

Day02回顾

爬取网站思路

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

数据持久化 - csv

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

数据持久化 - MySQL

```
1 import pymysql
2
3 # __init__(self):
4     self.db = pymysql.connect('IP',... ...)
5     self.cursor = self.db.cursor()
6
7 # save_html(self,r_list):
8     self.cursor.execute('sql',[data1])
9     self.cursor.executemany('sql',[(data1),(data2),(data3)])
10    self.db.commit()
11
12 # run(self):
13     self.cursor.close()
14     self.db.close()
```

数据持久化 - MongoDB

```

1  import pymongo
2
3
4  # __init__(self):
5      self.conn = pymongo.MongoClient('IP',27017)
6      self.db = self.conn['cardb']
7      self.myset = self.db['car_set']
8
9  # save_html(self,r_list):
10     self.myset.insert_one(dict)
11     self.myset.insert_many([{}},{},{})
12
13 # MongoDB - Command - 库->集合->文档
14 mongo
15 >show dbs
16 >use db_name
17 >show collections
18 >db.集合名.find().pretty()
19 >db.集合名.count()
20 >db.集合名.drop()
21 >db.dropDatabase()

```

多级页面数据抓取

```

1  【1】整体思路
2      1.1> 爬取一级页面,提取 所需数据+链接,继续跟进
3      1.2> 爬取二级页面,提取 所需数据+链接,继续跟进
4      1.3> ... ...
5
6  【2】代码实现思路
7      2.1> 避免重复代码 - 请求、解析需定义函数

```

增量爬虫

■ MySQL实现增量

```

1  【1】数据库中创建指纹表,表中存放所有抓取过的'对URL地址进行md5加密'后的指纹
2  【2】代码实现流程模板
3  import pymysql
4  from hashlib import md5
5  import sys
6
7  class XxxIncrSpider:
8      def __init__(self):
9          self.db = pymysql.connect('localhost','root','123456','xxxdb',charset='utf8')
10         self.cursor = self.db.cursor()
11
12         def url_md5(self,url):
13             """对URL进行md5加密函数"""
14             s = md5()

```

```

15         s.update(url.encode())
16         return s.hexdigest()
17
18     def run_spider(self):
19         href_list = ['url1', 'url2', 'url3', 'url4']
20         for href in href_list:
21             href_md5 = self.url_md5(href)
22             if href_md5 不在指纹表中:
23                 开始进行数据抓取, 完成后将指纹插入到指纹表中
24             else:
25                 sys.exit()

```

■ Redis实现增量

```

1  【1】原理
2      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
3
4  【2】代码实现模板
5  import redis
6  from hashlib import md5
7  import sys
8
9  class XxxIncrSpider:
10     def __init__(self):
11         self.r = redis.Redis(host='localhost', port=6379, db=0)
12
13     def url_md5(self, url):
14         """对URL进行md5加密函数"""
15         s = md5()
16         s.update(url.encode())
17         return s.hexdigest()
18
19     def run_spider(self):
20         href_list = ['url1', 'url2', 'url3', 'url4']
21         for href in href_list:
22             href_md5 = self.url_md5(href)
23             if self.r.sadd('spider:urls', href_md5) == 1:
24                 返回值为1表示添加成功, 即之前未抓取过, 则开始抓取
25             else:
26                 sys.exit()

```

Day03笔记

百度图片抓取

■ 案例解析

```

1  【1】百度图片官网指定图片抓取:
2      1.1> 百度图片官网: http://image.baidu.com/
3      1.2> 运行效果

```

```
4      请输入关键字：赵丽颖
5      则自动创建文件夹：/home/tarena/images/赵丽颖/ 并把首页30张图片保存到此文件夹下
6
7  【2】注意
8      2.1> 一定要以响应内容为主来写正则表达式（右键 - 查看网页源代码）
9
10 【3】颠覆前两天课程认知的一个现实
11     3.1> 页面结构 - Elements，为页面最终渲染完成后的结构，和响应内容不一定完全一样
12     3.2> 原因1：可能会有部分数据为动态加载的
13           原因2：响应内容中存在JavaScript，对页面结构做了一定调整
14
15 【4】那我们写正则表达式时要以谁为准？
16     4.1> 必须以响应内容为准!!!!!! -> 右键，查看网页源代码为准
17     4.1> 必须以响应内容为准!!!!!! -> 右键，查看网页源代码为准
18     4.1> 必须以响应内容为准!!!!!! -> 右键，查看网页源代码为准
19
20     @@ 重要的事情说三遍，必须以响应内容为准 @@
```

■ 实现步骤

```
1  【1】右键，查看网页源码，搜索图片链接关键字 -> 存在
2  【2】分析URL地址规律
3      https://image.baidu.com/search/index?tn=baiduimage&word={}
4  【3】正则表达式 - 以响应内容为准
5      "thumbURL": "(.*?)"
6  【4】代码实现
7      4.1> 知识点1 - Windows中路径如何表示
8            方式1: E:\\spider\\spider_day03\\
9            方式2: E:/spider/spider_day03/
10
11     4.2> 如何生成随机的User-Agent
12           sudo pip3 install fake_useragent
13           from fake_useragent import UserAgent
14           user_agent = UserAgent().random
```

■ 代码实现

```
1  import requests
2  import re
3  import time
4  import random
5  from fake_useragent import UserAgent
6  import os
7  from urllib import parse
8
9  class BaiduImageSpider(object):
10     def __init__(self):
11         self.url = 'https://image.baidu.com/search/index?tn=baiduimage&word={}'
12         self.word = input('请输入关键字:')
13         self.directory = '/home/tarena/images/{}/'.format(self.word)
14         if not os.path.exists(self.directory):
15             os.makedirs(self.directory)
16
17         self.i = 1
18
```

```

19     def get_images(self,one_url):
20         # 使用随机的User-Agent
21         headers = { 'User-Agent':UserAgent().random }
22         one_html = requests.get(url=one_url,headers=headers).text
23         regex = '"thumbURL": "(.*?)"'
24         pattern = re.compile(regex,re.S)
25         image_src_list = pattern.findall(one_html)
26         for image_src in image_src_list:
27             self.save_image(image_src)
28             # 控制爬取速度
29             time.sleep(random.uniform(0,1))
30
31     def save_image(self,image_src):
32         # 每次请求使用随机的User-Agent
33         headers = { 'User-Agent':UserAgent().random }
34         image_html = requests.get(url=image_src,headers=headers).content
35         filename = '{}{}_{}.jpg'.format(self.directory,self.word,self.i)
36         with open(filename,'wb') as f:
37             f.write(image_html)
38         print(filename,'下载成功')
39         self.i += 1
40
41     def run(self):
42         params = parse.quote(self.word)
43         one_url = self.url.format(params)
44         self.get_images(one_url)
45
46 if __name__ == '__main__':
47     spider = BaiduImageSpider()
48     spider.run()

```

Chrome浏览器安装插件

■ 安装方法

- 1 **【1】在线安装**
- 2 1.1> 下载插件 - google访问助手
- 3 1.2> 安装插件 - google访问助手: Chrome浏览器-设置-更多工具-扩展程序-开发者模式-拖拽(解压后的插件)
- 4 1.3> 在线安装其他插件 - 打开google访问助手 - google应用商店 - 搜索插件 - 添加即可
- 5
- 6 **【2】离线安装**
- 7 2.1> 下载插件 - xxx.crx 重命名为 xxx.zip
- 8 2.2> Chrome浏览器-设置-更多工具-扩展程序-开发者模式
- 9 2.3> 拖拽 插件(或者解压后文件夹) 到浏览器中
- 10 2.4> 重启浏览器, 使插件生效

■ 爬虫常用插件

- 1 【1】 google-access-helper : 谷歌访问助手,可访问 谷歌应用商店
- 2 【2】 Xpath Helper: 轻松获取HTML元素的XPath路径
- 3 打开/关闭: Ctrl + Shift + x
- 4 【3】 JsonView: 格式化输出json格式数据
- 5 【4】 Proxy SwitchyOmega: Chrome浏览器中的代理管理扩展程序

xpath解析

▪ 定义

- 1 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

▪ 示例HTML代码

```
1 <ul class="CarList">
2   <li class="bjd" id="car_001" href="http://www.bjd.com/">
3     <p class="name">布加迪</p>
4     <p class="model">威航</p>
5     <p class="price">2500万</p>
6     <p class="color">红色</p>
7   </li>
8
9   <li class="byd" id="car_002" href="http://www.byd.com/">
10    <p class="name">比亚迪</p>
11    <p class="model">秦</p>
12    <p class="price">15万</p>
13    <p class="color">白色</p>
14  </li>
15 </ul>
```

▪ 匹配演示

- 1 【1】 查找所有的li节点
- 2 //li
- 3 【2】 获取所有汽车的名称: 所有li节点下的子节点p的值 (class属性值为name)
- 4 //li/p[@class="name"]
- 5 //p[@class="name"]
- 6 //ul[@class="CarList"]/li/p[@class="name"]
- 7 【3】 获取ul节点下第2个li节点的汽车信息: 找比亚迪车的信息
- 8 //ul[@class="CarList"]/li[2]/p
- 9 【4】 获取所有汽车的链接: ul节点下所有li子节点的href属性的值
- 10 //ul[@class="CarList"]/li/@href
- 11
- 12 【注意】
- 13 1> 只要涉及到条件,加 [] : //li[@class="xxx"] //li[2]
- 14 2> 只要获取属性值,加 @ : //li[@class="xxx"] //li/@href

▪ 选取节点

- 1 【1】 // : 从所有节点中查找 (包括子节点和后代节点)

```

2  【2】@ ：获取属性值
3  2.1> 使用场景1（属性值作为条件）
4      //div[@class="movie-item-info"]
5  2.2> 使用场景2（直接获取属性值）
6      //div[@class="movie-item-info"]/a/img/@src
7
8  【3】练习 - 猫眼电影top100
9  3.1> 匹配电影名称
10     //div[@class="movie-item-info"]/p[1]/a/text()
11  3.2> 匹配电影主演
12     //div[@class="movie-item-info"]/p[2]/text()
13  3.3> 匹配上映时间
14     //div[@class="movie-item-info"]/p[3]/text()
15  3.4> 匹配电影链接
16     //div[@class="movie-item-info"]/p[1]/a/@href

```

■ 匹配多路径（或）

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

■ 常用函数

```

1  【1】contains() ：匹配属性值中包含某些字符串节点
2  1.1> 查找id属性值中包含字符串 "car_" 的 li 节点
3      //li[contains(@id,"car_")]
4
5  【2】text() ：获取节点的文本内容
6  2.1> 查找所有汽车的价格
7      //ul[@class="CarList"]/li/p[@class="price"]/text()

```

■ 终极总结

```

1  【1】xpath表达式的末尾为：/text()、/@href 得到的列表中为'字符串'
2
3  【2】其他剩余所有情况得到的列表中均为'节点对象'
4  [<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]
5  [<element div at xxxa>,<element div at xxxb>]
6  [<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]

```

■ 课堂练习

```

1  【1】匹配汽车之家-二手车,所有汽车的链接 :
2      //li[@class="cards-li list-photo-li"]/a[1]/@href
3      //a[@class="carinfo"]/@href
4  【2】匹配汽车之家-汽车详情页中,汽车的
5  2.1)名称: //div[@class="car-box"]/h3/text()
6  2.2)里程: //ul/li[1]/h4/text()
7  2.3)时间: //ul/li[2]/h4/text()
8  2.4)挡位+排量: //ul/li[3]/h4/text()
9  2.5)所在地: //ul/li[4]/h4/text()
10 2.6)价格: //div[@class="brand-price-item"]/span[@class="price"]/text()

```

lxml解析库

■ 安装

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

■ 使用流程

```
1 1、导模块
2     from lxml import etree
3 2、创建解析对象
4     parse_html = etree.HTML(html)
5 3、解析对象调用xpath
6     r_list = parse_html.xpath('xpath表达式')
```

■ xpath最常用

```
1 【1】基准xpath: 匹配所有电影信息的节点对象列表
2     //dl[@class="board-wrapper"]/dd
3     [<element dd at xxx>,<element dd at xxx>,...]
4
5 【2】遍历对象列表, 依次获取每个电影信息
6     item = {}
7     for dd in dd_list:
8         item['name'] = dd.xpath('..//p[@class="name"]/a/text()').strip()
9         item['star'] = dd.xpath('..//p[@class="star"]/text()').strip()[3:]
10        item['time'] = dd.xpath('..//p[@class="releasetime"]/text()').strip()[5:15]
```

■ 猫眼电影-xpath

```
1 """
2 猫眼电影 - xpath
3 """
4 import requests
5 from lxml import etree
6 import time
7 import random
8
9 class MaoyanSpider(object):
10     def __init__(self):
11         self.url = 'https://maoyan.com/board/4?offset={}'
12         self.headers = {
13             'Accept-Encoding': 'gzip, deflate, br',
14             'Upgrade-Insecure-Requests': '1',
15             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36',
16         }
17         # 计数变量
18         self.i = 0
19
20     def get_html(self,url):
21         html = requests.get(url=url,headers=self.headers).text
```



```

22         # 直接调用解析函数
23         self.parse_html(html)
24
25     def parse_html(self,html):
26         p = etree.HTML(html)
27         item = {}
28         # 1.基准xpath: dd节点对象列表 [dd1,dd2,dd3]
29         dd_list = p.xpath('//dl[@class="board-wrapper"]/dd')
30         # 2.for循环遍历,依次提取每个电影信息
31         for dd in dd_list:
32             item['name'] = dd.xpath('.//p[@class="name"]/a/text()')[0]
33             item['star'] = dd.xpath('.//p[@class="star"]/text()')[0].strip()
34             item['releasetime'] = dd.xpath('.//p[@class="releasetime"]/text()')[0].strip()
35
36             print(item)
37             self.i += 1
38
39     def save_html(self,film_list):
40         item = {}
41         for film in film_list:
42             item['name'] = film[0].strip()
43             item['star'] = film[1].strip()
44             item['time'] = film[2].strip()[5:15]
45             print(item)
46             self.i += 1
47
48     def run(self):
49         for offset in range(0,91,10):
50             url = self.url.format(offset)
51             self.get_html(url)
52             # 休眠
53             time.sleep(random.uniform(0,1))
54             print('数量:',self.i)
55
56 if __name__ == '__main__':
57     start = time.time()
58     spider = MaoyanSpider()
59     spider.run()
60     end = time.time()
61     print('执行时间:%.2f' % (end-start))

```

链家二手房案例 (xpath)

实现步骤

■ 确定是否为静态

1 | 打开二手房页面 -> 查看网页源码 -> 搜索关键字

■ xpath表达式

```

1  【1】基准xpath表达式(匹配每个房源信息节点列表)
2      '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'
3      //ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]
4
5  【2】依次遍历后每个房源信息xpath表达式
6      2.1)名称: .//div[@class="positionInfo"]/a[1]/text()
7      2.2)地址: .//div[@class="positionInfo"]/a[2]/text()
8      2.3)户型+面积+方位+是否精装+楼层+年代+类型
9          info_list: './/div[@class="houseInfo"]/text()' -> [0].strip().split('|')
10         a)户型: info_list[0]
11         b)面积: info_list[1]
12         c)方位: info_list[2]
13         d)精装: info_list[3]
14         e)楼层: info_list[4]
15         f)年代: info_list[5]
16         g)类型: info_list[6]
17
18     2.4)总价+单价
19         a)总价: .//div[@class="totalPrice"]/span/text()
20         b)单价: .//div[@class="unitPrice"]/span/text()

```

■ 代码实现

```

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          # 有问题页面,尝试3次,如果不行直接抓取下一页数据
14          for i in range(3):
15              try:
16                  html = requests.get(url=url,headers=headers,timeout=3).content.decode('utf-8','ignore')
17                  self.get_data(html)
18                  break
19              except Exception as e:
20                  print('Retry')
21
22
23      def get_data(self,html):
24          p = etree.HTML(html)
25          # 基准xpath: [<element li at xxx>,<element li>]
26          li_list = p.xpath('//ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]')
27          # for遍历,依次提取每个房源信息,放到字典item中
28          item = {}
29          for li in li_list:
30              # 名称+区域
31              name_list = li.xpath('.//div[@class="positionInfo"]/a[1]/text()')

```

```

32         item['name'] = name_list[0].strip() if name_list else None
33         address_list = li.xpath('..//div[@class="positionInfo"]/a[2]/text()')
34         item['address'] = address_list[0].strip() if address_list else None
35         # 户型+面积+方位+是否精装+楼层+年代+类型
36         # h_list: ['']
37         h_list = li.xpath('..//div[@class="houseInfo"]/text()')
38         if h_list:
39             info_list = h_list[0].split('|')
40             if len(info_list) == 7:
41                 item['model'] = info_list[0].strip()
42                 item['area'] = info_list[1].strip()
43                 item['direct'] = info_list[2].strip()
44                 item['perfect'] = info_list[3].strip()
45                 item['floor'] = info_list[4].strip()
46                 item['year'] = info_list[5].strip()[:-2]
47                 item['type'] = info_list[6].strip()
48             else:
49                 item['model'] = item['area'] = item['direct'] = item['perfect'] =
item['floor'] = item['year'] = item['type'] = None
50             else:
51                 item['model'] = item['area'] = item['direct'] = item['perfect'] =
item['floor'] = item['year'] = item['type'] = None
52
53         # 总价+单价
54         total_list = li.xpath('..//div[@class="totalPrice"]/span/text()')
55         item['total'] = total_list[0].strip() if total_list else None
56         unit_list = li.xpath('..//div[@class="unitPrice"]/span/text()')
57         item['unit'] = unit_list[0].strip() if unit_list else None
58
59         print(item)
60
61     def run(self):
62         for pg in range(1,101):
63             url = self.url.format(pg)
64             self.parse_html(url)
65             time.sleep(random.randint(1,2))
66
67 if __name__ == '__main__':
68     spider = LianjiaSpider()
69     spider.run()

```

■ 后续自己完成

- 1 【1】将数据存入MongoDB数据库
- 2 【2】将数据存入MySQL数据库

requests.get()参数

查询参数-params

■ 参数类型

1 | 字典,字典中键值对作为查询参数

■ 使用方法

```
1 | 1、 res = requests.get(url=baseurl,params=params,headers=headers)
2 | 2、 特点:
3 |     * url为基准的url地址, 不包含查询参数
4 |     * 该方法会自动对params字典编码,然后和url拼接
```

■ 示例

```
1 | import requests
2 |
3 | baseurl = 'http://tieba.baidu.com/f?'
4 | params = {
5 |     'kw' : '赵丽颖吧',
6 |     'pn' : '50'
7 | }
8 | headers = {'User-Agent' : 'Mozilla/4.0'}
9 | # 自动对params进行编码,然后自动和url进行拼接,去发请求
10 | html = requests.get(url=baseurl,params=params,headers=headers).content.decode()
```

SSL证书认证参数-verify

■ 适用网站及场景

```
1 | 【1】适用网站: https类型网站但是没有经过 证书认证机构 认证的网站
2 | 【2】适用场景: 抛出 SSLError 异常则考虑使用此参数
```

■ 参数类型

```
1 | 【1】verify=True(默认) : 检查证书认证
2 | 【2】verify=False (常用) : 忽略证书认证
3 | 【3】示例
4 |     res = requests.get(url=url,params=params,headers=headers,verify=False)
```

代理参数-proxies

■ 定义及分类

```
1 【1】定义：代替你原来的IP地址去对接网络的IP地址
2
3 【2】作用：隐藏自身真实IP,避免被封
4
5 【3】种类
6 3.1) 高匿代理：Web端只能看到代理IP
7 3.2) 普通代理：Web端知道有人通过此代理IP访问，但不知用户真实IP
8 3.3) 透明代理：Web能看到用户真实IP，也能看到代理IP
```

■ 普通代理

```
1 【1】获取代理IP网站
2 西刺代理、快代理、全网代理、代理精灵、... ...
3
4 【2】参数类型
5 proxies = { '协议': '协议://IP:端口号' }
6 proxies = {
7     'http': 'http://IP:端口号',
8     'https': 'https://IP:端口号',
9 }
```

■ 普通代理 - 示例

```
1 # 使用免费普通代理IP访问测试网站: http://httpbin.org/get
2 import requests
3
4 url = 'http://httpbin.org/get'
5 headers = {'User-Agent': 'Mozilla/5.0'}
6 # 定义代理,在代理IP网站中查找免费代理IP
7 proxies = {
8     'http': 'http://112.85.164.220:9999',
9     'https': 'https://112.85.164.220:9999'
10 }
11 html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
12 print(html)
```