Day05回顾

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(查询参数)
```

有道翻译流程梳理

```
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，分析并用python实现
7、断点调试
8、完善程序
```

cookie模拟登陆

```
1、适用网站类型：爬取网站页面时需要登录后才能访问，否则获取不到页面的实际响应数据
2、方法1（利用cookie）
   1、先登录成功1次,获取到携带登陆信息的Cookie（处理headers）
   2、利用处理的headers向URL地址发请求
3、方法2（利用session会话保持）
   1、登陆，找到POST地址：form -> action对应地址
   2、定义字典，创建session实例发送请求
      # 字典key ：<input>标签中name的值(email,password)
      # post_data = {'email':'','password':''}
```

Day06笔记

cookie模拟登录

- 适用网站及场景

```
抓取需要登录才能访问的页面
```

- **方法一**

```
1、先登录成功1次,获取到携带登陆信息的Cookie
   F12打开控制台,在页面输入用户名、密码,登录成功,找到/home(一般在抓到地址的上面)
2、携带着cookie发请求
   ** Cookie
   ** Referer(源,代表你从哪里转过来的)
   ** User-Agent
```

```
import requests
from lxml import etree

# url为需要登录才能正常访问的地址
url = 'http://www.renren.com/969255813/profile'
# headers中的cookie为登录成功后抓取到的cookie
headers = {
    "Accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "zh-CN,zh;q=0.9",
    "Connection": "keep-alive",
    # 此处注意cookie, 要自己抓取
    "Cookie": "",
    "Host": "www.renren.com",
    "Referer": "http://www.renren.com/SysHome.do",
    "Upgrade-Insecure-Requests": "1",
    "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
# 解析
parse_html = etree.HTML(html)
result = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')[0].strip()
# result:就读于中央戏剧学院
print(result)
```

- **方法二**

1. 知识点

```
利用requests模块中的session会话保持功能
```

2. session会话使用流程

```
1、实例化session对象
   session = requests.session()
2、让session对象发送get或者post请求
   res = session.get(url,headers=headers)
```

3. 具体步骤

```
1、寻找登录时POST的地址
   查看网页源码,查看form,找action对应的地址: http://www.renren.com/PLogin.do

2、发送用户名和密码信息到POST的地址
   * 用户名和密码信息以什么方式发送?  -- 字典
     键 : <input>标签中name的值(email,password)
     值 : 真实的用户名和密码
     post_data = {'email':'','password':''}
```

4. 程序实现

```
整体思路
1、先POST: 把用户名和密码信息POST到某个地址中
2、再GET:  正常请求去获取页面信息
```

```python
import requests
from lxml import etree

# 定义常用变量
post_url = 'http://www.renren.com/PLogin.do'
post_data = {
  'email' : 'xxxxxx',
  'password' : 'xxxxxx'
}
headers = {
  'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36',
  'Referer' : 'http://www.renren.com/SysHome.do'
}

# 实例化session会话保持对象
session = requests.session()
# 先POST,把用户名和密码信息POST到一个地址
session.post(post_url,data=post_data,headers=headers)

# 再get个人主页
url = 'http://www.renren.com/970294164/profile'
html = session.get(url,headers=headers).text

parse_html = etree.HTML(html)
result = parse_html.xpath('//*[@id="operate_area"]/div[1]/ul/li[1]/span/text()')[0].strip()
print(result)
```

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 # 基本固定,但也想办法获取
```

- **2、抓取相关JS文件**

```
右上角 - 搜索 - sign: - 找到具体JS文件(index_c8a141d.js) - 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: m(a),
2、通过设置断点，找到m(a)函数的位置，即生成sign的具体函数
   # 1. a 为要翻译的单词
   # 2. 鼠标移动到 m(a) 位置处，点击可进入具体m(a)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) {
    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] || "") || "";
//  断点调试,然后从网页源码中找到 window.gtk的值
    var u = '320305.131321201'

    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)
}
var i = null;
//此行报错，直接注释掉即可
//t.exports = e
```

- **5、直接将代码写入本地js文件,利用pyexecjs模块执行js代码进行调试**

```python
import execjs

with open('node.js','r') as f:
    js_data = f.read()
# 创建对象
exec_object = execjs.compile(js_data)
sign = exec_object.eval('e("hello")')
print(sign)
```

- **获取token**

```
# 在js中
token: window.common.token
# 在响应中想办法获取此值
token_url = 'https://fanyi.baidu.com/?aldtype=16047'
regex: "token: '(.*?)'"
```

- **具体代码实现**

```python
import requests
import re
import execjs

class BaiduTranslateSpider(object):
    def __init__(self):
        self.token_url = 'https://fanyi.baidu.com/?aldtype=16047'
        self.post_url = 'https://fanyi.baidu.com/v2transapi'
        self.headers = {
            'accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3',
            # 'accept-encoding': 'gzip, deflate, br',
            'accept-language': 'zh-CN,zh;q=0.9',
            'cache-control': 'no-cache',
```

```
            'cookie': 'BAIDUID=52920E829C1F64EE98183B703F4E37A9:FG=1;
BIDUPSID=52920E829C1F64EE98183B703F4E37A9; PSTM=1562657403;
to_lang_often=%5B%7B%22value%22%3A%22en%22%2C%22text%22%3A%22%u82F1%u8BED%22%7D%2C%7B%22val
ue%22%3A%22zh%22%2C%22text%22%3A%22%u4E2D%u6587%22%7D%5D; REALTIME_TRANS_SWITCH=1;
FANYI_WORD_SWITCH=1; HISTORY_SWITCH=1; SOUND_SPD_SWITCH=1; SOUND_PREFER_SWITCH=1; delPer=0;
BDORZ=B490B5EBF6F3CD402E515D22BCDA1598; BCLID=6890774803653935935;
BDSFRCVID=4XAsJeCCxG3DLCbwbJrKDGwjNA0UN_I3KhXZ3J;
H_BDCLCKID_SF=tRk8oIDaJCvSe6r1MtQ_M4F_qxby26nUQ5neaJ5n0-
nnhnL4W46bqJKFLtozKMoI3C7fotJJ5nololIRy6CKjjb-jaDqJ5n3bTnjstcS2RREHJrg-
trSMDCShGRGWlO9WDTm_D_KfxnkOnc6qJj0-jjXqqo8K5Ljaa5n-
pPKKRAaqD04bPbZL4DdMa7HLtAO3mkjbnczfn02OP5P5lJ_e-4syPRG2xRnWIvrKfA-
b4ncjRcTehoM3xI8LNj405OTt2LEoDPMJKIbMI_rMbbfhKC3hqJfaI62aKDs_RCMBhcqEIL4eJOIb6_w5gcq0T_Httj
tXR0atn7ZSMbSj4Qo5pK95p38bxnDK2rQLb5zah5nhMJS3j7JDMP0-4rJhxby523i5J6vQpnJ8hQ3DRoWXPIqbN7P-
p5Z5mAqKl0MLIOkbC_6j5DWDTvLeU7J-n8XbI60XRj85-
ohHJrFMtQ_q4tehHRMBUo9WDTm_DoTttt5fUj6qJj855jXqqo8KMtHJaFf-pPKKRAashnzWjrkqqOQ5pj-
WnQr3mkjbn5yfn02OpjPX6joht4syPRG2xRnWIvrKfA-
b4ncjRcTehoM3xI8LNj405OTt2LEoC0XtIDhMDvPMCTSMt_HMxrKetJyaR0JhpjbWJ5TEPnjDUOdLPDW-
46HBM3xbKQw5CJGBf7zhpvdWhC5y6ISKx-_J68Dtf5; ZD_ENTRY=baidu; PSINO=2;
H_PS_PSSID=26525_1444_21095_29578_29521_28518_29098_29568_28830_29221_26350_29459;
locale=zh; Hm_lvt_64ecd82404c51e03dc91cb9e8c025574=1563426293,1563996067;
from_lang_often=%5B%7B%22value%22%3A%22zh%22%2C%22text%22%3A%22%u4E2D%u6587%22%7D%2C%7B%22v
alue%22%3A%22en%22%2C%22text%22%3A%22%u82F1%u8BED%22%7D%5D;
Hm_lpvt_64ecd82404c51e03dc91cb9e8c025574=1563999768;
yjs_js_security_passport=2706b5b03983b8fa12fe756b8e4a08b98fb43022_1563999769_js',
            'pragma': 'no-cache',
            'upgrade-insecure-requests': '1',
            'user-agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/75.0.3770.142 Safari/537.36',
        }

    # 获取token
    def get_token(self):
        token_url = 'https://fanyi.baidu.com/?aldtype=16047'
        # 定义请求头
        r = requests.get(self.token_url,headers=self.headers)
        token = re.findall(r"token: '(.*?)'",r.text)
        window_gtk = re.findall(r"window.*?gtk = '(.*?)';</script>",r.text)
        if token:
            return token[0],window_gtk[0]

    # 获取sign
    def get_sign(self,word):
        with open('百度翻译.js','r') as f:
            js_data = f.read()

        exec_object = execjs.compile(js_data)
        sign = exec_object.eval('e("{}")'.format(word))

        return sign

    # 主函数
    def main(self,word,fro,to):
        token,gtk = self.get_token()
        sign = self.get_sign(word)
        # 找到form表单数据如下,sign和token需要想办法获取
        form_data = {
            'from': fro,
```

```
            'to': to,
            'query': word,
            'transtype': 'realtime',
            'simple_means_flag': '3',
            'sign': sign,
            'token': token
        }
        r = requests.post(self.post_url,data=form_data,headers=self.headers)
        print(r.json()['trans_result']['data'][0]['dst'])

if __name__ == '__main__':
    spider = BaiduTranslateSpider()
    menu = '1. 英译汉 2. 汉译英'
    choice = input('1. 英译汉 2. 汉译英 : ')
    word = input('请输入要翻译的单词:')
    if choice == '1':
        fro = 'en'
        to = 'zh'
    elif choice == '2':
        fro = 'zh'
        to = 'en'

    spider.main(word,fro,to)
```

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**

```
# 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 re.findall(r'.*?中华人民共和国县以上行政区划代码.*?', title, re.S):
        # 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('127.0.0.1','root','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_two_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.xpath('./@title')[0]
            # 正则匹配title中包含这个字符串的链接
            if re.findall(r'.*?中华人民共和国县以上行政区划代码.*?',title,re.S):
                # 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' + a.xpath('./@href')[0]
                break

        # 从已提取的two_link中提取二级页面的真实链接
        two_html = requests.get(two_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]
        # 实现增量爬取
        self.cursor.execute('select * from version')
        result = self.cursor.fetchall()
        if result:
            if result[-1][0] == real_two_link:
                print('已是最新')
        else:
            self.get_data(real_two_link)
            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()')
            city_info['name'] = tr.xpath('./td[3]/text()')
            print(city_info)



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

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)
        for app in app_json['data']:
            # 应用名称
            name = app['displayName']
            # 应用链接
            link = 'http://app.mi.com/details?id={}'.format(app['packageName'])
            d = { '名称' : name,'链接' : link }
            print(d)

    # 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))
```

json解析模块

json.loads(json格式字符串)

- 作用

```
把json格式的字符串转为Python数据类型
```

- 示例

```
html_json = json.loads(res.text)
```

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

- 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
```

- 参数说明

```
第1个参数: python类型的数据(字典, 列表等)
第2个参数: 文件对象
第3个参数: ensure_ascii=False # 序列化时编码
```

- 示例

```python
import json

app_dict = {
    '应用名称' : 'QQ',
    '应用链接' : 'http://qq.com'
}
with open('小米.json','a') as f:
    json.dump(app_dict,f,ensure_ascii=False)
```

今日作业

```
1、有道翻译案例复写一遍
2、百度翻译案例复写一遍
3、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
4、把链家二手房案例改写为多线程
```