Day02回顾

数据抓取

- **思路步骤**

```
【1】先确定是否为动态加载网站
【2】找URL规律
【3】正则表达式 | xpath表达式
【4】定义程序框架，补全并测试代码
```

- **多级页面数据抓取思路**

```
【1】整体思路
    1.1> 爬取一级页面,提取 所需数据+链接,继续跟进
    1.2> 爬取二级页面,提取 所需数据+链接,继续跟进
    1.3> ... ...

【2】代码实现思路
    2.1> 避免重复代码 - 请求、解析需定义函数
```

- **增量爬虫实现思路**

```
【1】原理
```

利用Redis集合特性，可将抓取过的指纹添加到redis集合中，根据返回值来判定是否需要抓取

返回值为1 ： 代表之前未抓取过，需要进行抓取

返回值为0 ： 代表已经抓取过，无须再次抓取

【2】代码实现模板

```
import redis
from hashlib import md5
import sys

class XxxIncrSpider:
  def __init__(self):
    self.r =
redis.Redis(host='localhost',port=6379,
db=0)

  def url_md5(self,url):
    """对URL进行md5加密函数"""
    s = md5()
    s.update(url.encode())
    return s.hexdigest()

  def run_spider(self):
    href_list =
['url1','url2','url3','url4']
    for href in href_list:
      href_md5 = self.url_md5(href)
```

```
        if
self.r.sadd('spider:urls',href_md5) ==
1:
            返回值为1表示添加成功，即之前未抓取
过，则开始抓取
        else:
            sys.exit()
```

- **目前反爬处理**

```
【1】基于User-Agent反爬
    1.1) 发送请求携带请求头: headers=
{'User-Agent' : 'Mozilla/5.0 xxxxxx'}
    1.2) 多个请求时随机切换User-Agent
        a) 定义py文件存放大量User-Agent，导
入后使用random.choice()每次随机选择
        b) 使用fake_useragent模块每次访问
随机生成User-Agent
            from fake_useragent import
UserAgent
            agent = UserAgent().random

【2】响应内容存在特殊字符
    解码时使用ignore参数
    html = requests.get(url=url,
headers=headers).content.decode('',
'ignore')
```

数据持久化

- csv

```
 import csv
 with open('xxx.csv','w',encoding='utf-8',newline='') as f:
    writer = csv.writer(f)
    writer.writerow([])
```

- MySQL

```
import pymysql

# __init__(self):
    self.db = pymysql.connect('IP',.....)
    self.cursor = self.db.cursor()

# save_html(self,r_list):
    self.cursor.execute('sql',[data1])
    self.db.commit()

# run(self):
    self.cursor.close()
    self.db.close()
```

- MongoDB

```

```

```
import pymongo


# __init__(self):
    self.conn =
pymongo.MongoClient('IP',27017)
    self.db = self.conn['cardb']
    self.myset = self.db['car_set']

# save_html(self,r_list):
    self.myset.insert_one(dict)

# MongoDB - Commmand - 库->集合->文档
mongo
>show dbs
>use db_name
>show collections
>db.集合名.find().pretty()
>db.集合名.count()
>db.集合名.drop()
>db.dropDatabase()
```

Day03笔记

==xpath解析==

- **定义**

XPath即为XML路径语言，它是一种用来确定XML文档中某部分位置的语言，同样适用于HTML文档的检索

- **匹配演示 - 猫眼电影top100**

```
【1】查找所有的dd节点
    //dd
【2】获取所有电影的名称的a节点： 所有class属性值为name的a节点
    //p[@class="name"]/a
【3】获取dl节点下第2个dd节点的电影节点
    //dl[@class="board-wrapper"]/dd[2]

【4】获取所有电影详情页链接： 获取每个电影的a节点的href的属性值
    //p[@class="name"]/a/@href

【注意】
    1> 只要涉及到条件,加 [] ：
//dl[@class="xxx"]    //dl/dd[2]
    2> 只要获取属性值,加 @  ：
//dl[@class="xxx"]    //p/a/@href
```

- **选取节点**

```
【1】// ： 从所有节点中查找（包括子节点和后代节点）
【2】@  ： 获取属性值
  2.1> 使用场景1（属性值作为条件）
       //div[@class="movie-item-info"]
```

```
  2.2> 使用场景2（直接获取属性值）
       //div[@class="movie-item-
info"]/a/img/@src

【3】练习 - 猫眼电影top100
  3.1> 匹配电影名称
      //p[@class="name"]/a/@title
  3.2> 匹配电影主演
      //div[@class="movie-item-
info"]/p[2]/text()
  3.3> 匹配上映时间
      //div[@class="movie-item-
info"]/p[3]/text()
  3.4> 匹配电影链接
      //p[@class="name"]/a/@href
```

- **匹配多路径（或）**

```
xpath表达式1 | xpath表达式2 | xpath表达式3
```

- **常用函数**

【1】text() ：获取节点的文本内容

xpath表达式末尾不加 /text() :则得到的结果为节点对象

xpath表达式末尾加 /text() 或者 /@href : 则得到结果为字符串

【2】contains() ： 匹配属性值中包含某些字符串节点

匹配class属性值中包含 'movie-item' 这个字符串的 div 节点

```
//div[contains(@class,"movie-item")]
```

- **终极总结**

【1】字符串： xpath表达式的末尾为： /text()、/@href 得到的列表中为'字符串'

【2】节点对象： 其他剩余所有情况得到的列表中均为'节点对象'

[<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]

[<element div at xxxa>,<element div at xxxb>]

[<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]

- **课堂练习**

```
【1】匹配汽车之家-二手车,所有汽车的链接 :

 //ul[@class="viewlist_ul"]/li/a/@href
【2】匹配汽车之家-汽车详情页中,汽车的
     2.1)名称: //div[@class="car-box"]/h3/text()
     2.2)里程: //ul[@class="brand-unit-item fn-clear"]/li[1]/h4/text()
     2.3)时间: //ul[@class="brand-unit-item fn-clear"]/li[2]/h4/text()
     2.4)挡位+排量: //ul[@class="brand-unit-item fn-clear"]/li[3]/h4/text()
     2.5)所在地: //ul[@class="brand-unit-item fn-clear"]/li[4]/h4/text()
     2.6)价格: //div[@class="brand-price-item"]/span/text()
```

==lxml解析库==

- **安装**

```
【1】Ubuntu:  sudo pip3 install lxml
【2】Windows: python -m pip install lxml
```

- **使用流程**

```
1、导模块
    from lxml import etree
2、创建解析对象
    parse_html = etree.HTML(html)
3、解析对象调用xpath
    r_list = parse_html.xpath('xpath表达式')
    r_list情况1: [<element dd at xxx>, <element xx>]
    r_list情况2: ['','','','']
```

- **xpath最常用**

```
【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()
```

豆瓣图书信息抓取 - xpath

- **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？
```

- **步骤分析**

```
【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()
```

- **代码实现**

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

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

    def get_html(self, url):
        """使用随机的User-Agent"""
        headers = {'User-
Agent':UserAgent().random}
        html = requests.get(url=url,
headers=headers).text
        self.parse_html(html)

    def parse_html(self, html):
        """lxml+xpath进行数据解析"""
        parse_obj = etree.HTML(html)
        # 1.基准xpath：提取每本书的节点对象
列表
        table_list =
parse_obj.xpath('//div[@class="indent"]
/table')
        for table in table_list:
            item = {}
            # 书名
            name_list =
table.xpath('.//div[@class="pl2"]/a/@ti
tle')
```

```
                item['name'] =
name_list[0].strip() if name_list else
None
                # 描述
                content_list =
table.xpath('.//p[@class="pl"]/text()')
                item['content'] =
content_list[0].strip() if content_list
else None
                # 评分
                score_list =
table.xpath('.//span[@class="rating_num
s"]/text()')
                item['score'] =
score_list[0].strip() if score_list
else None
                # 评价人数
                nums_list =
table.xpath('.//span[@class="pl"]/text(
)')
                item['nums'] = nums_list[0]
[1:-1].strip() if nums_list else None
                # 类别
                type_list =
table.xpath('.//span[@class="inq"]/text
()')
                item['type'] =
type_list[0].strip() if type_list else
None
```

```
            print(item)

    def run(self):
        for i in range(5):
            start = (i - 1) * 25
            page_url =
self.url.format(start)
            self.get_html(page_url)

 time.sleep(random.randint(1,2))

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

链家二手房案例（xpath）

- **确定是否为静态**

```
打开二手房页面 -> 查看网页源码 -> 搜索关键字
```

- **xpath表达式**

```
【1】基准xpath表达式(匹配每个房源信息节点列表)
    '此处滚动鼠标滑轮时,li节点的class属性值会发生变化,通过查看网页源码确定xpath表达式'
```

```
 //ul[@class="sellListContent"]/li[@cla
ss="clear LOGVIEWDATA LOGCLICKDATA"]
```

【2】依次遍历后每个房源信息xpath表达式

2.1)名称:

```
.//div[@class="positionInfo"]/a[1]/text
()
```

2.2)地址:

```
.//div[@class="positionInfo"]/a[2]/text
()
```

2.3)户型+面积+方位+是否精装+楼层+年代+类型

info_list:

```
'.//div[@class="houseInfo"]/text()' ->
[0].strip().split('|')
```

a)户型: info_list[0]

b)面积: info_list[1]

c)方位: info_list[2]

d)精装: info_list[3]

e)楼层: info_list[4]

f)年代: info_list[5]

g)类型: info_list[6]

2.4)总价+单价

a)总价:

```
.//div[@class="totalPrice"]/span/text()
```

b)单价:

```
.//div[@class="unitPrice"]/span/text()
```

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

- **示意代码**

```
import requests
from lxml import etree
from fake_useragent import UserAgent
```

```
# 1.定义变量
url =
'https://bj.lianjia.com/ershoufang/pg1/
'
headers = {'User-
Agent':UserAgent().random}
# 2.获取响应内容
html =
requests.get(url=url,headers=headers).t
ext
# 3.解析提取数据
parse_obj = etree.HTML(html)
# 3.1 基准xpath,得到每个房源信息的li节点对象
列表，如果此处匹配出来空，则一定要查看响应内容
li_list =
parse_obj.xpath('//ul[@class="sellListC
ontent"]/li[@class="clear LOGVIEWDATA
LOGCLICKDATA"]')
for li in li_list:
    item = {}
    # 名称
    name_list =
li.xpath('.//div[@class="positionInfo"]
/a[1]/text()')
    item['name'] = name_list[0].strip()
if name_list else None
    # 地址
```

```
        add_list =
li.xpath('.//div[@class="positionInfo"]
/a[2]/text()')
        item['add'] = add_list[0].strip()
if add_list else None
        # 户型 + 面积 + 方位 + 是否精装 + 楼层
+ 年代 + 类型
        house_info_list =
li.xpath('.//div[@class="houseInfo"]/te
xt()')
        item['content'] =
house_info_list[0].strip() if
house_info_list else None
        # 总价
        total_list =
li.xpath('.//div[@class="totalPrice"]/s
pan/text()')
        item['total'] =
total_list[0].strip() if total_list
else None
        # 单价
        unit_list =
li.xpath('.//div[@class="unitPrice"]/sp
an/text()')
        item['unit'] = unit_list[0].strip()
if unit_list else None

        print(item)
```

- **完整代码实现 - 自己实现**

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

class LianjiaSpider(object):
    def __init__(self):
        self.url =
'https://bj.lianjia.com/ershoufang/pg{}
/'

    def parse_html(self,url):
        headers = {'User-
Agent':UserAgent().random}
        html =
requests.get(url=url,headers=headers).c
ontent.decode('utf-8','ignore')
        self.get_data(html)


    def get_data(self,html):
        p = etree.HTML(html)
        # 基准xpath: [<element li at
xxx>,<element li>]
```

```
        li_list =
p.xpath('//ul[@class="sellListContent"]
/li[@class="clear LOGVIEWDATA
LOGCLICKDATA"]')
        # for遍历,依次提取每个房源信息,放到
字典item中
        item = {}
        for li in li_list:
            # 名称+区域
            name_list =
li.xpath('.//div[@class="positionInfo"]
/a[1]/text()')
            item['name'] =
name_list[0].strip() if name_list else
None
            address_list =
li.xpath('.//div[@class="positionInfo"]
/a[2]/text()')
            item['address'] =
address_list[0].strip() if address_list
else None
            # 户型+面积+方位+是否精装+楼层
+年代+类型
            # h_list: ['']
            h_list =
li.xpath('.//div[@class="houseInfo"]/te
xt()')
            if h_list:
```

```
                info_list =
h_list[0].split('|')
                if len(info_list) == 7:
                    item['model'] =
info_list[0].strip()
                    item['area'] =
info_list[1].strip()
                    item['direct'] =
info_list[2].strip()
                    item['perfect'] =
info_list[3].strip()
                    item['floor'] =
info_list[4].strip()
                    item['year'] =
info_list[5].strip()[:-2]
                    item['type'] =
info_list[6].strip()
                else:
                    item['model'] =
item['area'] = item['direct'] =
item['perfect'] = item['floor'] =
item['year'] = item['type'] = None
            else:
                item['model'] =
item['area'] = item['direct'] =
item['perfect'] = item['floor'] =
item['year'] = item['type'] = None

            # 总价+单价
```

```
            total_list =
li.xpath('.//div[@class="totalPrice"]/s
pan/text()')
            item['total'] =
total_list[0].strip() if total_list
else None
            unit_list =
li.xpath('.//div[@class="unitPrice"]/sp
an/text()')
            item['unit'] =
unit_list[0].strip() if unit_list else
None

            print(item)

    def run(self):
        for pg in range(1,101):
            url = self.url.format(pg)
            self.parse_html(url)

 time.sleep(random.randint(1,2))

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

- **持久化到数据库中 - 自己实现**

【1】将数据存入MongoDB数据库
【2】将数据存入MySQL数据库

代理参数-proxies

- **定义及分类**

【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

- **普通代理**

【1】获取代理IP网站
    快代理、全网代理、代理精灵、... ...

【2】参数类型

```
    proxies = { '协议':'协议://IP:端口号'
}
    proxies = {
          'http':'http://IP:端口号',
          'https':'https://IP:端口号',
    }
```

- **普通代理 - 示例**

```
# 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,header
s=headers,timeout=5).text
print(html)
```

- **私密代理+独享代理**

```
【1】语法结构
   proxies = { '协议':'协议://用户名:密码@IP:端口号' }

【2】示例
   proxies = {
      'http':'http://用户名:密码@IP:端口号',
      'https':'https://用户名:密码@IP:端口号',
   }
```

- **私密代理+独享代理 - 示例代码**

```
import requests
url = 'http://httpbin.org/get'
proxies = {
    'http':
'http://309435365:szayclhp@106.75.71.14
0:16816',

 'https':'https://309435365:szayclhp@10
6.75.71.140:16816',
}
headers = {
    'User-Agent' : 'Mozilla/5.0',
}

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

- **建立自己的代理IP池 - 开放代理 | 私密代理**

```
"""
收费代理:
    建立开放代理的代理IP池
思路:
    1、获取到开放代理
    2、依次对每个代理IP进行测试,能用的保存到文
件中
"""
```

```
import requests

class ProxyPool:
    def __init__(self):
        self.url = '代理网站的API链接'
        self.headers = {'User-
Agent':'Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/81.0.4044.122
Safari/537.36'}
        # 打开文件,用来存放可用的代理IP
        self.f = open('proxy.txt', 'w')

    def get_html(self):
        html =
requests.get(url=self.url,
headers=self.headers).text
        proxy_list = html.split('\r\n')
        for proxy in proxy_list:
            # 依次测试每个代理IP是否可用
            if self.check_proxy(proxy):
                self.f.write(proxy +
'\n')

    def check_proxy(self, proxy):
        """测试1个代理IP是否可用,可用返回
True,否则返回False"""
        test_url =
'http://httpbin.org/get'
```

```
            proxies = {
                'http' :
'http://{}'.format(proxy),
                'https':
'https://{}'.format(proxy)
            }
            try:
                res =
requests.get(url=test_url,
proxies=proxies, headers=self.headers,
timeout=2)
                if res.status_code == 200:
                    print(proxy,'\033[31m可
用\033[0m')
                    return True
                else:
                    print(proxy,'无效')
                    return False
            except:
                print(proxy,'无效')
                return False

    def run(self):
        self.get_html()
        # 关闭文件
        self.f.close()

if __name__ == '__main__':
    spider = ProxyPool()
```

```
    spider.run()
```

requests.post()

- **适用场景**

```
【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
```

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

有道翻译破解案例(post)

- **目标**

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
```

- **实现步骤**

【1】浏览器F12开启网络抓包,Network-All,页面翻译单词后找Form表单数据
【2】在页面中多翻译几个单词，观察Form表单数据变化（有数据是加密字符串）
【3】刷新有道翻译页面，抓取并分析JS代码（本地JS加密）
【4】找到JS加密算法，用Python按同样方式加密生成加密数据
【5】将Form表单数据处理为字典，通过requests.post()的data参数发送

- **具体实现**

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

```
【方法1】 : Network - JS选项 - 搜索关键词salt
【方法2】 : 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 + "n%A-rKaT5fb[Gy?;N5@Tj")
    python实现)
    from hashlib import md5
    string = "fanyideskweb" + e + salt + "n%A-rKaT5fb[Gy?;N5@Tj"
    s = md5()
    s.update(string.encode())
    sign = s.hexdigest()
```

4、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
```

5、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=570559528.1224
236;
_ntes_nnid=96bc13a2f5ce64962adfd6a2784672
14,1551873108952;
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()
```

今日作业

【1】将猫眼电影使用 lxml + xpath 实现
【2】将汽车之家案例使用 lxml + xpath 实现
【3】完善链家二手房案例，使用 lxml + xpath
【4】将 周测题 - 电影天堂案例 使用 lxml + xpath 实现
【5】将 周测题 - 4567TV案例 使用 lxml + xpath 实现
【6】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/

【7】仔细熟悉有道翻译案例抓包及流程分析

百度贴吧图片和视频抓取

【1】最终要求

    请输入贴吧名：赵丽颖吧
    请输入起始页：1
    请输入终止页：2
    最终：把赵丽颖吧前2页的所有帖子中的图片和视频保存到本地文件

【2】思路提示

2.1) 一级页面提取内容：提取1页中所有帖子的链接(50个)

2.2) 二级页面提取内容：提取所有图片的链接(0个、1个、多个)

2.3) 依次向每个图片链接发请求,把图片保存到本地文件

【3】知识点提示

3.1) 文件名命名问题（可以借助图片的URL地址）

```
image_url = 'http://...tiae736d12e2e9577.jpg'
filename = image_url[-12:]
```

3.2) 使用IE的User-Agent

```
headers = {'User-Agent':'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)'}
```

【4】注意

一切要以响应为主（正则表达式 或者 xpath表达式）