

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

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
```

■ 代码实现 - 全站抓取

```
1  """
2  豆瓣电影 - 全站抓取
3  """
4  import requests
5  from fake_useragent import UserAgent
6  import time
7  import random
8  import re
9  import json
10
11  class DoubanSpider:
12      def __init__(self):
13          self.url = 'https://movie.douban.com/j/chart/top_list?'
14          self.i = 0
15          # 存入json文件
16          self.f = open('douban.json', 'w', encoding='utf-8')
17          self.all_film_list = []
18
19      def get_agent(self):
20          """获取随机的User-Agent"""
21          return UserAgent().random
22
23      def get_html(self, params):
24          headers = {'User-Agent': self.get_agent()}
25          html = requests.get(url=self.url, params=params, headers=headers).text
26          # 把json格式的字符串转为python数据类型
27          html = json.loads(html)
28
29          self.parse_html(html)
30
31      def parse_html(self, html):
32          """解析"""
33          # html: [{},{},{},{}]
34          item = {}
35          for one_film in html:
36              item['rank'] = one_film['rank']
37              item['title'] = one_film['title']
38              item['score'] = one_film['score']
39              print(item)
40              self.all_film_list.append(item)
41              self.i += 1
42
43      def run(self):
```

```

44     # d: {'剧情': '11', '爱情': '13', '喜剧': '5', ..., ...}
45     d = self.get_d()
46     # 1、给用户提示,让用户选择
47     menu = ''
48     for key in d:
49         menu += key + '|'
50     print(menu)
51     choice = input('请输入电影类别: ')
52     if choice in d:
53         code = d[choice]
54         # 2、total: 电影总数
55         total = self.get_total(code)
56         for start in range(0, total, 20):
57             params = {
58                 'type': code,
59                 'interval_id': '100:90',
60                 'action': '',
61                 'start': str(start),
62                 'limit': '20'
63             }
64             self.get_html(params=params)
65             time.sleep(random.randint(1, 2))
66
67         # 把数据存入json文件
68         json.dump(self.all_film_list, self.f, ensure_ascii=False)
69         self.f.close()
70         print('数量:', self.i)
71     else:
72         print('请做出正确的选择')
73
74     def get_d(self):
75         """{'剧情': '11', '爱情': '13', '喜剧': '5', ..., ...}"""
76         url = 'https://movie.douban.com/chart'
77         html = requests.get(url=url, headers={'User-Agent': self.get_agent()}).text
78         regex = '<span><a href=".*?type_name=(.*?)&type=(.*?)&interval_id=100:90&action=">'
79         pattern = re.compile(regex, re.S)
80         # r_list: [('剧情', '11'), ('喜剧', '5'), ('爱情', '13')... ..]
81         r_list = pattern.findall(html)
82         # d: {'剧情': '11', '爱情': '13', '喜剧': '5', ..., ...}
83         d = {}
84         for r in r_list:
85             d[r[0]] = r[1]
86
87         return d
88
89     def get_total(self, code):
90         """获取某个类别下的电影总数"""
91         url = 'https://movie.douban.com/j/chart/top_list_count?type=
92         {}&interval_id=100%3A90'.format(code)
93         html = requests.get(url=url, headers={'User-Agent': self.get_agent()}).text
94         html = json.loads(html)
95
96         return html['total']
97
98 if __name__ == '__main__':
99     spider = DoubanSpider()
100     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
```

execjs

■ 安装

```
1  【1】Linux
2      1.1) 首先安装nodejs执行环境：sudo apt-get install nodejs
3      1.2) 然后安装execjs模块：sudo pip3 install pyexecjs
4
5  【2】Windows
6      python -m pip install pyexecjs
```

■ 使用说明

```
1  【1】作用
2      python中执行js代码, js逆向解决反爬
3
4  【2】使用流程
5      2.1) 导入模块: import pyexecjs
6      2.2) 读取js文件的js代码
7      2.3) 创建编译对象: loader = execjs.compile(js代码)
8      2.4) 执行js代码: loader.call('js中函数名', '函数参数')
```

■ 使用示例1

```

1 import execjs
2
3 js_data = """
4     function test(name){
5         return "Hello, " + name;
6     }
7 """
8 loader = execjs.compile(js_data)
9 result = loader.call("test", "张三丰")
10 print(result)

```

■ 使用示例2

```

1 # output.js
2 function test(name){
3     return "Hello, " + name;
4 }
5
6 # output.py
7 import execjs
8
9 with open('output.js', 'r') as f:
10     js_data = f.read()
11
12 loader = execjs.compile(js_data)
13 result = loader.call("test", "张三丰")
14 print(result)

```

JS逆向 - 百度翻译破解案例

■ 目标

1 破解百度翻译接口，抓取翻译结果数据

■ 实现步骤

1. F12抓包,找到json的地址,观察查询参数

```

1 1、POST地址: https://fanyi.baidu.com/v2transapi
2 2、Form表单数据 (多次抓取在变的字段)
3 from: zh
4 to: en
5 sign: 54706.276099 #这个是如何生成的?
6 token: a927248ae7146c842bb4a94457ca35ee # 固定不变

```

2. 抓取相关JS文件

1 右上角 - 搜索 - sign: - 找到具体JS文件 - 格式化输出

3. 在JS中寻找sign的生成代码

- 1 1、在格式化输出的JS代码中搜索: sign: 找到如下JS代码: sign: y(n),
- 2 2、通过设置断点, 找到y(n)函数的位置, 即生成sign的具体函数
- 3 2.1) n为要翻译的单词
- 4 2.2) 鼠标移动到 y(n) 位置处, 点击可进入具体y(n)函数代码块

4. 生成sign的m(a)函数具体代码如下(在一个大的define中)

```

1 function a(r) {
2     if (Array.isArray(r)) {
3         for (var o = 0, t = Array(r.length); o < r.length; o++)
4             t[o] = r[o];
5         return t
6     }
7     return Array.from(r)
8 }
9 function n(r, o) {
10     for (var t = 0; t < o.length - 2; t += 3) {
11         var a = o.charAt(t + 2);
12         a = a >= "a" ? a.charCodeAt(0) - 87 : Number(a),
13         a = "+" === o.charAt(t + 1) ? r >>> a : r << a,
14         r = "+" === o.charAt(t) ? r + a & 4294967295 : r ^ a
15     }
16     return r
17 }
18 function e(r) {
19     // 断点调试, 发现i的值不变, 所以在此处定义, 否则运行时会报错: i 未定义
20     var i = "320305.131321201";
21     var o = r.match(/[\uD800-\uDBFF][\uDC00-\uDFFF]/g);
22     if (null === o) {
23         var t = r.length;
24         t > 30 && (r = "" + r.substr(0, 10) + r.substr(Math.floor(t / 2) - 5, 10) +
25             r.substr(-10, 10))
26     } else {
27         for (var e = r.split(/[\uD800-\uDBFF][\uDC00-\uDFFF]/), C = 0, h = e.length, f = []; h
28             > C; C++)
29             "" !== e[C] && f.push.apply(f, a(e[C].split(""))),
30             C !== h - 1 && f.push(o[C]);
31         var g = f.length;
32         g > 30 && (r = f.slice(0, 10).join("") + f.slice(Math.floor(g / 2) - 5, Math.floor(g /
33             2) + 5).join("") + f.slice(-10).join(""))
34     }
35     var u = void 0
36     , l = "" + String.fromCharCode(103) + String.fromCharCode(116) + String.fromCharCode(107);
37     u = null !== i ? i : (i = window[l] || "") || "";
38     for (var d = u.split("."), m = Number(d[0]) || 0, s = Number(d[1]) || 0, S = [], c = 0, v
39         = 0; v < r.length; v++) {
40         var A = r.charCodeAt(v);
41         128 > A ? S[c++] = A : (2048 > A ? S[c++] = A >> 6 | 192 : (55296 === (64512 & A) && v
42             + 1 < r.length && 56320 === (64512 & r.charCodeAt(v + 1)) ? (A = 65536 + ((1023 & A) << 10) +
43             (1023 & r.charCodeAt(++v)),
44             S[c++] = A >> 18 | 240,
45             S[c++] = A >> 12 & 63 | 128) : S[c++] = A >> 12 | 224,
46             S[c++] = A >> 6 & 63 |
47             128),
48             S[c++] = 63 & A | 128)

```



```

42     }
43     for (var p = m, F = "" + String.fromCharCode(43) + String.fromCharCode(45) +
String.fromCharCode(97) + ("" + String.fromCharCode(94) + String.fromCharCode(43) +
String.fromCharCode(54)), D = "" + String.fromCharCode(43) + String.fromCharCode(45) +
String.fromCharCode(51) + ("" + String.fromCharCode(94) + String.fromCharCode(43) +
String.fromCharCode(98)) + ("" + String.fromCharCode(43) + String.fromCharCode(45) +
String.fromCharCode(102)), b = 0; b < S.length; b++)
44         p += S[b],
45         p = n(p, F);
46     return p = n(p, D),
47         p ^= s,
48         0 > p && (p = (2147483647 & p) + 2147483648),
49         p %= 1e6,
50         p.toString() + "." + (p ^ m)
51 }

```

5. 直接将4中代码写入本地translate.js文件,利用pyexecjs模块执行js代码进行调试

```

1  # test_translate.py
2  import execjs
3
4  with open('translate.js', 'r', encoding='utf-8') as f:
5      js_code = f.read()
6
7  obj = execjs.compile(js_code)
8  print(obj.call('e', 'python'))

```

6. 代码实现

```

1  import requests
2  import execjs
3
4  class BaiduTranslateSpider:
5      def __init__(self):
6          self.url = 'https://fanyi.baidu.com/v2transapi?from=en&to=zh'
7          self.headers = {
8              'Cookie': 'BIDUPSID=46D0471B72D849FC7EDF21BA4702F83C; PSTM=1587698693;
BAIDUID=46D0471B72D849FCE9A270A451DF87D1:FG=1; BDORZ=B490B5EBF6F3CD402E515D22BCDA1598;
delPer=0; PSINO=2; H_PS_PSSID=30969_1463_31326_21107_31427_31341_31228_30824_31164;
Hm_lvt_64ecd82404c51e03dc91cb9e8c025574=1587727393; REALTIME_TRANS_SWITCH=1;
FANYI_WORD_SWITCH=1; HISTORY_SWITCH=1; SOUND_SPD_SWITCH=1; SOUND_PREFER_SWITCH=1;
Hm_lpv=64ecd82404c51e03dc91cb9e8c025574=1587727413;
__yjsv5_shitong=1.0_7_6a5f66b7527ef7c72b25325159665a94b252_300_1587727413634_101.30.19.86_2f87
d549; yjs_js_security_passport=81df417e6e29094c4b7fa337affa272f3e7a7bfb_1587727414_js',
9              'Referer': 'https://fanyi.baidu.com/',
10             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/81.0.4044.122 Safari/537.36'
11         }
12
13     def get_sign(self, word):
14         """获取sign"""
15         with open('translate.js', 'r', encoding='utf-8') as f:
16             js_code = f.read()
17
18         obj = execjs.compile(js_code)

```

```

19         sign = obj.call('e', word)
20
21         return sign
22
23     def get_result(self, word):
24         sign = self.get_sign(word)
25         data = {
26             "from": "en",
27             "to": "zh",
28             "query": word,
29             "transtype": "realtime",
30             "simple_means_flag": "3",
31             "sign": sign,
32             "token": "4cf7c952bf4500c7446f7cb3ab40860f",
33             "domain": "common",
34         }
35         html = requests.post(url=self.url, data=data, headers=self.headers).json()
36         result = html['trans_result'][0]['dst']
37
38         return result
39
40     def run(self):
41         word = input('请输入要翻译的单词:')
42         print(self.get_result(word))
43
44 if __name__ == '__main__':
45     spider = BaiduTranslateSpider()
46     spider.run()

```

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():

```

■ 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()
```

■ 多线程爬虫示例代码

```
1 # 抓取豆瓣电影剧情类别下的电影信息
2 """
3 豆瓣电影 - 剧情 - 抓取
4 """
5 import requests
6 from fake_useragent import UserAgent
7 import time
8 import random
9 from threading import Thread, Lock
10 from queue import Queue
11
12 class DoubanSpider:
13     def __init__(self):
14         self.url = 'https://movie.douban.com/j/chart/top_list?
15         type=13&interval_id=100%3A90&action=&start={}&limit=20'
16         self.i = 0
17         # 队列 + 锁
18         self.q = Queue()
19         self.lock = Lock()
20
21     def get_agent(self):
22         """获取随机的User-Agent"""
23         return UserAgent().random
24
25     def url_in(self):
26         """把所有要抓取的URL地址入队列"""
27         for start in range(0, 684, 20):
28             url = self.url.format(start)
29             # url入队列
30             self.q.put(url)
```

```

30
31 # 线程事件函数: 请求+解析+数据处理
32 def get_html(self):
33     while True:
34         # 从队列中获取URL地址
35         # 一定要在判断队列是否为空 和 get() 地址 前后加锁,防止队列中只剩一个地址时出现重复判断
36         self.lock.acquire()
37         if not self.q.empty():
38             headers = {'User-Agent': self.get_agent()}
39             url = self.q.get()
40             self.lock.release()
41
42             html = requests.get(url=url, headers=headers).json()
43             self.parse_html(html)
44         else:
45             # 如果队列为空,则最终必须释放锁
46             self.lock.release()
47             break
48
49 def parse_html(self, html):
50     """解析"""
51     # html: [{},{},{},{}]
52     item = {}
53     for one_film in html:
54         item['rank'] = one_film['rank']
55         item['title'] = one_film['title']
56         item['score'] = one_film['score']
57         print(item)
58         # 加锁 + 释放锁
59         self.lock.acquire()
60         self.i += 1
61         self.lock.release()
62
63 def run(self):
64     # 先让URL地址入队列
65     self.url_in()
66     # 创建多个线程,开干吧
67     t_list = []
68     for i in range(1):
69         t = Thread(target=self.get_html)
70         t_list.append(t)
71         t.start()
72
73     for t in t_list:
74         t.join()
75
76     print('数量:',self.i)
77
78 if __name__ == '__main__':
79     start_time = time.time()
80     spider = DoubanSpider()
81     spider.run()
82     end_time = time.time()
83     print('执行时间:%.2f' % (end_time-start_time))

```

今日作业

【1】肯德基餐厅门店信息抓取 (POST请求练习, 非多线程)

1.1) URL地址: <http://www.kfc.com.cn/kfccda/storelist/index.aspx>

1.2) 所抓数据: 餐厅编号、餐厅名称、餐厅地址、城市

1.3) 数据存储: 请保存到本地json文件中: kfc.json

1.4) 程序运行效果:

请输入城市名: 北京

会把北京所有肯德基门店信息保存到 kfc.json 中

【2】小米应用商店数据抓取 - 多线程

2.1) 网址: 百度搜 - 小米应用商店, 进入官网 <http://app.mi.com/>

2.2) 目标: 抓取聊天社交分类下的

a> 应用名称

b> 应用链接

【3】腾讯招聘职位信息抓取

1) 网址: 腾讯招聘官网 - 职位信息 <https://careers.tencent.com/search.html>

2) 目标: 所有职位的如下信息:

a> 职位名称

b> 职位地址

c> 职位类别 (技术类、销售类...)

d> 发布时间

e> 工作职责

f> 工作要求

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)

4) 可以不使用多线程

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