : link }
43                 print(d)
44
45         # 主函数

```



```

45     def main(self):
46         self.url_in()
47         # 存放所有线程的列表
48         t_list = []
49
50         for i in range(10):
51             t = Thread(target=self.get_page)
52             t.start()
53             t_list.append(t)
54
55         # 统一回收线程
56         for p in t_list:
57             p.join()
58
59 if __name__ == '__main__':
60     start = time.time()
61     spider = XiaomiSpider()
62     spider.main()
63     end = time.time()
64     print('执行时间:%.2f' % (end-start))

```

json解析模块

json.loads(json 格式字符串)

■ 作用

```
1 把json格式的字符串转为Python数据类型
```

■ 示例

```
1 html_json = json.loads(res.text)
```

json.dump(python,f,ensure_ascii=False)

■ 作用

```

1 把python数据类型 转为 json格式的字符串
2 # 一般让你把抓取的数据保存为json文件时使用

```

■ 参数说明

```

1 第1个参数: python类型的数据(字典, 列表等)
2 第2个参数: 文件对象
3 第3个参数: ensure_ascii=False # 序列化时编码

```

■ 示例

```
1 import json
2
3 app_dict = {
4     '应用名称' : 'QQ',
5     '应用链接' : 'http://qq.com'
6 }
7 with open('小米.json','a') as f:
8     json.dump(app_dict,f,ensure_ascii=False)
```

scrapy框架

■ 定义

1 异步处理框架,可配置和可扩展程度非常高,Python中使用最广泛的爬虫框架

■ 安装

```
1 # Ubuntu16.04安装
2 1、安装依赖包
3     1、sudo apt-get install libffi-dev
4     2、sudo apt-get install libssl-dev
5     3、sudo apt-get install libxml2-dev
6     4、sudo apt-get install python3-dev
7     5、sudo apt-get install libxslt1-dev
8     6、sudo apt-get install zlib1g-dev
9     7、sudo pip3 install -I -U service_identity
10 2、安装scrapy框架
11     1、sudo pip3 install Scrapy
```

```
1 # Windows安装
2 cmd命令行(管理员): pip install Scrapy
```

■ Scrapy框架五大组件

```
1 1、引擎(Engine)      : 整个框架核心
2 2、调度器(Scheduler) : 维护请求队列
3 3、下载器(Downloader): 获取响应对象
4 4、爬虫文件(Spider)  : 数据解析提取
5 5、项目管道(Pipeline): 数据入库处理
6 *****
7 # 下载器中间件(Downloader Middlewares) : 引擎->下载器,包装请求(随机代理等)
8 # 蜘蛛中间件(Spider Middlewares) : 引擎->爬虫文件,可修改响应对象属性
```

■ scrapy爬虫工作流程

```

1  # 爬虫项目启动
2  1、由引擎向爬虫程序索要第一个要爬取的URL,交给调度器去入队列
3  2、调度器处理请求后出队列,通过下载器中间件交给下载器去下载
4  3、下载器得到响应对象后,通过蜘蛛中间件交给爬虫程序
5  4、爬虫程序进行数据提取:
6      1、数据交给管道文件去入库处理
7      2、对于需要继续跟进的URL,再次交给调度器入队列,依次循环

```

■ scrapy常用命令

```

1  # 1、创建爬虫项目
2  scrapy startproject 项目名
3  # 2、创建爬虫文件
4  scrapy genspider 爬虫名 域名
5  # 3、运行爬虫
6  scrapy crawl 爬虫名

```

■ scrapy项目目录结构

```

1  Baidu                # 项目文件夹
2  └─ Baidu             # 项目目录
3  │   └─ items.py      # 定义数据结构
4  │   └─ middlewares.py # 中间件
5  │   └─ pipelines.py  # 数据处理
6  │   └─ settings.py   # 全局配置
7  │   └─ spiders
8  │       └─ baidu.py  # 爬虫文件
9  └─ scrapy.cfg        # 项目基本配置文件

```

■ 全局配置文件settings.py详解

```

1  # 1、定义User-Agent
2  USER_AGENT = 'Mozilla/5.0'
3  # 2、是否遵循robots协议,一般设置为False
4  ROBOTSTXT_OBEY = False
5  # 3、最大并发量,默认为16
6  CONCURRENT_REQUESTS = 32
7  # 4、下载延迟时间
8  DOWNLOAD_DELAY = 1
9  # 5、请求头,此处也可以添加User-Agent
10 DEFAULT_REQUEST_HEADERS={}
11 # 6、项目管道
12 ITEM_PIPELINES={
13     '项目目录名.pipelines.类名':300
14 }

```

■ 创建爬虫项目步骤

```
1 1、新建项目 : scrapy startproject 项目名
2 2、cd 项目文件夹
3 3、新建爬虫文件 : scrapy genspider 文件名 域名
4 4、明确目标(items.py)
5 5、写爬虫程序(文件名.py)
6 6、管道文件(pipelines.py)
7 7、全局配置(settings.py)
8 8、运行爬虫 : scrapy crawl 爬虫名
```

■ pycharm运行爬虫项目

```
1 1、创建begin.py(和scrapy.cfg文件同目录)
2 2、begin.py中内容:
3     from scrapy import cmdline
4     cmdline.execute('scrapy crawl maoyan'.split())
```

今日作业

- 1、熟记多线程爬虫原理及会写，更改腾讯招聘项目为多线程爬虫
- 2、熟记如下问题

```
1 1、scrapy框架有哪几大组件？
2 2、各个组件之间是如何工作的？
```

3、Windows安装scrapy

```
1 Windows : python -m pip install Scrapy
2 # Error: Microsoft Visual C++ 14.0 is required
3 # 解决 : 下载安装 Microsoft Visual C++ 14.0
```

4、Ubuntu 16.04 安装Scrapy