Day03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
```

- **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
【2】实现(根据sadd的返回值)
    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
```

- **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
```

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**

```python
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**

```python
import pymongo

# __init__(self):
    self.conn = pymongo.MongoClient('localhost', 27017)
    self.db = self.conn['库名']
    self.myset = self.db['集合名']

# save_html(self,r_list):
    self.myset.insert_one({})
```

Day04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
     //div[@class="movie-item-info"]/p[1]/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
     //div[@class="movie-item-info"]/p[1]/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 ：
   //li[@class="cards-li list-photo-li"]/a[1]/@href
   //a[@class="carinfo"]/@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/li[1]/h4/text()
    2.3)时间：  //ul/li[2]/h4/text()
    2.4)挡位+排量：//ul/li[3]/h4/text()
    2.5)所在地：//ul/li[4]/h4/text()
    2.6)价格：   //div[@class="brand-price-item"]/span[@class="price"]/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')
```

- **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**

```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()
```

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[@class="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
```

- **示意代码**

```python
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).text
# 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="sellListContent"]/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"]/text()')
    item['content'] = house_info_list[0].strip() if house_info_list else None
    # 总价
    total_list = li.xpath('.//div[@class="totalPrice"]/span/text()')
    item['total'] = total_list[0].strip() if total_list else None
    # 单价
    unit_list = li.xpath('.//div[@class="unitPrice"]/span/text()')
    item['unit'] = unit_list[0].strip() if unit_list else None
```

```
31
32      print(item)
```

- **完整代码实现 - 自己实现**

```
1  import requests
2  from lxml import etree
3  import time
4  import random
5  from fake_useragent import UserAgent
6
7  class LianjiaSpider(object):
8      def __init__(self):
9          self.url = 'https://bj.lianjia.com/ershoufang/pg{}/'
10
11     def parse_html(self,url):
12         headers = {'User-Agent':UserAgent().random}
13         html = requests.get(url=url,headers=headers).content.decode('utf-8','ignore')
14         self.get_data(html)
15
16
17     def get_data(self,html):
18         p = etree.HTML(html)
19         # 基准xpath: [<element li at xxx>,<element li>]
20         li_list = p.xpath('//ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA
   LOGCLICKDATA"]')
21         # for遍历,依次提取每个房源信息,放到字典item中
22         item = {}
23         for li in li_list:
24             # 名称+区域
25             name_list = li.xpath('.//div[@class="positionInfo"]/a[1]/text()')
26             item['name'] = name_list[0].strip() if name_list else None
27             address_list = li.xpath('.//div[@class="positionInfo"]/a[2]/text()')
28             item['address'] = address_list[0].strip() if address_list else None
29             # 户型+面积+方位+是否精装+楼层+年代+类型
30             # h_list: ['']
31             h_list = li.xpath('.//div[@class="houseInfo"]/text()')
32             if h_list:
33                 info_list = h_list[0].split('|')
34                 if len(info_list) == 7:
35                     item['model'] = info_list[0].strip()
36                     item['area'] = info_list[1].strip()
37                     item['direct'] = info_list[2].strip()
38                     item['perfect'] = info_list[3].strip()
39                     item['floor'] = info_list[4].strip()
40                     item['year'] = info_list[5].strip()[:-2]
41                     item['type'] = info_list[6].strip()
42                 else:
43                     item['model'] = item['area'] = item['direct'] = item['perfect'] =
   item['floor'] = item['year'] = item['type'] = None
44             else:
45                 item['model'] = item['area'] = item['direct'] = item['perfect'] =
   item['floor'] = item['year'] = item['type'] = None
46
47             # 总价+单价
48             total_list = li.xpath('.//div[@class="totalPrice"]/span/text()')
```

```
            item['total'] = total_list[0].strip() if total_list else None
            unit_list = li.xpath('.//div[@class="unitPrice"]/span/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**

```python
# 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)
```

- **代理IP池建立**

```python
"""
建立开放代理的代理ip池
"""
import requests

class ProxyPool:
    def __init__(self):
        self.api_url = 'http://dev.kdlapi.com/api/getproxy/?orderid=999955248138592&num=20&protocol=2&method=2&an_ha=1&sep=1'
        self.test_url = 'http://httpbin.org/get'
        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'}

    def get_proxy(self):
        html = requests.get(url=self.api_url, headers=self.headers).text
        # proxy_list: ['1.1.1.1:8888','2.2.2.2:9999,...]
        proxy_list = html.split('\r\n')
        for proxy in proxy_list:
            # 测试proxy是否可用
            self.test_proxy(proxy)

    def test_proxy(self, proxy):
        """测试1个代理ip是否可用"""
        proxies = {
            'http' : 'http://{}'.format(proxy),
            'https': 'https://{}'.format(proxy),
        }
        try:
            resp = requests.get(url=self.test_url, proxies=proxies, headers=self.headers, timeout=3)
            if resp.status_code == 200:
                print(proxy,'\033[31m可用\033[0m')
            else:
                print(proxy,'不可用')
        except Exception as e:
            print(proxy, '不可用')

    def run(self):
        self.get_proxy()
```

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

- **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@106.75.71.140:16816',
    '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)
```

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】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
```

今日作业

```
【1】将汽车之家案例使用 lxml + xpath 实现
【2】完善链家二手房案例，使用 lxml + xpath
【3】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/
【4】仔细熟悉有道翻译案例抓包及流程分析（至少操作5遍）
```