Day04回顾

- **requests.get()参数**

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

- **requests.post()**

```
data : 字典, Form表单数据
```

- **常见的反爬机制及处理方式**

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

【2】IP限制
  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
  2.2) 解决方案:
       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
       b) 购买开放代理或私密代理IP
       c) 降低爬取的速度

【3】User-Agent限制
  3.1) 类似于IP限制, 检测频率
  3.2) 解决方案: 构造自己的User-Agent池,每次访问随机选择
       a> fake_useragent模块
       b> 新建py文件,存放大量User-Agent
       c> 程序中定义列表,存放大量的User-Agent

【4】对响应内容做处理
  4.1) 页面结构和响应内容不同
  4.2) 解决方案: 打印并查看响应内容,用xpath或正则做处理
```

【5】JS加密
5.1) 抓取到对应的JS文件,寻找加密算法
5.2) 用Python实现加密算法,生成指定的参数

- **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
  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
       F12-Network-XHR-Headers-Grneral-Request URL

【4】发现规律
  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
  4.2) 分析对比 Network - All(或者XHR) - Form Data，发现对应的规律

【5】寻找JS加密文件
  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{}

【6】查看JS代码
  6.1) 搜索关键字，找到相关加密方法，用python实现加密算法

【7】断点调试
  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

【8】完善程序
```

Day05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
  2.1) 控制台中 XHR ： 异步加载的数据包
  2.2) 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**

```
"""
豆瓣电影 - 全站抓取
"""
import requests
from fake_useragent import UserAgent
import time
import random
import re
import json

class DoubanSpider:
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?'
        self.i = 0
        # 存入json文件
        self.f = open('douban.json', 'w', encoding='utf-8')
        self.all_film_list = []

    def get_agent(self):
        """获取随机的User-Agent"""
        return UserAgent().random

    def get_html(self, params):
        headers = {'User-Agent':self.get_agent()}
        html = requests.get(url=self.url, params=params, headers=headers).text
        # 把json格式的字符串转为python数据类型
        html = json.loads(html)

        self.parse_html(html)

    def parse_html(self, html):
        """解析"""
        # html: [{},{},{},{}]
        item = {}
        for one_film in html:
            item['rank'] = one_film['rank']
            item['title'] = one_film['title']
            item['score'] = one_film['score']
            print(item)
            self.all_film_list.append(item)
            self.i += 1

    def run(self):
```

```
        # d: {'剧情':'11','爱情':'13','喜剧':'5',...,...}
        d = self.get_d()
        # 1、给用户提示,让用户选择
        menu = ''
        for key in d:
            menu += key + '|'
        print(menu)
        choice = input('请输入电影类别: ')
        if choice in d:
            code = d[choice]
            # 2、total: 电影总数
            total = self.get_total(code)
            for start in range(0,total,20):
                params = {
                    'type': code,
                    'interval_id': '100:90',
                    'action': '',
                    'start': str(start),
                    'limit': '20'
                }
                self.get_html(params=params)
                time.sleep(random.randint(1,2))

            # 把数据存入json文件
            json.dump(self.all_film_list, self.f, ensure_ascii=False)
            self.f.close()
            print('数量:',self.i)
        else:
            print('请做出正确的选择')

    def get_d(self):
        """{'剧情':'11','爱情':'13','喜剧':'5',...,...}"""
        url = 'https://movie.douban.com/chart'
        html = requests.get(url=url,headers={'User-Agent':self.get_agent()}).text
        regex = '<span><a href=".*?type_name=(.*?)&type=(.*?)&interval_id=100:90&action=">'
        pattern = re.compile(regex, re.S)
        # r_list: [('剧情','11'),('喜剧','5'),('爱情':'13')... ...]
        r_list = pattern.findall(html)
        # d: {'剧情': '11', '爱情': '13', '喜剧': '5', ..., ...}
        d = {}
        for r in r_list:
            d[r[0]] = r[1]

        return d

    def get_total(self, code):
        """获取某个类别下的电影总数"""
        url = 'https://movie.douban.com/j/chart/top_list_count?type=
{}&interval_id=100%3A90'.format(code)
        html = requests.get(url=url,headers={'User-Agent':self.get_agent()}).text
        html = json.loads(html)

        return html['total']

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

json解析模块

- **json.loads(json)**

```
【1】作用 ： 把json格式的字符串转为Python数据类型

【2】示例 ： html = json.loads(res.text)
```

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

```
【1】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

【2】参数说明
  2.1) 第1个参数: python类型的数据(字典，列表等)
  2.2) 第2个参数: 文件对象
  2.3) 第3个参数: ensure_ascii=False 序列化时编码

【3】示例代码
    # 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)**

```
【1】作用 ： 把 python 类型 转为 json 格式的字符串

【2】 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)**

```
【1】作用 ： 将json文件读取,并转为python类型

【2】 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
```

execjs

- **安装**

```
【1】Linux
    1.1) 首先安装nodejs执行环境 : sudo apt-get install nodejs
    1.2) 然后安装execjs模块 : sudo pip3 install pyexecjs

【2】Windows
    python -m pip install pyexecjs
```

- **使用说明**

```
【1】作用
    python中执行js代码，js逆向解决反爬

【2】使用流程
    2.1) 导入模块: import pyexecjs
    2.2) 读取js文件的js代码
    2.3) 创建编译对象: loader = execjs.compile(js代码)
    2.4) 执行js代码:   loader.call('js中函数名', '函数参数')
```

- **使用示例1**

```python
import execjs

js_data = """
    function test(name){
        return "Hello, " + name;
    }
"""
loader = execjs.compile(js_data)
result = loader.call("test", "张三丰")
print(result)
```

- **使用示例2**

```python
# output.js
function test(name){
    return "Hello, " + name;
}

# output.py
import execjs

with open('output.js', 'r') as f:
    js_data = f.read()

loader = execjs.compile(js_data)
result = loader.call("test", "张三丰")
print(result)
```

JS逆向 - 百度翻译破解案例

- **目标**

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

2. 抓取相关JS文件

```
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、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、通过设置断点，找到y(n)函数的位置，即生成sign的具体函数
   2.1) n为要翻译的单词
   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)

```
function a(r) {
    if (Array.isArray(r)) {
        for (var o = 0, t = Array(r.length); o < r.length; o++)
            t[o] = r[o];
        return t
    }
    return Array.from(r)
}
function n(r, o) {
    for (var t = 0; t < o.length - 2; t += 3) {
        var a = o.charAt(t + 2);
        a = a >= "a" ? a.charCodeAt(0) - 87 : Number(a),
            a = "+" === o.charAt(t + 1) ? r >>> a : r << a,
            r = "+" === o.charAt(t) ? r + a & 4294967295 : r ^ a
    }
    return r
}
function e(r) {
//    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
    var i = "320305.131321201";
    var o = r.match(/[\uD800-\uDBFF][\uDC00-\uDFFF]/g);
    if (null === o) {
        var t = r.length;
        t > 30 && (r = "" + r.substr(0, 10) + r.substr(Math.floor(t / 2) - 5, 10) + r.substr(-10, 10))
    } else {
        for (var e = r.split(/[\uD800-\uDBFF][\uDC00-\uDFFF]/), C = 0, h = e.length, f = []; h > C; C++)
            "" !== e[C] && f.push.apply(f, a(e[C].split(""))),
                C !== h - 1 && f.push(o[C]);
        var g = f.length;
        g > 30 && (r = f.slice(0, 10).join("") + f.slice(Math.floor(g / 2) - 5, Math.floor(g / 2) + 5).join("") + f.slice(-10).join(""))
    }
    var u = void 0
    , l = "" + String.fromCharCode(103) + String.fromCharCode(116) + String.fromCharCode(107);
    u = null !== i ? i : (i = window[l] || "") || "";
    for (var d = u.split("."), m = Number(d[0]) || 0, s = Number(d[1]) || 0, S = [], c = 0, v = 0; v < r.length; v++) {
        var A = r.charCodeAt(v);
        128 > A ? S[c++] = A : (2048 > A ? S[c++] = A >> 6 | 192 : (55296 === (64512 & A) && v + 1 < r.length && 56320 === (64512 & r.charCodeAt(v + 1)) ? (A = 65536 + ((1023 & A) << 10) + (1023 & r.charCodeAt(++v)),
            S[c++] = A >> 18 | 240,
            S[c++] = A >> 12 & 63 | 128) : S[c++] = A >> 12 | 224,
                                                                    S[c++] = A >> 6 & 63 | 128),
                                S[c++] = 63 & A | 128)
```

```
    }
    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++)
        p += S[b],
            p = n(p, F);
    return p = n(p, D),
        p ^= s,
        0 > p && (p = (2147483647 & p) + 2147483648),
        p %= 1e6,
        p.toString() + "." + (p ^ m)
}
```

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

```
# test_translate.py
import execjs

with open('translate.js', 'r', encoding='utf-8') as f:
    js_code = f.read()

obj = execjs.compile(js_code)
print(obj.call('e', 'python'))
```

6. 代码实现

```
import requests
import execjs

class BaiduTranslateSpider:
    def __init__(self):
        self.url = 'https://fanyi.baidu.com/v2transapi?from=en&to=zh'
        self.headers = {
            '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_lpvt_64ecd82404c51e03dc91cb9e8c025574=1587727413;
__yjsv5_shitong=1.0_7_6a5f66b7527ef7c72b25325159665a94b252_300_1587727413634_101.30.19.86_2f87
d549; yjs_js_security_passport=81df417e6e29094c4b7fa337affa272f3e7a7bfb_1587727414_js',
            'Referer':'https://fanyi.baidu.com/',
            'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/81.0.4044.122 Safari/537.36'
        }

    def get_sign(self, word):
        """获取sign"""
        with open('translate.js', 'r', encoding='utf-8') as f:
            js_code = f.read()

        obj = execjs.compile(js_code)
```

```
        sign = obj.call('e', word)

        return sign

    def get_result(self, word):
        sign = self.get_sign(word)
        data = {
            "from": "en",
            "to": "zh",
            "query": word,
            "transtype": "realtime",
            "simple_means_flag": "3",
            "sign": sign,
            "token": "4cf7c952bf4500c7446f7cb3ab40860f",
            "domain": "common",
        }
        html = requests.post(url=self.url, data=data, headers=self.headers).json()
        result = html['trans_result']['data'][0]['dst']

        return result

    def run(self):
        word = input('请输入要翻译的单词:')
        print(self.get_result(word))

if __name__ == '__main__':
    spider = BaiduTranslateSpider()
    spider.run()
```

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
  from queue import Queue

【2】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

【3】q.get()解除阻塞方式
  3.1) q.get(block=False)
  3.2) q.get(block=True,timeout=3)
  3.3) if not q.empty():
```

```
            q.get()
```

- **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()
```

- **多线程爬虫示例代码**

```
# 抓取豆瓣电影剧情类别下的电影信息
"""
豆瓣电影 - 剧情 - 抓取
"""
import requests
from fake_useragent import UserAgent
import time
import random
from threading import Thread,Lock
from queue import Queue

class DoubanSpider:
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?type=13&interval_id=100%3A90&action=&start={}&limit=20'
        self.i = 0
        # 队列 + 锁
        self.q = Queue()
        self.lock = Lock()

    def get_agent(self):
        """获取随机的User-Agent"""
        return UserAgent().random

    def url_in(self):
        """把所有要抓取的URL地址入队列"""
        for start in range(0,684,20):
            url = self.url.format(start)
            # url入队列
            self.q.put(url)
```

```python

    # 线程事件函数：请求+解析+数据处理
    def get_html(self):
        while True:
            # 从队列中获取URL地址
            # 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
            self.lock.acquire()
            if not self.q.empty():
                headers = {'User-Agent': self.get_agent()}
                url = self.q.get()
                self.lock.release()

                html = requests.get(url=url, headers=headers).json()
                self.parse_html(html)
            else:
                # 如果队列为空,则最终必须释放锁
                self.lock.release()
                break

    def parse_html(self, html):
        """解析"""
        # html: [{},{},{},{}]
        item = {}
        for one_film in html:
            item['rank'] = one_film['rank']
            item['title'] = one_film['title']
            item['score'] = one_film['score']
            print(item)
            # 加锁 + 释放锁
            self.lock.acquire()
            self.i += 1
            self.lock.release()

    def run(self):
        # 先让URL地址入队列
        self.url_in()
        # 创建多个线程,开干吧
        t_list = []
        for i in range(1):
            t = Thread(target=self.get_html)
            t_list.append(t)
            t.start()

        for t in t_list:
            t.join()

        print('数量:',self.i)

if __name__ == '__main__':
    start_time = time.time()
    spider = DoubanSpider()
    spider.run()
    end_time = time.time()
    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
```