

# DAY06

## Day05回顾

### 控制台抓包

打开方式及常用选项

- 1 1、打开浏览器，F12打开控制台，找到Network选项卡
- 2 2、控制台常用选项
  - 3 1、Network：抓取网络数据包
    - 4 1、ALL：抓取所有的网络数据包
    - 5 2、XHR：抓取异步加载的网络数据包
    - 6 3、JS：抓取所有的JS文件
  - 7 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
  - 8 3、Console：交互模式，可对JavaScript中的代码进行测试
- 9 3、抓取具体网络数据包后
  - 10 1、单击左侧网络数据包地址，进入数据包详情，查看右侧
  - 11 2、右侧：
    - 12 1、Headers：整个请求信息  
General、Response Headers、Request Headers、Query String、Form Data
    - 13 2、Preview：对响应内容进行预览
    - 14 3、Response：响应内容

### 有道翻译过程梳理

- 1 1. 打开首页
- 2 2. 准备抓包：F12开启控制台
  - 3 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 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 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 6. 查看JS代码  
搜索关键字，找到相关加密方法
  - 7 7. 断点调试
  - 8 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

### 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 1、Request URL(基准URL地址) : https://movie.douban.com/j/chart/top_list?
2 2、Query String(查询参数)
3
4 # 抓取的查询参数如下:
5 type: 13 # 电影类型
6 interval_id: 100:90
7 action: ''
8 start: 0 # 每次加载电影的起始索引值 0 20 40 60
9 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

```
1 |
```

# 腾讯招聘数据抓取

## ■ 确定URL地址及目标

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

## ■ 要求与分析

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

```
1 https://careers.tencent.com/tencentcareer/api/post/Query?
timestamp=1563912271089&countryId=&cityId=&bgIds=&productId=&categoryId=&parentCategoryId=&attr
Id=&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)

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

## ■ 代码实现

```
1 |
```

# 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

```
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?????
```

## 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
- 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、页面局部刷新
- 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

```
1 # 注意多线程写入的线程锁问题
2 from threading import Lock
3 lock = Lock()
4 # 加锁
5 lock.acquire()
6 python语句
7 # 释放锁
8 lock.release()
```

#### ■ 整体思路

```
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
3 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
4
5 3、所有数据抓取完成关闭文件
6
```

#### ■ 代码实现

```
1
```

## 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             'Cookie': 'xxxxxx',
11             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
12             Gecko) Chrome/76.0.3809.100 Safari/537.36'
13         }
14
15     def get_html(self):
16         html = requests.get(url=self.url, headers=self.headers).text
17         self.parse_html(html)
18
19     def parse_html(self, html):
20         parse_html = etree.HTML(html)
21         r_list = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')
22         print(r_list)
23
24 if __name__ == '__main__':
25     spider = RenrenLogin()
26     spider.get_html()
```

## ■ 方法二 - 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 |
```

## ▪ 方法三

### 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
```

## 代码实现

```
1 |
```

## json解析模块

## json.loads(json)

### ▪ 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.dumps(python)

### ▪ 作用

```
1 | 把 python 类型 转为 json 类型
```

### ▪ 示例

```
1 | import json
2 |
3 | # json.dumps()之前
4 | item = {'name':'QQ','app_id':1}
5 | print('before dumps',type(item)) # dict
6 | # json.dumps之后
7 | item = json.dumps(item)
8 | print('after dumps',type(item)) # str
```

## json.load(f)

### 作用

```
1 | 将json文件读取,并转为python类型
```

### 示例

```
1 | import json
2 |
3 | with open('D:\\spider_test\\xiaomi.json','r') as f:
4 |     data = json.load(f)
5 |
6 | print(data)
```

## 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



```
1 import json
2
3 item = {'name':'QQ','app_id':1}
4 with open('小米.json','a') as f:
5     json.dump(item,f,ensure_ascii=False)
```

## ■ 示例2

```
1 import json
2
3 item_list = []
4 for i in range(3):
5     item = {'name':'QQ','id':i}
6     item_list.append(item)
7
8 with open('xiaomi.json','a') as f:
9     json.dump(item_list,f,ensure_ascii=False)
```

练习: 将腾讯招聘数据存入到json文件

## json模块总结

```
1 # 爬虫最常用
2 1、数据抓取 - json.loads(html)
3     将响应内容由: json 转为 python
4 2、数据保存 - json.dump(item_list,f,ensure_ascii=False)
5     将抓取的数据保存到本地 json文件
6
7 # 抓取数据一般处理方式
8 1、txt文件
9 2、csv文件
10 3、json文件
11 4、MySQL数据库
12 5、MongoDB数据库
13 6、Redis数据库
```

## 今日作业

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