SPIDER-DAY03

1. lxml解析库

1.1 安装适用流程

```
【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. 链家二手房爬虫

3.1 需求分析

```
【1】官网地址：进入链家官网，点击二手房
    https://bj.lianjia.com/ershoufang/

【2】目标
    抓取100页的二手房源信息，包含房源的：
    2.1》名称
    2.2》地址
    2.3》户型、面积、方位、是否精装、楼层、年代、类型
    2.4》总价
    2.5》单价
```

3.2 实现流程

```
【1】确认数据来源 ： 静态！！！

【2】观察URL地址规律
    第1页：https://bj.lianjia.com/ershoufang/pg1/
    第2页：https://bj.lianjia.com/ershoufang/pg2/
    第n页：https://bj.lianjia.com/ershoufang/pgn/

【3】xpath表达式
    3.1》基准xpath（匹配每个房源的的li节点对象列表）
        '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/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]'

    3.2》每个房源具体信息的xpath表达式
        A) 名称：'.//div[@class="positionInfo"]/a[1]/text()'
        B) 地址：'.//div[@class="positionInfo"]/a[2]/text()'
        C) 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('|')
        D) 总价：'.//div[@class="totalPrice"]/span/text()'
        E) 单价：'.//div[@class="totalPrice"]/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
```

3.3 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/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

    print(item)
```

3.4 完整代码

```
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).content.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"]/text()')
```

```
            try:
                info_list = h_list[0].split('|')
                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()
            except Exception as e:
                print('get data error', e)
                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"]/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()
```

3.5 练习

```
【1】将链家二手房数据存入MongoDB数据库
【2】将链家二手房数据存入MySQL数据库
【3】将链家二手房数据存入本地csv文件
```

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',
}
```

```
15
16 html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
17 print(html)
```

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()
```

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
```

5.3 有道翻译爬虫

- **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
```

6. 今日作业

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