DAY06

Day05回顾

控制台抓包

打开方式及常用选项

```
1、打开浏览器，F12打开控制台，找到Network选项卡
2、控制台常用选项
    1、Network：抓取网络数据包
          1、ALL：抓取所有的网络数据包
          2、XHR：抓取异步加载的网络数据包
          3、JS ：抓取所有的JS文件
    2、Sources：格式化输出并打断点调试JavaScript代码，助于分析爬虫中一些参数
    3、Console：交互模式，可对JavaScript中的代码进行测试
3、抓取具体网络数据包后
    1、单击左侧网络数据包地址，进入数据包详情，查看右侧
    2、右侧：
        1、Headers：整个请求信息
             General、Response Headers、Request Headers、Query String、Form Data
        2、Preview：对响应内容进行预览
        3、Response：响应内容
```

有道翻译过程梳理

```
1. 打开首页
2. 准备抓包：F12开启控制台
3. 寻找地址
   页面中输入翻译单词，控制台中抓取到网络数据包，查找并分析返回翻译数据的地址
4. 发现规律
   找到返回具体数据的地址，在页面中多输入几个单词，找到对应URL地址，分析对比 Network - All(或者XHR) - Form Data，发现对应的规律
5. 寻找JS文件
   右上角 ... -> Search -> 搜索关键字 -> 单击 -> 跳转到Sources，左下角格式化符号{}
6、查看JS代码
   搜索关键字，找到相关加密方法
7、断点调试
8、完善程序
```

增量爬取思路

```
1、将爬取过的地址存放到数据库中
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
```

动态加载网站数据抓取

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

数据抓取最终梳理

```
# 响应内容中存在
1、确认抓取数据在响应内容中是否存在
2、分析页面结构，观察URL地址规律
    1、大体查看响应内容结构,查看是否有更改 --（百度视频案例）
    2、查看页面跳转时URL地址变化,查看是否新跳转 --（民政部案例）
3、开始码代码进行数据抓取

# 响应内容中不存在
1、确认抓取数据在响应内容中是否存在
2、F12抓包,开始刷新页面或执行某些行为,主要查看XHR异步加载数据包
    1、GET请求: Request Headers、Query String Paramters
    2、POST请求:Request Headers、FormData
3、观察查询参数或者Form表单数据规律,如果需要进行进一步抓包分析处理
    1、比如有道翻译的 salt+sign,抓取并分析JS做进一步处理
    2、此处注意请求头中的cookie和referer以及User-Agent
4、使用res.json()获取数据,利用列表或者字典的方法获取所需数据
```

Day06笔记

动态加载数据抓取-Ajax

- **特点**

```
1、右键 -> 查看网页源码中没有具体数据
2、滚动鼠标滑轮或其他动作时加载,或者页面局部刷新
```

- **抓取**

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

豆瓣电影数据抓取案例

- **目标**

```
1、地址：豆瓣电影 - 排行榜 - 剧情
2、目标：电影名称、电影评分
```

- **F12抓包（XHR）**

```
1、Request URL(基准URL地址) ：https://movie.douban.com/j/chart/top_list?
2、Query String(查询参数)

# 抓取的查询参数如下：
type: 13 # 电影类型
interval_id: 100:90
action: ''
start: 0  # 每次加载电影的起始索引值 0 20 40 60
limit: 20 # 每次加载的电影数量
```

- **代码实现 - 全站抓取 - 完美接口 - 指定类型所有电影信息**

```python
import requests
import time
import random
import re
from useragents import ua_list

class DoubanSpider(object):
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?'
        self.i = 0

    # 获取随机headers
    def get_headers(self):
        headers = {'User-Agent':random.choice(ua_list)}

        return headers

    # 获取页面
    def get_page(self,params):
        headers = self.get_headers()
        res = requests.get(url=self.url,params=params,headers=headers)
        res.encoding = 'utf-8'
        # 返回 python 数据类型
```

```
        html = res.json()
        self.parse_page(html)

    # 解析并保存数据
    def parse_page(self,html):
        item = {}
        # html为大列表 [{电影1信息},{},{}]
        for one in html:
            # 名称 + 评分
            item['name'] = one['title'].strip()
            item['score'] = float(one['score'].strip())
            # 打印测试
            print(item)
            self.i += 1

    # 获取电影总数
    def total_number(self,type_number):
        # F12抓包抓到的地址
        url = 'https://movie.douban.com/j/chart/top_list_count?type=
{}&interval_id=100%3A90'.format(type_number)
        headers = self.get_headers()
        html = requests.get(url=url,headers=headers).json()
        total = int(html['total'])

        return total

    # 获取所有电影的名字和对应type值
    def get_all_type_films(self):
        # 获取 类型和类型码
        url = 'https://movie.douban.com/chart'
        headers = self.get_headers()
        html = requests.get(url=url,headers=headers).text
        re_bds = r'<a href=.*?type_name=(.*?)&type=(.*?)&.*?</a>'
        pattern = re.compile(re_bds,re.S)
        r_list = pattern.findall(html)
        # 存放所有类型和对应类型码大字典
        type_dict = {}
        menu = ''
        for r in r_list:
            type_dict[r[0].strip()] = r[1].strip()
            # 获取input的菜单，显示所有电影类型
            menu += r[0].strip() + '|'

        return type_dict,menu


    # 主函数
    def main(self):
        # 获取type的值
        type_dict,menu = self.get_all_type_films()
        menu = menu + '\n请做出你的选择:'
        name = input(menu)
        type_number = type_dict[name]
        # 获取电影总数
        total = self.total_number(type_number)
        for start in range(0,(total+1),20):
            params = {
```

```
                'type' : type_number,
                'interval_id' : '100:90',
                'action' : '',
                'start' : str(start),
                'limit' : '20'
            }
            # 调用函数,传递params参数
            self.get_page(params)
            # 随机休眠1-3秒
            time.sleep(random.randint(1,3))
        print('电影数量:',self.i)

if __name__ == '__main__':
    spider = DoubanSpider()
    spider.main()
```

json解析模块

json.loads(json)

- **作用**

```
把json格式的字符串转为Python数据类型
```

- **示例**

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

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

- **作用**

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

- **参数说明**

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

- **示例1**

```
import json

item = {'name':'QQ','app_id':1}
with open('小米.json','a') as f:
  json.dump(item,f,ensure_ascii=False)
```

- **示例2**

```
import json

item_list = []
for i in range(3):
  item = {'name':'QQ','id':i}
  item_list.append(item)

with open('xiaomi.json','a') as f:
  json.dump(item_list,f,ensure_ascii=False)
```

json.dumps(python)

- **作用**

```
把 python 类型 转为 json 类型
```

- **示例**

```
import json

# json.dumps()之前
item = {'name':'QQ','app_id':1}
print('before dumps',type(item)) # dict
# json.dumps之后
item = json.dumps(item)
print('after dumps',type(item)) # str
```

json.load(f)

作用

```
将json文件读取,并转为python类型
```

示例

```
import json

with open('D:\\spider_test\\xiaomi.json','r') as f:
    data = json.load(f)

print(data)
```

json模块总结

```
# 爬虫最常用
1、数据抓取 - json.loads(html)
   将响应内容由: json 转为 python
2、数据保存 - json.dump(item_list,f,ensure_ascii=False)
   将抓取的数据保存到本地 json文件

# 抓取数据一般处理方式
1、txt文件
2、csv文件
3、json文件
4、MySQL数据库
5、MongoDB数据库
6、Redis数据库
```

腾讯招聘数据抓取

- **确定URL地址及目标**

```
1、URL: 百度搜索腾讯招聘 - 查看工作岗位
2、目标: 职位名称、工作职责、岗位要求
```

- **要求与分析**

```
1、通过查看网页源码,得知所需数据均为 Ajax 动态加载
2、通过F12抓取网络数据包,进行分析
3、一级页面抓取数据: 职位名称
4、二级页面抓取数据: 工作职责、岗位要求
```

- **一级页面json地址(index在变,timestamp未检查)**

```
https://careers.tencent.com/tencentcareer/api/post/Query?timestamp=1563912271089&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attrId=&keyword=&pageIndex={}&pageSize=10&language=zh-cn&area=cn
```

- **二级页面地址(postId在变,在一级页面中可拿到)**

```
https://careers.tencent.com/tencentcareer/api/post/ByPostId?timestamp=1563912374645&postId={}&language=zh-cn
```

- **代码实现**

```
import requests
import json
import time
import random
from useragents import ua_list

class TencentSpider(object):
  def __init__(self):
```

```
    self.one_url = 'https://careers.tencent.com/tencentcareer/api/post/Query?timestamp=1563912271089&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attrId=&keyword=&pageIndex={}&pageSize=10&language=zh-cn&area=cn'
    self.two_url = 'https://careers.tencent.com/tencentcareer/api/post/ByPostId?timestamp=1563912374645&postId={}&language=zh-cn'
    # 打开文件
    self.f = open('tencent.json','a')
    # 存放抓取的item字典数据
    self.item_list = []

  # 获取响应内容函数
  def get_page(self,url):
    headers = {'User-Agent':random.choice(ua_list)}
    html = requests.get(url=url,headers=headers).text
    # json格式字符串 -> Python
    html = json.loads(html)

    return html

  # 主线函数：获取所有数据
  def parse_page(self,one_url):
    html = self.get_page(one_url)
    item = {}
    for job in html['Data']['Posts']:
      # postId
      post_id = job['PostId']
      # 拼接二级地址,获取职责和要求
      two_url = self.two_url.format(post_id)
      item['name'],item['duty'],item['require'] = self.parse_two_page(two_url)

      print(item)
      # 添加到大列表中
      self.item_list.append(item)

  # 解析二级页面函数
  def parse_two_page(self,two_url):
    html = self.get_page(two_url)
    # 职位名称
    name = html['Data']['RecruitPostName']
    # 用replace处理一下特殊字符
    duty = html['Data']['Responsibility']
    duty = duty.replace('\r\n','').replace('\n','')
    # 处理要求
    require = html['Data']['Requirement']
    require = require.replace('\r\n','').replace('\n','')

    return name,duty,require

  # 获取总页数
  def get_numbers(self):
    url = self.one_url.format(1)
    html = self.get_page(url)
    numbers = int(html['Data']['Count']) // 10 + 1

    return numbers

  def main(self):
```

```
    number = self.get_numbers()
    for page in range(1,3):
      one_url = self.one_url.format(page)
      self.parse_page(one_url)

    # 保存到本地json文件:json.dump
    json.dump(self.item_list,self.f,ensure_ascii=False)
    self.f.close()

if __name__ == '__main__':
  spider = TencentSpider()
  spider.main()
```

多线程爬虫

应用场景

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

知识点回顾

- **队列**

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

- **线程模块**

```
# 导入模块
from threading import Thread

# 使用流程
t = Thread(target=函数名) # 创建线程对象
t.start() # 创建并启动线程
t.join()  # 阻塞等待回收线程

# 如何创建多线程
t_list = []

for i in range(5):
    t = Thread(target=函数名)
    t_list.append(t)
    t.start()

for t in t_list:
```

```
    t.join()
```

小米应用商店抓取(多线程)

目标

```
1、网址 ： 百度搜 - 小米应用商店，进入官网
2、目标 ： 所有应用分类
   应用名称
   应用链接
```

实现步骤

- **1、确认是否为动态加载**

```
1、页面局部刷新
2、右键查看网页源代码，搜索关键字未搜到
# 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、抓取返回json数据的URL地址（Headers中的Request URL）
   http://app.mi.com/categotyAllListApi?page={}&categoryId=2&pageSize=30

2、查看并分析查询参数（headers中的Query String Parameters）
   page: 1
   categoryId: 2
   pageSize: 30
   # 只有page在变，0 1 2 3 ... ... ，这样我们就可以通过控制page的值拼接多个返回json数据的URL地址
```

- **将抓取数据保存到csv文件**

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

- **整体思路**

```
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()
```

- **代码实现**

```
import requests
from threading import Thread
from queue import Queue
import time
from useragents import ua_list
from lxml import etree
import csv
from threading import Lock
import random

class XiaomiSpider(object):
  def __init__(self):
    self.url = 'http://app.mi.com/categotyAllListApi?page={}&categoryId={}&pageSize=30'
    # 存放所有URL地址的队列
    self.q = Queue()
    self.i = 0
    # 存放所有类型id的空列表
    self.id_list = []
    # 打开文件
    self.f = open('xiaomi.csv','a')
    self.writer = csv.writer(self.f)
    # 创建锁
    self.lock = Lock()


  def get_cateid(self):
    # 请求
    url = 'http://app.mi.com/'
    headers = { 'User-Agent': random.choice(ua_list)}
    html = requests.get(url=url,headers=headers).text
    # 解析
    parse_html = etree.HTML(html)
    xpath_bds = '//ul[@class="category-list"]/li'
    li_list = parse_html.xpath(xpath_bds)
    for li in li_list:
      typ_name = li.xpath('./a/text()')[0]
      typ_id = li.xpath('./a/@href')[0].split('/')[-1]
      # 计算每个类型的页数
      pages = self.get_pages(typ_id)
      self.id_list.append( (typ_id,pages) )

    # 入队列
    self.url_in()

```

```python
    # 获取counts的值并计算页数
    def get_pages(self,typ_id):
        # 每页返回的json数据中,都有count这个key
        url = self.url.format(0,typ_id)
        html = requests.get(
            url=url,
            headers={'User-Agent':random.choice(ua_list)}
        ).json()
        count = html['count']
        pages = int(count) // 30 + 1

        return pages

    # url入队列
    def url_in(self):
        for id in self.id_list:
            # id为元组,('2',pages)
            for page in range(1,id[1]+1):
                url = self.url.format(page,id[0])
                # 把URL地址入队列
                self.q.put(url)

    # 线程事件函数: get() - 请求 - 解析 - 处理数据
    def get_data(self):
        while True:
            if not self.q.empty():
                url = self.q.get()
                headers = {'User-Agent':random.choice(ua_list)}
                html = requests.get(url=url,headers=headers).json()
                self.parse_html(html)
            else:
                break

    # 解析函数
    def parse_html(self,html):
        # 存放1页的数据 - 写入到csv文件
        app_list = []

        for app in html['data']:
            # 应用名称 + 链接 + 分类
            name = app['displayName']
            link = 'http://app.mi.com/details?id=' + app['packageName']
            typ_name = app['level1CategoryName']
            # 把每一条数据放到app_list中,目的为了 writerows()
            app_list.append([name,typ_name,link])

            print(name,typ_name)
            self.i += 1

        # 开始写入1页数据 - app_list
        self.lock.acquire()
        self.writer.writerows(app_list)
        self.lock.release()

    # 主函数
    def main(self):
        # URL入队列
```

```
    self.get_cateid()
    t_list = []
    # 创建多个线程
    for i in range(1):
      t = Thread(target=self.get_data)
      t_list.append(t)
      t.start()

    # 回收线程
    for t in t_list:
      t.join()

    # 关闭文件
    self.f.close()
    print('数量:',self.i)

if __name__ == '__main__':
  start = time.time()
  spider = XiaomiSpider()
  spider.main()
  end = time.time()
  print('执行时间:%.2f' % (end-start))
```

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
```

```
# 1、将self.url改为 个人主页的URL地址
# 2、将Cookie的值改为 登录成功的Cookie值
```

```
import requests
from lxml import etree

class RenrenLogin(object):
  def __init__(self):
    self.url = 'xxxxxxx'
    self.headers = {
      # 自己抓到的cookie
      'Cookie':'xxxxxx',
      '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'
    }

  def get_html(self):
    html = requests.get(url=self.url,headers=self.headers).text
    self.parse_html(html)

  def parse_html(self,html):
    parse_html = etree.HTML(html)
    r_list = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')
    print(r_list)

if __name__ == '__main__':
  spider = RenrenLogin()
  spider.get_html()
```

- **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

```
import requests
from lxml import etree

class RenrenLogin(object):
  def __init__(self):
    self.url = 'http://www.renren.com/967469305/profile'
    self.headers = {
      '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'
    }

```

```
  # 获取字典形式cookie的函数
  def get_cookie_dict(self):
    cookie_dict = {}
    # 自己抓到的cookie
    cookies = 'xxxxxxx'
    for kv in cookies.split('; '):
      # kv: 'td_cookie=184xxx'
      key = kv.split('=')[0]
      value = kv.split('=')[1]
      cookie_dict[key] = value

    return cookie_dict

  def get_html(self):
    # 获取cookies
    cookies = self.get_cookie_dict()
    html = requests.get(
      url=self.url,
      headers=self.headers,
      cookies=cookies,
    ).text
    self.parse_html(html)

  def parse_html(self,html):
    parse_html = etree.HTML(html)
    r_list = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')
    print(r_list)

if __name__ == '__main__':
  spider = RenrenLogin()
  spider.get_html()
```

- **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

```python
# 把Formdata中的 email 和 password 的改为自己真实的用户名和密码
import requests
from lxml import etree

class RenrenSpider(object):
  def __init__(self):
    self.post_url = 'http://www.renren.com/PLogin.do'
    self.get_url = 'http://www.renren.com/967469305/profile'
    # 实例化session对象
    self.session = requests.session()

  def get_html(self):
    # email和password为<input>节点中name的属性值
    form_data = {
      'email' : 'xxxx',
      'password' : 'xxxx'
    }
    # 先session.post()
    self.session.post(url=self.post_url,data=form_data)
    # 再session.get()
    html = self.session.get(url=self.get_url).text
    self.parse_html(html)

  def parse_html(self,html):
    parse_html = etree.HTML(html)
    r_list = parse_html.xpath('//li[@class="school"]/span/text()')
    print(r_list)

if __name__ == '__main__':
  spider = RenrenSpider()
  spider.get_html()
```

今日作业

```
1、多线程改写 - 腾讯招聘案例
2、多线程改写 - 链家二手房案例
3、尝试破解百度翻译
```