Day04回顾

requests.get()参数

```
1、url
2、params -> {} : 查询参数 Query String
3、proxies -> {}
   proxies = {
      'http':'http://username:password@1.1.1.1:8888',
      'https':'https://username:password@1.1.1.1:8888'
   }
4、auth -> ('tarenacode','code_2013')
5、verify -> True/False
6、timeout
```

requests.post()

```
data -> {} Form表单数据 : Form Data
```

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.
```
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
```

常见的反爬机制及处理方式

```
1、Headers反爬虫 ：Cookie、Referer、User-Agent
   解决方案：通过F12获取headers,传给requests.get()方法

2、IP限制 ：网站根据IP地址访问频率进行反爬,短时间内进制IP访问
   解决方案：
        1、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
        2、购买开放代理或私密代理IP
        3、降低爬取的速度

3、User-Agent限制 ：类似于IP限制
   解决方案：构造自己的User-Agent池,每次访问随机选择

5、对查询参数或Form表单数据认证(salt、sign)
   解决方案：找到JS文件,分析JS处理方法,用Python按同样方式处理

6、对响应内容做处理
   解决方案：打印并查看响应内容,用xpath或正则做处理
```

python中正则处理headers和formdata

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

Day05笔记

有道翻译代码实现

有道翻译验证了什么？ - headers

```
1、Cookie
2、Referer
3、User-Agent
```

代码实现

```
import requests
import time
import random
from hashlib import md5

class YdSpider(object):
  def __init__(self):
    # url一定为F12抓到的 headers -> General -> Request URL
    self.url = 'http://fanyi.youdao.com/translate_o?smartresult=dict&smartresult=rule'
    self.headers = {
      "Cookie": "OUTFOX_SEARCH_USER_ID=970246104@10.169.0.83; OUTFOX_SEARCH_USER_ID_NCOO=570559528.1224236; _ntes_nnid=96bc13a2f5ce64962adfd6a278467214,1551873108952; JSESSIONID=aaae9i7plXPlKaJH_gkYw; td_cookie=18446744072941336803; SESSION_FROM_COOKIE=unknown; ___rl__test__cookies=1565689460872",
      "Referer": "http://fanyi.youdao.com/",
      "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salt,sign,ts
  def get_salt_sign_ts(self,word):
    # salt
    salt = str(int(time.time()*1000)) + str(random.randint(0,9))
    # sign
    string = "fanyideskweb" + word + salt + "n%A-rKaT5fb[Gy?;N5@Tj"
    s = md5()
    s.update(string.encode())
    sign = s.hexdigest()
    # ts
    ts = str(int(time.time()*1000))
    return salt,sign,ts

  # 主函数
  def attack_yd(self,word):
    # 1. 先拿到salt,sign,ts
    salt, sign, ts = self.get_salt_sign_ts(word)
    # 2. 定义form表单数据为字典: data={}
    data = {
      'i': word,
```

```
        'from': 'AUTO',
        'to': 'AUTO',
        'smartresult': 'dict',
        'client': 'fanyideskweb',
        'salt': salt,
        'sign': sign,
        'ts': ts,
        'bv': 'cf156b581152bd0b259b90070b1120e6',
        'doctype': 'json',
        'version': '2.1',
        'keyfrom': 'fanyi.web',
        'action': 'FY_BY_REALTlME'
    }
    # 3. 直接发请求:requests.post(url,data=data,headers=xxx)
    json_html = requests.post(self.url, data=data, headers=self.headers).json()
    result = json_html['translateResult'][0][0]['tgt']
    return result

  # 主函数
  def main(self):
    # 输入翻译单词
    word = input('请输入要翻译的单词: ')
    result = self.attack_yd(word)
    print('翻译结果:',result)

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

民政部网站数据抓取

目标

```
1、URL: http://www.mca.gov.cn/ - 民政数据 - 行政区划代码
   即: http://www.mca.gov.cn/article/sj/xzqh/2019/
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、从民政数据网站中提取最新行政区划代码链接**

```
# 特点
1、最新的在上面
2、命名格式: 2019年x月中华人民共和国县以上行政区划代码
# 代码实现
import requests
from lxml import etree
import re

url = 'http://www.mca.gov.cn/article/sj/xzqh/2019/'
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36'}
```

```
html = requests.get(url, headers=headers).text
parse_html = etree.HTML(html)
article_list = parse_html.xpath('//a[@class="artitlelist"]')

for article in article_list:
    title = article.xpath('./@title')[0]
    # 正则匹配title中包含这个字符串的链接
    if title.endswith('代码'):
        # 获取到第1个就停止即可，第1个永远是最新的链接
        two_link = 'http://www.mca.gov.cn' + article.xpath('./@href')[0]
        print(two_link)
        break
```

- **2、从二级页面链接中提取真实链接（反爬-响应内容中嵌入JS，指向新的链接）**

```
1、向二级页面链接发请求得到响应内容，并查看嵌入的JS代码
2、正则提取真实的二级页面链接
# 相关思路代码
two_html = requests.get(two_link, headers=headers).text
# 从二级页面的响应中提取真实的链接（此处为JS动态加载跳转的地址）
new_two_link = re.findall(r'window.location.href="(.*?)"', html2, re.S)[0]
```

- **3、在数据库表中查询此条链接是否已经爬取，建立增量爬虫**

```
1、数据库中建立version表，存储爬取的链接
2、每次执行程序和version表中记录核对，查看是否已经爬取过
# 思路代码
cursor.execute('select * from version')
result = self.cursor.fetchall()
if result:
    if result[-1][0] == two_link:
        print('已是最新')
    else:
        # 有更新，开始抓取
        # 将链接再重新插入version表记录
```

- **4、代码实现**

```
'''民政部网站数据抓取（增量爬虫）'''
import requests
from lxml import etree
import re
import pymysql

class Govement(object):
    def __init__(self):
        self.one_url = 'http://www.mca.gov.cn/article/sj/xzqh/2019/'
        self.headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36'}
        self.db = pymysql.connect('192.168.153.138','tiger','123456','govdb')
        self.cursor = self.db.cursor()

    # 获取二级页面真实链接，并和数据库中比对
    def get_flase_link(self):
```

```
        html = requests.get(self.one_url,headers=self.headers).text
        # 此处隐藏了真实的二级页面的url链接，通过js脚本生成，保存本地文件查看
        parse_html = etree.HTML(html)
        a_list = parse_html.xpath('//a[@class="artitlelist"]')
        for a in a_list:
            title = a.get('title')
            # 正则匹配title中包含这个字符串的链接
            if title.endswith('代码'):
                # 获取到第1个就停止即可，第1个永远是最新的链接
                false_link = 'http://www.mca.gov.cn' + a.get('href')
                break

        # 提取二级页面真实链接
        self.get_real_link(false_link)

    def get_real_link(self,false_link):
        # 从已提取的two_link中提取二级页面的真实链接
        two_html = requests.get(false_link, headers=self.headers).text
        # 从二级页面的响应中提取真实的链接（此处为JS动态加载跳转的地址）
        real_two_link = re.findall(r'window.location.href="(.*?)"', two_html, re.S)[0]

        self.incre_spider(real_two_link)

    def incre_spider(self,real_two_link):
        # 实现增量爬取
        self.cursor.execute('select * from version')
        result = self.cursor.fetchall()
        if result:
            if result[0][0] == real_two_link:
                print('已是最新，无须爬取')
        else:
            self.get_data(real_two_link)
            self.cursor.execute('delete from version')
            self.cursor.execute('insert into version values(%s)',[real_two_link])
            self.db.commit()

    # 用xpath直接提取数据
    def get_data(self,real_two_link):
        real_two_html = requests.get(real_two_link,headers=self.headers).text
        parse_html = etree.HTML(real_two_html)
        # 基准xpath,提取每个信息的节点列表对象
        tr_list = parse_html.xpath('//tr[@height=19]')
        city_info = {}
        for tr in tr_list:
            city_info['code'] = tr.xpath('./td[2]/text()')[0]
            city_info['name'] = tr.xpath('./td[3]/text()')[0]
            print(city_info)


if __name__ == '__main__':
    spider = Govement()
    spider.get_flase_link()
```

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
```

- 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
limit: 20 # 每次加载的电影数量
```

- **代码实现**

```python
import requests
import json

class DoubanSpider(object):
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?'
        self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36'}
        self.i = 0

    # 获取页面
    def get_page(self,params):
        res = requests.get(url=self.url,params=params,headers=self.headers,verify=True)
        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
            item['name'] = one['title'].strip()
            # 评分
            item['score'] = float(one['score'].strip())
            # 打印测试
            print(item)
            self.i += 1

    # 主函数
    def main(self):
        for start in range(0,41,20):
            params = {
                'type' : '24',
                'interval_id' : '100:90',
                'action' : '',
                'start' : str(start),
                'limit' : '20'
            }
            # 调用函数,传递params参数
            self.get_page(params)
        print('电影数量:',self.i)

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

练习: 能否抓取指定类型的所有电影信息? - 无须指定数量

```
import requests
import json

class DoubanSpider(object):
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?'
        self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36'}
        self.i = 0

    # 获取页面
    def get_page(self,params):
        res = requests.get(url=self.url,params=params,headers=self.headers,verify=True)
        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
            item['name'] = one['title'].strip()
            # 评分
            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):
        # F12抓包抓到的地址
        url = 'https://movie.douban.com/j/chart/top_list_count?type=24&interval_id=100%3A90'
        html = requests.get(url=url,headers=self.headers,verify=False).json()
        total = int(html['total'])

        return total

    # 主函数
    def main(self):
        # 获取type的值
        name = input('请输入电影类型(剧情|喜剧|爱情):')
        typ_dic = {'剧情':'11','喜剧':'24','爱情':'13'}
        typ = str(typ_dic[name])
        # 获取电影总数
        total = self.total_number()
        for start in range(0,(total+1),20):
            params = {
                'type' : typ,
                'interval_id' : '100:90',
                'action' : '',
                'start' : str(start),
                'limit' : '20'
            }
            # 调用函数,传递params参数
            self.get_page(params)
        print('电影数量:',self.i)

if __name__ == '__main__':
    spider = DoubanSpider()
    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，如下方法你觉得怎么样？？？？？
for i in range(5):
    t = Thread(target=函数名)
    t.start()
    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 ：应用分类 - 聊天社交
    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再变，0 1 2 3 ... ... ，这样我们就可以通过控制page的直拼接多个返回json数据的URL地址
```

- 代码实现

```python
import requests
from threading import Thread
from queue import Queue
import json
import time

class XiaomiSpider(object):
    def __init__(self):
        self.headers = {'User-Agent':'Mozilla/5.0'}
        self.url = 'http://app.mi.com/categotyAllListApi?page={}&categoryId=2&pageSize=30'
        # 定义队列，用来存放URL地址
        self.url_queue = Queue()

    # URL入队列
    def url_in(self):
        # 拼接多个URL地址,然后put()到队列中
        for i in range(67):
            self.url.format((str(i)))
            self.url_queue.put(self.url)

    # 线程事件函数(请求,解析提取数据)
    def get_page(self):
        # 先get()URL地址,发请求
        # json模块做解析
        while True:
            # 当队列不为空时,获取url地址
            if not self.url_queue.empty():
                url = self.url_queue.get()
                html = requests.get(url,headers=self.headers).text
                self.parse_page(html)
            else:
                break
    # 解析函数
    def parse_page(self,html):
        app_json = json.loads(html)
        item = {}
        for app in app_json['data']:
            # 应用名称
            item['name'] = app['displayName']
            # 应用链接
            item['link'] = 'http://app.mi.com/details?id={}'.format(app['packageName'])

            print(item)

```

```python
    # 主函数
    def main(self):
        self.url_in()
        # 存放所有线程的列表
        t_list = []

        for i in range(10):
            t = Thread(target=self.get_page)
            t.start()
            t_list.append(t)

        # 统一回收线程
        for p in t_list:
            p.join()

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

今日作业

```
1、有道翻译案例复写一遍
2、抓取腾讯招聘数据(两级页面 - 职位名称、岗位职责、工作要求)
3、把腾讯招聘案例改写为多线程
4、把链家二手房案例改写为多线程
5、民政部数据抓取案例完善
   # 1、将抓取的数据存入数据库，最好分表按照层级关系去存
   # 2、增量爬取时表中数据也要更新
```