

Day04回顾

■ requests.get()参数

```
1  【1】 url
2  【2】 params -> {} : 查询参数 Query String
3  【3】 proxies -> {}
4      proxies = {
5          'http': 'http://1.1.1.1:8888',
6          'https': 'https://1.1.1.1:8888'
7      }
8  【4】 verify -> True/False, 当程序中抛出 SSLError 时添加 verify=False
9  【5】 timeout
10 【6】 headers
11 【7】 cookies
```

■ requests.post()

```
1  data : 字典, Form表单数据
```

■ 常见的反爬机制及处理方式

```
1  【1】 Headers反爬虫
2      1.1) 检查: Cookie、Referer、User-Agent
3      1.2) 解决方案: 通过F12获取headers,传给requests.get()方法
4
5  【2】 IP限制
6      2.1) 网站根据IP地址访问频率进行反爬,短时间内限制IP访问
7      2.2) 解决方案:
8          a) 构造自己IP代理池,每次访问随机选择代理,经常更新代理池
9          b) 购买开放代理或私密代理IP
10         c) 降低爬取的速度
11
12 【3】 User-Agent限制
13     3.1) 类似于IP限制,检测频率
14     3.2) 解决方案: 构造自己的User-Agent池,每次访问随机选择
15         a> fake_useragent模块
16         b> 新建py文件,存放大量User-Agent
17         c> 程序中定义列表,存放大量的User-Agent
18
19 【4】 对响应内容做处理
20     4.1) 页面结构和响应内容不同
21     4.2) 解决方案: 打印并查看响应内容,用xpath或正则做处理
```

■ 有道翻译过程梳理

- 1 【1】打开首页
- 2
- 3 【2】准备抓包：F12开启控制台
- 4
- 5 【3】寻找地址
- 6 3.1) 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
- 7 F12-Network-XHR-Headers-General-Request URL
- 8
- 9 【4】发现规律
- 10 4.1) 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
- 11 4.2) 分析对比 Network - All(或者XHR) - Form Data，发现对应的规律
- 12
- 13 【5】寻找JS加密文件
- 14 5.1) 控制台右上角 ...->Search->搜索关键字->单击->跳转到Sources，左下角格式化符号{}
- 15
- 16 【6】查看JS代码
- 17 6.1) 搜索关键字，找到相关加密方法，用python实现加密算法
- 18
- 19 【7】断点调试
- 20 7.1) JS代码中部分参数不清楚可通过断点调试来分析查看
- 21
- 22 【8】完善程序

Day05笔记

有道翻译步骤梳理

- 1、开启F12抓包，找到Form表单数据如下：

```
1 i: 喵喵叫
2 from: AUTO
3 to: AUTO
4 smartresult: dict
5 client: fanyideskweb
6 salt: 15614112641250
7 sign: 94008208919faa19bd531acde36aac5d
8 ts: 1561411264125
9 bv: f4d62a2579ebb44874d7ef93ba47e822
10 doctype: json
11 version: 2.1
12 keyfrom: fanyi.web
13 action: FY_BY_REALTIME
```

- 2、在页面中多翻译几个单词，观察Form表单数据变化

```
1 salt: 15614112641250
2 sign: 94008208919faa19bd531acde36aac5d
3 ts: 1561411264125
4 bv: f4d62a2579ebb44874d7ef93ba47e822
5 # 但是bv的值不变
```

■ 3、一般为本地js文件加密，刷新页面，找到js文件并分析JS代码

```
1 # 方法1
2 Network - JS选项 - 搜索关键词salt
3 # 方法2
4 控制台右上角 - Search - 搜索salt - 查看文件 - 格式化输出
5
6 # 最终找到相关JS文件 : fanyi.min.js
```

■ 4、打开JS文件，分析加密算法，用Python实现

```
1 # ts : 经过分析为13位的时间戳，字符串类型
2 js代码实现: "" + (new Date).getTime()
3 python实现: str(int(time.time()*1000))
4
5 # salt
6 js代码实现: ts + parseInt(10 * Math.random(), 10);
7 python实现: ts + str(random.randint(0,9))
8
9 # sign (设置断点调试，来查看 e 的值，发现 e 为要翻译的单词)
10 js代码实现: n.md5("fanyideskweb" + e + salt + "n%A-rKaT5fb[Gy?;N5@Tj")
11 python实现:
12 from hashlib import md5
13 s = md5()
14 s.update('').encode())
15 sign = s.hexdigest()
```

■ 5、pycharm中正则处理headers和formdata

```
1 1、pycharm进入方法 : Ctrl + r , 选中 Regex
2 2、处理headers和formdata
3 (.?): (.*)
4 "$1": "$2",
5 3、点击 Replace All
```

民政部网站数据抓取

■ 目标

```
1 【1】URL: http://www.mca.gov.cn/ - 民政数据 - 行政区划代码
2 即: http://www.mca.gov.cn/article/sj/xzqh/2020/
3
4 【2】目标: 抓取最新中华人民共和国县以上行政区划代码
```

■ 实现步骤

```
1  【1】从民政数据网站中提取最新行政区划代码链接
2    1.1) 新的在上面第2个
3    1.2) xpath表达式: //table//tr[2]/td[2]/a/@href
4
5
6  【2】从二级页面响应内容中提取真实链接
7    2.1) 反爬 - 响应内容中嵌入JS, 指向新的链接
8    2.2) 打印响应内容, 搜索真实链接URL, 找到位置
9    2.3) 正则匹配: window.location.href="(.*?)"
10
11 【3】从真实链接中提取所需数据
12    3.1) 基准xpath(以响应内容为主): //table/tr[2]/td[2]/a/@href
13    3.2) for循环依次遍历提取数据
14          编码: ./td[2]/text() | ./td[2]/span/text()
15          名称: ./td[3]/text()
16
17 【4】扩展 - 补充
18    4.1) 数据存入到 MySQL 数据库, 一定要分表存储
19    4.2) 三张表
20          a> 省表(province) : 名称 编号
21          b> 市表(city)      : 名称 编号 对应省的编号
22          c> 县表(county)    : 名称 编号 对应市的编号
23
24 【5】MySQL建库建表语句
25 create database govdb charset utf8;
26 use govdb;
27 create table province(
28 id int primary key auto_increment,
29 pname varchar(50),
30 pcode varchar(20)
31 )charset=utf8;
32 create table city(
33 id int primary key auto_increment,
34 cname varchar(50),
35 ccode varchar(20),
36 cfcode varchar(20)
37 )charset=utf8;
38 create table county(
39 id int primary key auto_increment,
40 xname varchar(50),
41 xcode varchar(20),
42 xfcode varchar(20)
43 )charset=utf8;
```

■ 代码实现 - 使用redis实现增量

```
1 import requests
2 from lxml import etree
3 import re
4 import redis
5 from hashlib import md5
6 import pymysql
7 import sys
8
```

```

9 class GovementSpider(object):
10     def __init__(self):
11         self.index_url = 'http://www.mca.gov.cn/article/sj/xzqh/2020/'
12         self.headers = {
13             "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/80.0.3987.149 Safari/537.36",
14         }
15         # redis指纹增量
16         self.r = redis.Redis(host='localhost',port=6379,db=0)
17         # MySQL相关变量
18         self.db = pymysql.connect('localhost', 'root', '123456', 'govdb', charset='utf8')
19         self.cursor = self.db.cursor()
20         # 插入语句
21         self.ins1 = 'insert into province(pname,pcode) values(%s,%s)'
22         self.ins2 = 'insert into city(cname,ccode,cfcode) values(%s,%s,%s)'
23         self.ins3 = 'insert into county(xname,xcode,xfcode) values(%s,%s,%s)'
24
25     def get_html(self,url):
26         """请求功能函数"""
27         html = requests.get(url=url,headers=self.headers).text
28
29         return html
30
31     def xpath_func(self, html, xpath_bds):
32         """解析功能函数"""
33         p = etree.HTML(html)
34         r_list = p.xpath(xpath_bds)
35
36         return r_list
37
38     def md5_url(self,url):
39         """URL加密函数"""
40         s = md5()
41         s.update(url.encode())
42
43         return s.hexdigest()
44
45     def get_false_url(self):
46         """获取最新月份链接 - 假链接"""
47         html = self.get_html(self.index_url)
48         print(html)
49         # 解析提取最新月份链接 - 假链接
50         one_xpath = '//table/tr[2]/td[2]/a/@href'
51         false_href_list = self.xpath_func(html,one_xpath)
52         if false_href_list:
53             false_href = false_href_list[0]
54             false_url = 'http://www.mca.gov.cn' + false_href
55             # 生成指纹
56             finger = self.md5_url(false_url)
57             # redis集合增量判断
58             if self.r.sadd('govspider:fingers',finger):
59                 self.get_real_url(false_url)
60             else:
61                 sys.exit('数据已是最新')
62         else:
63             print('提取最新月份链接失败')
64

```

```

65 def get_real_url(self, false_url):
66     """获取真链接"""
67     # 嵌入JS执行URL跳转,提取真实链接
68     html = self.get_html(false_url)
69     regex = r'window.location.href="(.*?)"'
70     pattern = re.compile(regex, re.S)
71     true_url_list = pattern.findall(html)
72     if true_url_list:
73         true_url = true_url_list[0]
74         # 提取具体的数据
75         self.get_data(true_url)
76     else:
77         print('提取真实链接失败')
78
79 def get_data(self, true_url):
80     """提取具体的数据"""
81     html = self.get_html(true_url)
82     # xpath提取数据
83     two_xpath = '//tr[@height="19"]'
84     tr_list = self.xpath_func(html, two_xpath)
85     # 在存入新数据之前先清空数据库
86     self.delete()
87     for tr in tr_list:
88         code_list = tr.xpath('./td[2]/text() | ./td[2]/span/text()')
89         name_list = tr.xpath('./td[3]/text()')
90         code = code_list[0].strip() if code_list else None
91         name = name_list[0].strip() if name_list else None
92         print(name, code)
93         # 将所抓数据存入MySQL数据库
94         if code[-4:] == '0000':
95             self.insert(self.ins1, [name, code])
96             # 四个直辖市特殊,需要存到市表中一份
97             if name in ['北京市', '天津市', '上海市', '重庆市']:
98                 self.insert(self.ins2, [name, code, code])
99             elif code[-2:] == '00':
100                 self.insert(self.ins2, [name, code, code[:2] + '0000'])
101             # 记录最近1次城市的编号
102             last_city = code
103         else:
104             if code[:2] in ['11', '12', '31', '50']:
105                 xfcode = code[:2] + '0000'
106             else:
107                 xfcode = last_city
108             self.insert(self.ins3, [name, code, xfcode])
109
110
111 def delete(self):
112     """删除表记录功能函数"""
113     del1 = 'delete from province;'
114     del2 = 'delete from city;'
115     del3 = 'delete from county;'
116     self.cursor.execute(del1)
117     self.cursor.execute(del2)
118     self.cursor.execute(del3)
119     self.db.commit()
120
121 def insert(self, ins, li):

```

```

122         """存入MySQL功能函数"""
123         self.cursor.execute(ins, li)
124         self.db.commit()
125
126     def run(self):
127         """程序入口函数"""
128         self.get_false_url()
129
130 if __name__ == '__main__':
131     spider = GovementSpider()
132     spider.run()

```

动态加载数据抓取-Ajax

■ 特点

- 1 【1】右键 -> 查看网页源码中没有具体数据
- 2 【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 【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 2.1) 控制台中 XHR : 异步加载的数据包
- 4 2.2) XHR -> QueryStringParameters(查询参数)

豆瓣电影数据抓取案例

■ 目标

- 1 【1】地址: 豆瓣电影 - 排行榜 - 剧情
- 2 【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 # 抓取的查询参数如下:
- 4 type: 13 # 电影类型
- 5 interval_id: 100:90
- 6 action: ''
- 7 start: 0 # 每次加载电影的起始索引值 0 20 40 60
- 8 limit: 20 # 每次加载的电影数量
- 9
- 10 【3】URL
- 11 https://movie.douban.com/j/chart/top_list?type=13&interval_id=100%3A90&action=&start={}&limit=20

■ 代码实现 - 全站抓取

```
1 import requests
2 import time
3 import random
4 import re
5 from fake_useragent import UserAgent
6
7 class DoubanSpider(object):
8     def __init__(self):
9         self.url = 'https://movie.douban.com/j/chart/top_list?'
10        self.i = 0
11
12        # 获取随机headers
13        def get_headers(self):
14            headers = {'User-Agent':UserAgent().random }
15
16            return headers
17
18        # 获取页面
19        def get_page(self,params):
20            # 返回 python 数据类型
21            html = requests.get(url=self.url,params=params,headers=self.get_headers()).json()
22            self.parse_page(html)
23
24        # 解析并保存数据
25        def parse_page(self,html):
26            item = {}
27            # html为大列表 [{电影1信息},{},{}]
28            for one in html:
29                # 名称 + 评分
30                item['name'] = one['title'].strip()
31                item['score'] = float(one['score'].strip())
32                # 打印测试
33                print(item)
34                self.i += 1
35
36        # 主函数
37        def run(self):
38            # 获取type的值
39            type_dict = self.get_all_type_films()
40            # 生成菜单
41            menu = ''
42            for key in type_dict:
43                menu += key + '|'
44
45            menu = menu + '\n请做出你的选择:'
46            name = input(menu)
47            type_number = type_dict[name]
48            # 获取电影总数
49            total = self.total_number(type_number)
50            for start in range(0,(total+1),20):
51                params = {
52                    'type' : type_number,
53                    'interval_id' : '100:90',
54                    'action' : '',
```



```

55         'start' : str(start),
56         'limit' : '20'
57     }
58     # 调用函数,传递params参数
59     self.get_page(params)
60     # 随机休眠1-3秒
61     time.sleep(random.randint(1,3))
62     print('电影数量:',self.i)
63
64     # 获取电影总数
65     def total_number(self,type_number):
66         # F12抓包抓到的地址
67         url = 'https://movie.douban.com/j/chart/top_list_count?type=
68         {}&interval_id=100%3A90'.format(type_number)
69         headers = self.get_headers()
70         html = requests.get(url=url,headers=headers).json()
71         total = int(html['total'])
72
73         return total
74
75     # 获取所有电影的名字和对应type值
76     def get_all_type_films(self):
77         # 获取 类型和类型码
78         url = 'https://movie.douban.com/chart'
79         headers = self.get_headers()
80         html = requests.get(url=url,headers=headers).text
81         re_bds = r'<a href=.*?type_name=(.*?)&type=(.*?)&.*?>'
82         pattern = re.compile(re_bds,re.S)
83         r_list = pattern.findall(html)
84         # 存放所有类型和对应类型码大字典
85         type_dict = {}
86         for r in r_list:
87             type_dict[r[0].strip()] = r[1].strip()
88
89         return type_dict
90
91 if __name__ == '__main__':
92     spider = DoubanSpider()
93     spider.run()

```

json解析模块

■ json.loads(json)

- | | |
|---|-----------------------------------|
| 1 | 【1】作用：把json格式的字符串转为Python数据类型 |
| 2 | |
| 3 | 【2】示例：html = json.loads(res.text) |

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

- | | |
|---|--------------------------------------------------|
| 1 | 【1】作用 |
| 2 | 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 |

```

3
4 【2】 参数说明
5     2.1) 第1个参数: python类型的数据(字典, 列表等)
6     2.2) 第2个参数: 文件对象
7     2.3) 第3个参数: ensure_ascii=False 序列化时编码
8
9 【3】 示例代码
10 # 示例1
11 import json
12
13 item = {'name':'QQ','app_id':1}
14 with open('小米.json','a') as f:
15     json.dump(item,f,ensure_ascii=False)
16
17 # 示例2
18 import json
19
20 item_list = []
21 for i in range(3):
22     item = {'name':'QQ','id':i}
23     item_list.append(item)
24
25 with open('xiaomi.json','a') as f:
26     json.dump(item_list,f,ensure_ascii=False)

```

■ json.dumps(python)

```

1 【1】 作用 : 把 python 类型 转为 json 格式的字符串
2
3 【2】 示例
4 import json
5
6 # json.dumps()之前
7 item = {'name':'QQ','app_id':1}
8 print('before dumps',type(item)) # dict
9 # json.dumps之后
10 item = json.dumps(item)
11 print('after dumps',type(item)) # str

```

■ json.load(f)

```

1 【1】 作用 : 将json文件读取,并转为python类型
2
3 【2】 示例
4 import json
5 with open('D:/spider_test/xiaomi.json','r') as f:
6     data = json.load(f)
7
8 print(data)

```

■ 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
```

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 【1】导入模块
2     from queue import Queue
3
4 【2】使用
5     q = Queue()
6     q.put(url)
7     q.get() # 当队列为空时, 阻塞
8     q.empty() # 判断队列是否为空, True/False
9
10 【3】q.get()解除阻塞方式
11     3.1) q.get(block=False)
12     3.2) q.get(block=True, timeout=3)
13     3.3) if not q.empty():
14         q.get()
```

■ 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
```

```
10 t_list = []
11
12 for i in range(5):
13     t = Thread(target=函数名)
14     t_list.append(t)
15     t.start()
16
17 for t in t_list:
18     t.join()
```

今日作业

- 1 **【1】肯德基餐厅门店信息抓取 (POST请求练习, 非多线程)**
 - 2 1.1) URL地址: <http://www.kfc.com.cn/kfccda/storelist/index.aspx>
 - 3 1.2) 所抓数据: 餐厅编号、餐厅名称、餐厅地址、城市
 - 4 1.3) 数据存储: 请保存到本地json文件中: kfc.json
 - 5 1.4) 程序运行效果:
 - 6 请输入城市名: 北京
 - 7 会把北京所有肯德基门店信息保存到 kfc.json 中
- 8
- 9 **【2】小米应用商店数据抓取 - 多线程**
 - 10 2.1) 网址: 百度搜 - 小米应用商店, 进入官网 <http://app.mi.com/>
 - 11 2.2) 目标: 抓取聊天社交分类下的
 - 12 a> 应用名称
 - 13 b> 应用链接
- 14
- 15 **【3】腾讯招聘职位信息抓取**
 - 16 1) 网址: 腾讯招聘官网 - 职位信息 <https://careers.tencent.com/search.html>
 - 17 2) 目标: 所有职位的如下信息:
 - 18 a> 职位名称
 - 19 b> 职位地址
 - 20 c> 职位类别 (技术类、销售类...)
 - 21 d> 发布时间
 - 22 e> 工作职责
 - 23 f> 工作要求
 - 24 3) 最终信息详情要通过二级页面拿到, 因为二级页面信息很全, 而一级页面信息不全(无工作要求)
 - 25 4) 可以不使用多线程
 - 26 假如说你想要使用多线程, 则思考一下: 是否需要两个队列, 分别存储一级页面的URL地址和二级的