

Day06回顾

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、实例化session对象
7         session = requests.session()
8     2、先post : session.post(post_url,data=post_data,headers=headers)
9         1、登陆，找到POST地址：form -> action对应地址
10        2、定义字典，创建session实例发送请求
11            # 字典key : <input>标签中name的值(email,password)
12            # post_data = {'email':'','password':''}
13    3、再get : session.get(url,headers=headers)
```

设置断点调试

```
1 1、抓取到js文件的相关代码
2 2、单击前面的行号,刷新页面,会执行JS到断点处
3 3、鼠标移动到相关位置,可进行函数的跳转
```

在程序中执行js文件

```
1 import execjs
2
3 with open('文件名.js','r') as f:
4     js_data = f.read()
5
6 js_obj = execjs.compile(js_data)
7 result = js_obj.eval('js函数名("参数")')
```

增量爬取思路

- 1 1、将爬取过的地址存放到数据库中
- 2 2、程序爬取时先到数据库中查询比对，如果已经爬过则不会继续爬取

Day07笔记

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 # 创建多线程
10 t_list = []
11 for i in range(5):
12     t = Thread(target=函数名)
```

```
13     t_list.append(t)
14     t.start()
15
16 for i in t_list:
17     i.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
```

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)
```

练习: 将链家二手房代码改写为多线程方式

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)
3   # 查看python安装路径: where python
4 3、验证
5   cmd命令行: chromedriver
6
7   # 下载地址
8   chromedriver : 下载对应版本
9   http://chromedriver.storage.googleapis.com/index.html
10
11  geckodriver
12  https://github.com/mozilla/geckodriver/releases
```

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 |
```

示例代码二: 打开百度, 搜索赵丽颖, 查看

```
1 |
```

▪ 浏览器对象(browser)方法

```
1 1、 browser = webdriver.Chrome(executable_path='path')
2 2、 browser.get(url)
3 3、 browser.page_source # 查看响应内容
4 4、 browser.page_source.find('字符串')
5   # 从html源码中搜索指定字符串,没有找到返回: -1
6 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
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

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 |
```

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')
```

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  # Ubuntu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): python -m 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
9

```

■ 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