SPIDER-DAY03

1. lxml解析库

1.1 安装使用流程

```
【1】安装
   sudo pip3 install lxml

【2】使用流程
   2.1》导模块 :           from lxml import etree
   2.2》创建解析对象 :      parse_html = etree.HTML(html)
   2.3》解析对象调用xpath : r_list = parse_html.xpath('xpath表达式')
```

1.2 lxml+xpath使用

```
【1】基准xpath：匹配所有电影信息的节点对象列表
  //dl[@class="board-wrapper"]/dd
  [<element dd at xxx>,<element dd at xxx>,...]

【2】遍历对象列表，依次获取每个电影信息
  item = {}
  for dd in dd_list:
       item['name'] = dd.xpath('.//p[@class="name"]/a/text()').strip()
       item['star'] = dd.xpath('.//p[@class="star"]/text()').strip()
       item['time'] = dd.xpath('.//p[@class="releasetime"]/text()').strip()
```

2. 豆瓣图书爬虫

2.1 需求分析

```
【1】抓取目标 - 豆瓣图书top250的图书信息
    https://book.douban.com/top250?start=0
    https://book.douban.com/top250?start=25
    https://book.douban.com/top250?start=50
    ... ...

【2】抓取数据
    2.1) 书籍名称 : 红楼梦
    2.2) 书籍描述 : [清] 曹雪芹 著 / 人民文学出版社 / 1996-12 / 59.70元
    2.3) 书籍评分 : 9.6
    2.4) 评价人数 : 286382人评价
    2.5) 书籍类型 : 都云作者痴，谁解其中味?
```

2.2 实现流程

```
【1】确认数据来源 - 响应内容存在
【2】分析URL地址规律 - start为0 25 50 75 ...
【3】xpath表达式
    3.1) 基准xpath,匹配每本书籍的节点对象列表
         //div[@class="indent"]/table

    3.2) 依次遍历每本书籍的节点对象，提取具体书籍数据
        书籍名称 :  .//div[@class="pl2"]/a/@title
        书籍描述 :  .//p[@class="pl"]/text()
        书籍评分 :  .//span[@class="rating_nums"]/text()
        评价人数 :  .//span[@class="pl"]/text()
        书籍类型 :  .//span[@class="inq"]/text()
```

2.3 代码实现

```python
import requests
from lxml import etree
import time
import random
from fake_useragent import UserAgent

class DoubanBookSpider:
    def __init__(self):
        self.url = 'https://book.douban.com/top250?start={}'

    def get_html(self, url):
        headers = { 'User-Agent':UserAgent().random }
        html = requests.get(url=url, headers=headers).content.decode('utf-8','ignore')
        # 直接调用解析函数
        self.parse_html(html)

    def parse_html(self, html):
        p = etree.HTML(html)
```

```
        # 基准xpath,匹配每本书的节点对象列表
        table_list = p.xpath('//div[@class="indent"]/table')
        for table in table_list:
            item = {}
            # 书名
            name_list = table.xpath('.//div[@class="pl2"]/a/@title')
            item['book_name'] = name_list[0].strip() if name_list else None
            # 信息
            info_list = table.xpath('.//p[@class="pl"]/text()')
            item['book_info'] = info_list[0].strip() if info_list else None
            # 评分
            score_list = table.xpath('.//span[@class="rating_nums"]/text()')
            item['book_score'] = score_list[0].strip() if score_list else None
            # 人数
            number_list = table.xpath('.//span[@class="pl"]/text()')
            item['book_number'] = number_list[0].strip()[1:-1].strip() if number_list else None
            # 描述
            comment_list = table.xpath('.//span[@class="inq"]/text()')
            item['book_comment'] = comment_list[0].strip() if comment_list else None

            print(item)

    def run(self):
        for i in range(10):
            start = i * 25
            page_url = self.url.format(start)
            self.get_html(url=page_url)
            # 控制数据抓取的频率,uniform生成指定范围内浮点数
            time.sleep(random.uniform(0, 3))


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

3. 百度贴吧小视频爬虫

3.1 需求分析

```
【1】官网地址：进入某个百度贴吧，寻找有视频的帖子，比如如下帖子链接：
   https://tieba.baidu.com/p/7185877941

【2】目标
   2.1> 在此帖子中提取中具体视频的链接(src)
   2.2> 将视频抓取保存到本地文件(向src发请求获取bytes数据类型,以wb方式保存到本地)
```

3.2 实现流程

```
【1】确认数据来源 ： 静态！！！
【2】帖子中视频的xpath表达式

### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
### 重要：页面中xpath不能全信，一切以响应内容为主
```

3.3 代码实现

```python
import requests
from lxml import etree

# 向帖子链接发请求
url = 'https://tieba.baidu.com/p/7185877941'
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36'}
html = requests.get(url=url, headers=headers).text

# 视频链接的xpath表达式 - 一切以响应内容为准
x = '//div[@class="video_src_wrapper"]/embed/@data-video'
eobj = etree.HTML(html)
video_url_list = eobj.xpath(x)

# 将视频抓取保存到本地文件
if video_url_list:
    video_url = video_url_list[0]
    video_html = requests.get(url=video_url, headers=headers).content
    with open('girl.mp4', 'wb') as f:
        f.write(video_html)
else:
    print('提取视频链接失败')
```

4. 代理参数

4.1 代理IP概述

```
【1】定义
    代替你原来的IP地址去对接网络的IP地址

【2】作用
```

```
    隐藏自身真实IP,避免被封

【3】获取代理IP网站
    快代理、全网代理、代理精灵、... ...

【4】参数类型
    proxies
    proxies = { '协议':'协议://IP:端口号' }
    proxies = { '协议':'协议://用户名:密码@IP:端口号' }
```

4.2 代理分类

4.2.1 普通代理

```
【1】代理格式
    proxies = { '协议':'协议://IP:端口号' }

【2】使用免费普通代理IP访问测试网站: http://httpbin.org/get

import requests
url = 'http://httpbin.org/get'
headers = {'User-Agent':'Mozilla/5.0'}
# 定义代理,在代理IP网站中查找免费代理IP
proxies = {
    'http':'http://112.85.164.220:9999',
    'https':'https://112.85.164.220:9999'
}
html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
print(html)
```

4.2.2 私密代理和独享代理

```
【1】代理格式
    proxies = { '协议':'协议://用户名:密码@IP:端口号' }

【2】使用私密代理或独享代理IP访问测试网站: http://httpbin.org/get

import requests
url = 'http://httpbin.org/get'
proxies = {
    'http': 'http://309435365:szayclhp@106.75.71.140:16816',
    'https':'https://309435365:szayclhp@106.75.71.140:16816',
}
headers = {
    'User-Agent' : 'Mozilla/5.0',
}

html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
print(html)
```

```
18
```

4.3 建立代理IP池

```
1  """
2  建立开放代理的代理ip池
3  """
4  import requests
5
6  class ProxyPool:
7      def __init__(self):
8          self.api_url = 'http://dev.kdlapi.com/api/getproxy/?
   orderid=999955248138592&num=20&protocol=2&method=2&an_ha=1&sep=1'
9          self.test_url = 'http://httpbin.org/get'
10         self.headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36'}
11
12     def get_proxy(self):
13         html = requests.get(url=self.api_url, headers=self.headers).text
14         # proxy_list: ['1.1.1.1:8888','2.2.2.2:9999,...]
15         proxy_list = html.split('\r\n')
16         for proxy in proxy_list:
17             # 测试proxy是否可用
18             self.test_proxy(proxy)
19
20     def test_proxy(self, proxy):
21         """测试1个代理ip是否可用"""
22         proxies = {
23             'http' : 'http://{}'.format(proxy),
24             'https': 'https://{}'.format(proxy),
25         }
26         try:
27             resp = requests.get(url=self.test_url, proxies=proxies, headers=self.headers,
   timeout=3)
28             if resp.status_code == 200:
29                 print(proxy,'\033[31m可用\033[0m')
30             else:
31                 print(proxy,'不可用')
32         except Exception as e:
33             print(proxy, '不可用')
34
35     def run(self):
36         self.get_proxy()
37
38 if __name__ == '__main__':
39     spider = ProxyPool()
40     spider.run()
41
```

5. requests.post()

5.1 POST请求

```
【1】适用场景 ： Post类型请求的网站

【2】参数 ： data={}
  2.1) Form表单数据：字典
  2.2) res = requests.post(url=url,data=data,headers=headers)

【3】POST请求特点 ： Form表单提交数据

```

5.2 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
  2.1) Network：抓取网络数据包
    a> ALL：抓取所有的网络数据包
    b> XHR：抓取异步加载的网络数据包
    c> JS ： 抓取所有的JS文件
  2.2) 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
  2.3) 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
  3.1) 单击左侧网络数据包地址，进入数据包详情，查看右侧
  3.2) 右侧：
    a> Headers：整个请求信息
       General、Response Headers、Request Headers、Query String、Form Data
    b> Preview：对响应内容进行预览
    c> Response：响应内容

```

6. 今日作业

```
【1】链家二手房爬虫
   # 注意：一切以响应内容为准
   1.1> 官网地址：进入链家官网，点击二手房 ： https://bj.lianjia.com/ershoufang/
   1.2> 目标 ： 抓取100页的二手房源信息，包含房源的：
        名称
        地址
        户型、面积、方位、是否精装、楼层、年代、类型
        总价
        单价
```

```
    1.3> 数据处理
        将数据分别存入：MySQL、MongoDB、csv文件中

【2】抓取快代理网站免费高匿代理，并测试是否可用来建立自己的代理IP池
    https://www.kuaidaili.com/free/
```

SPIDER-DAY04

1. 有道翻译爬虫

1.1 项目需求

```
破解有道翻译接口，抓取翻译结果

# 结果展示
请输入要翻译的词语：elephant
翻译结果：大象
************************
请输入要翻译的词语：喵喵叫
翻译结果：mews
```

1.2 项目分析流程

```
【1】准备抓包：F12开启控制台，刷新页面
【2】寻找地址
    2.1) 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-General-Request URL
【3】发现规律
    3.1) 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
    3.2) 分析对比 Network - All(或者XHR) - Form Data，发现对应的规律
【4】寻找JS加密文件
    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{}
【5】查看JS代码
    搜索关键字，找到相关加密方法，用python实现加密算法
【6】断点调试
    JS代码中部分参数不清楚可通过断点调试来分析查看
【7】Python实现JS加密算法
```

1.3 项目步骤

1、开启F12抓包，找到Form表单数据如下：

```
i: 喵喵叫
from: AUTO
to: AUTO
smartresult: dict
client: fanyideskweb
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
doctype: json
version: 2.1
keyfrom: fanyi.web
action: FY_BY_REALTlME
```

2、在页面中多翻译几个单词，观察Form表单数据变化

```
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
# 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

```
控制台右上角 - Search - 搜索salt - 查看文件 - 格式化输出

【结果】 ：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

```
【ts】经过分析为13位的时间戳，字符串类型
  js代码实现)  "" + (new Date).getTime()
  python实现) str(int(time.time()*1000))

【salt】
  js代码实现)  ts + parseInt(10 * Math.random(), 10);
  python实现)  ts + str(random.randint(0, 9))

【sign】 ('设置断点调试，来查看 e 的值，发现 e 为要翻译的单词')
  js代码实现) n.md5("fanyideskweb" + e + salt + "]BjuETDhU)zqSxf-=B#7m")

   from hashlib import md5
   s = md5()
   s.update(''.encode())
   sign = s.hexdigest()
```

5、pycharm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
```

1.4 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 = {
      # 检查频率最高 - 3个
      "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):
    # ts
    ts = str(int(time.time()*1000))
    # salt
    salt = ts + 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()

    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={}
    # 检查了salt sign
    data = {
      "i": word,
      "from": "AUTO",
```

```
      "to": "AUTO",
      "smartresult": "dict",
      "client": "fanyideskweb",
      "salt": salt,
      "sign": sign,
      "ts": ts,
      "bv": "7e3150ecbdf9de52dc355751b074cf60",
      "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)
    html = requests.post(
      url=self.url,
      data=data,
      headers=self.headers
    ).json()
    # res.json() 将json格式的字符串转为python数据类型
    result = html['translateResult'][0][0]['tgt']

    print(result)

  # 主函数
  def run(self):
    # 输入翻译单词
    word = input('请输入要翻译的单词:')
    self.attack_yd(word)

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

2. 百度翻译JS逆向爬虫

2.1 JS逆向详解

```
【1】应用场景
    当JS加密的代码过于复杂,没有办法破解时,考虑使用JS逆向思想

【2】模块
    2.1》模块名: execjs
    2.2》安装:  sudo pip3 install pyexecjs
    2.3》使用流程
        import execjs
        with open('xxx.js', 'r') as f:
            js_code = f.read()

        js_obj = execjs.compile(js_code)
        js_obj.eval('函数名("参数")')
```

2.2 JS代码调试

- **抓到 JS 加密文件，存放到 translate.js 文件中**

```
// e(r, gtk)  增加了gtk参数
// i = window[l] 改为了 i = gtk
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,gtk) {
    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 = gtk || "") || "";
    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;
```

- **test_translate.py调试JS文件**

```
import execjs

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

jsobj = execjs.compile(jscode)
sign = jsobj.eval('e("hello","320305.131321201")')
print(sign)
```

2.3 百度翻译代码实现

```
import requests
import execjs
import re

class BaiduTranslateSpider:
    def __init__(self):
        self.url = 'https://fanyi.baidu.com/v2transapi?from=en&to=zh'
        self.index_url = 'https://fanyi.baidu.com/'
        self.post_headers = {
            "accept": "*/*",
            "accept-encoding": "gzip, deflate, br",
            "accept-language": "zh-CN,zh;q=0.9",
            "cache-control": "no-cache",
            "content-length": "135",
            "content-type": "application/x-www-form-urlencoded; charset=UTF-8",
```

```
            "cookie": "PSTM=1607343359; BIDUPSID=537631C02856FFE7766E81A428137630;
BAIDUID=BD4764B5157F4DA011C301C831041961:FG=1; REALTIME_TRANS_SWITCH=1; FANYI_WORD_SWITCH=1;
HISTORY_SWITCH=1; SOUND_SPD_SWITCH=1; SOUND_PREFER_SWITCH=1;
BAIDUID_BFESS=BD4764B5157F4DA011C301C831041961:FG=1;
H_PS_PSSID=33213_1455_33126_33060_33261_31254_33183_33181_32845_26350_33198_33238_33217_33216_
33215_33185; BA_HECTOR=80ag2ga5818g242h9s1ftf0kk0q; BDRCVFR[X_XKQks0S63]=mk3SLVN4HKm;
BDRCVFR[dG2JNJb_ajR]=mk3SLVN4HKm; BDRCVFR[-pGxjrCMryR]=mk3SLVN4HKm;
Hm_lvt_64ecd82404c51e03dc91cb9e8c025574=1607421950,1607960594;
Hm_lpvt_64ecd82404c51e03dc91cb9e8c025574=1607960601;
ab_sr=1.0.0_ZDBkODQ4YWExMTBkMWYzM2ZhODM1OGU0MDc4Yzg1NDlmNjM0N2U2MjdjMjEzY2RhMmYxZmNkNGY3OTMyZm
VjM2VjYzBlMjFiMjk1ZGExNDJhNmY4YmY4NThjZjZmZmM3;
__yjsv5_shitong=1.0_7_53041fb6476666e15c96dc5687b0b683b387_300_1607960602008_110.251.244.204_3
74e2e38; yjs_js_security_passport=0ba0cacde2b0d5bffc7c1c2fc7be1c1694369731_1607960602_js",
            "origin": "https://fanyi.baidu.com",
            "pragma": "no-cache",
            "referer": "https://fanyi.baidu.com/",
            "sec-fetch-dest": "empty",
            "sec-fetch-mode": "cors",
            "sec-fetch-site": "same-origin",
            "user-agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/86.0.4240.193 Safari/537.36",
            "x-requested-with": "XMLHttpRequest",
        }

    def get_gtk_token(self):
        """获取gtk和token"""
        get_headers = {
            "accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=
0.8,application/signed-exchange;v=b3;q=0.9",
            "accept-encoding": "gzip, deflate, br",
            "accept-language": "zh-CN,zh;q=0.9",
            "cache-control": "no-cache",
            "cookie": "PSTM=1607343359; BIDUPSID=537631C02856FFE7766E81A428137630;
BAIDUID=BD4764B5157F4DA011C301C831041961:FG=1; REALTIME_TRANS_SWITCH=1; FANYI_WORD_SWITCH=1;
HISTORY_SWITCH=1; SOUND_SPD_SWITCH=1; SOUND_PREFER_SWITCH=1;
BAIDUID_BFESS=BD4764B5157F4DA011C301C831041961:FG=1;
H_PS_PSSID=33213_1455_33126_33060_33261_31254_33183_33181_32845_26350_33198_33238_33217_33216_
33215_33185; BA_HECTOR=80ag2ga5818g242h9s1ftf0kk0q; BDRCVFR[X_XKQks0S63]=mk3SLVN4HKm;
BDRCVFR[dG2JNJb_ajR]=mk3SLVN4HKm; BDRCVFR[-pGxjrCMryR]=mk3SLVN4HKm;
Hm_lvt_64ecd82404c51e03dc91cb9e8c025574=1607421950,1607960594;
Hm_lpvt_64ecd82404c51e03dc91cb9e8c025574=1607960594;
ab_sr=1.0.0_ODI3MThkMDlhZjkwNWZiZThhZTg3Njc2ZWRkNjRhY2MwNjdhYTVhMDY3MjliZGY3NWJjYjkxNzZlZjU1Yj
M5NTRiM2YyMmVhMDNiZTdiZDU2NmNiODZiNWJiMmRjYzRk;
__yjsv5_shitong=1.0_7_53041fb6476666e15c96dc5687b0b683b387_300_1607960594747_110.251.244.204_b
b4b61ab; yjs_js_security_passport=fa4d2e13a89ef434f713f2cb621120928516a173_1607960595_js",
            "pragma": "no-cache",
            "sec-fetch-dest": "document",
            "sec-fetch-mode": "navigate",
            "sec-fetch-site": "none",
            "sec-fetch-user": "?1",
            "upgrade-insecure-requests": "1",
            "user-agent": "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/86.0.4240.193 Safari/537.36",
        }
        html = requests.get(url=self.index_url,
                            headers=get_headers).text
```

```
        gtk = re.findall("window.gtk = '(.*?)'", html, re.S)[0]
        token = re.findall("token: '(.*?)'", html, re.S)[0]

        return gtk, token

    def get_sign(self, word):
        """功能函数:生成sign"""
        # 先获取到gtk和token
        gtk, token = self.get_gtk_token()
        with open('translate.js', 'r', encoding='utf-8') as f:
            js_code = f.read()

        js_obj = execjs.compile(js_code)
        sign = js_obj.eval('e("{}","{}")'.format(word, gtk))

        return sign

    def attack_bd(self, word):
        """爬虫逻辑函数"""
        gtk, token = self.get_gtk_token()
        sign = self.get_sign(word)
        data = {
            "from": "en",
            "to": "zh",
            "query": word,
            "transtype": "realtime",
            "simple_means_flag": "3",
            "sign": sign,
            "token": token,
            "domain": "common",
        }
        # json():把json格式的字符串转为python数据类型
        html = requests.post(url=self.url,
                             data=data,
                             headers=self.post_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.attack_bd(word))

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

3. 动态加载数据抓取

3.1 AJAX动态加载

- **数据特点**

```
【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
```

- **分析流程**

```
【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)
```

3.2 豆瓣电影爬虫

3.2.1 项目需求

```
【1】地址：豆瓣电影 - 排行榜 - 剧情
【2】目标：电影名称、电影评分

<span><a href=".*?type_name=(.*?)&type=(.*?)&
https://movie.douban.com/chart
```

3.2.2 抓包分析

```
【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
```

3.2.3 代码实现

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

class DoubanSpider:
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?type=
{}&interval_id=100%3A90&action=&start={}&limit=20'

```

```
    def get_html(self, url):
        """功能函数1: 获取html"""
        headers = {'User-Agent':UserAgent().random}
        html = requests.get(url=url, headers=headers).text

        return html

    def parse_html(self, url):
        # 提取数据函数
        # html: [{},{},{},....]
        html = json.loads(self.get_html(url=url))
        for one_film_dict in html:
            item = {}
            item['rank'] = one_film_dict['rank']
            item['name'] = one_film_dict['title']
            item['time'] = one_film_dict['release_date']
            item['score'] = one_film_dict['score']

            print(item)

    def get_total(self, typ):
        """获取电影总数"""
        total_url = 'https://movie.douban.com/j/chart/top_list_count?type={}&interval_id=100%3A90'.format(typ)
        total_html = json.loads(self.get_html(url=total_url))

        return total_html['total']

    def get_all_film_dict(self):
        """获取所有电影类别及对应的type值的字典"""
        all_type_url = 'https://movie.douban.com/chart'
        all_type_html = self.get_html(url=all_type_url)
        regex = '<span><a href=.*?type_name=(.*?)&type=(.*?)&interval_id=100:90&action=">'
        pattern = re.compile(regex, re.S)
        # all_list: [('剧情','11'),('喜剧','5'),...]
        all_list = pattern.findall(all_type_html)
        all_film_dict = {}
        for one in all_list:
            all_film_dict[one[0]] = one[1]

        return all_film_dict

    def run(self):
        # {'剧情':'5', '喜剧':'23', '爱情':'13',... ...}
        all_film_dict = self.get_all_film_dict()
        # 生成提示菜单
        menu = ''
        for key in all_film_dict:
            menu = menu + key + '|'
        print(menu)
        # 接收用户输入,并获取对应的type的值
        film_type = input('请输入电影类别:')
        typ = all_film_dict[film_type]
        # 获取此类别下的电影总数
        total = self.get_total(typ)
        for start in range(0, total, 20):
            page_url = self.url.format(typ, start)
```

```
            self.parse_html(url=page_url)
            # 控制频率
            time.sleep(random.randint(1, 2))


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

4. 今日作业

```
【1】肯德基餐厅门店信息抓取（POST请求练习）
    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：保存到数据库
    1.4) 程序运行效果：
         请输入城市名称：北京
         把北京的所有肯德基门店的信息保存到数据库中
```