DAY07

Day06回顾

多线程爬虫

- **思路**

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

- **将抓取数据保存到同一文件**

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

- **代码实现思路**

```
# 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
  self.f = open('xiaomi.csv','a',newline='')
  self.writer = csv.writer(self.f)
  self.lock = Lock()
# 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
  def parse_html(self):
    app_list = []
    for xxx in xxx:
        app_list.append([name,link,typ])
    self.lock.acquire()
    self.wirter.writerows(app_list)
    self.lock.release()
# 3、所有数据抓取完成关闭文件
  def main(self):
```

```
    self.f.close()
```

解析模块汇总

re、lxml+xpath、json

```
# re
import re
pattern = re.compile(r'',re.S)
r_list = pattern.findall(html)

# lxml+xpath
from lxml import etree
parse_html = etree.HTML(html)
r_list = parse_html.xpath('')

# json
# 响应内容由json转为python
html = json.loads(res.text)
# 所抓数据保存到json文件
with open('xxx.json','a') as f:
  json.dump(item_list,f,ensure_ascii=False)

# 或
f = open('xxx.json','a')
json.dump(item_list,f,ensure_ascii=False)
f.close()
```

Day07笔记

cookie模拟登录

适用网站及场景

```
抓取需要登录才能访问的页面
```

cookie和session机制

```
# http协议为无连接协议
cookie：存放在客户端浏览器
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、先登录成功1次,获取到携带登录信息的Cookie
   登录成功 - 个人主页 - F12抓包 - 刷新个人主页 - 找到主页的包(profile)
2、携带着cookie发请求
   ** Cookie
   ** User-Agent
```

```

```

- **方法二**

原理

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

处理cookie为字典

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

代码实现

```

```

- **方法三 - requests模块处理Cookie**

原理思路及实现

```
# 1. 思路
requests模块提供了session类,来实现客户端和服务端的会话保持

# 2. 原理
1、实例化session对象
   session = requests.session()
2、让session对象发送get或者post请求
   res = session.post(url=url,data=data,headers=headers)
   res = session.get(url=url,headers=headers)

# 3. 思路梳理
浏览器原理: 访问需要登录的页面会带着之前登录过的cookie
程序原理: 同样带着之前登录的cookie去访问 - 由session对象完成
1、实例化session对象
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
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、寻找Form表单提交地址 - 寻找登录时POST的地址
   查看网页源码,查看form表单,找action对应的地址: http://www.renren.com/PLogin.do

2、发送用户名和密码信息到POST的地址
   * 用户名和密码信息以什么方式发送?  -- 字典
     键 : <input>标签中name的值(email,password)
     值 : 真实的用户名和密码
     post_data = {'email':'','password':''}

session = requests.session()
session.post(url=url,data=data)
```

程序实现

```

```

selenium+phantomjs/Chrome/Firefox

selenium

- **定义**

```
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、只是工具，必须与第三方浏览器结合使用
```

- **安装**

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

phantomjs浏览器

- **定义**

```
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、下载对应版本的phantomjs、chromedriver、geckodriver
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、验证
   cmd命令行：chromedriver

# 下载地址
1、chromedriver ： 下载对应版本
http://chromedriver.storage.googleapis.com/index.html
2、geckodriver
https://github.com/mozilla/geckodriver/releases
3、phantomjs
https://phantomjs.org/download.html
```

Linux

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

- **使用**

示例代码一：使用 selenium+浏览器 打开百度

```
# 导入seleinum的webdriver接口
from selenium import webdriver
import time

# 创建浏览器对象
browser = webdriver.PhantomJS()
browser.get('http://www.baidu.com/')

time.sleep(5)

# 关闭浏览器
browser.quit()
```

示例代码二：打开百度，搜索赵丽颖，点击搜索，查看

```python
from selenium import webdriver
import time

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

- **浏览器对象(browser)方法**

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

- **定位节点**

单元素查找(1个节点对象)

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

多元素查找([节点对象列表])

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

- 节点对象操作

```python
1、ele.send_keys('') # 搜索框发送内容
2、ele.click()
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、ele.get_attribute('src') # 获取属性值
```

京东爬虫案例

- **目标**

```
1、目标网址 ： https://www.jd.com/
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、打开京东，到商品搜索页
2、匹配所有商品节点对象列表
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、提取完1页后，判断如果不是最后1页，则点击下一页
   # 如何判断是否为最后1页？？？
```

- **实现步骤**

找节点

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

执行JS脚本，获取动态加载数据

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

代码实现

```

```