

# DAY03

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

## requests模块

## ▪ 安装

```

1 # 1. Linux
2 sudo pip3 install requests
3
4 # 2. Windows
5 方法1: cmd命令行 -> python -m pip install requests
6 方法2: 右键管理员进入cmd命令行 : pip install requests

```

#### ■ requests.get()

```

1 # 1. 作用
2 向目标网站发起请求,并获取响应对象
3 res = requests.get(url=url,headers=headers)
4
5 # 2. 参数
6 1、url : 需要抓取的URL地址
7 2、headers : 请求头
8 3、timeout : 超时时间, 超过时间会抛出异常
9
10 # 3. 响应对象(res)属性
11 1、text : 字符串
12 2、content : 字节流
13 3、status_code : HTTP响应码
14 4、url : 实际数据的URL地址

```

#### ■ 非结构化数据保存

```

1 with open('xxx.jpg','wb') as f:
2     f.write(res.content)

```

#### ■ 示例代码 - 图片抓取

```

1 # 保存赵丽颖图片到本地
2
3 import requests
4
5 url = 'https://timgsa.baidu.com/timg?
6 image&quality=80&size=b9999_10000&sec=1567090051520&di=77e8b97b3280f999cf51340af4315b4b&img
7 type=jpg&src=http%3A%2F%2F5b0988e595225.cdn.sohucs.com%2Fimages%2F20171121%2F4e6759d153d04c
8 6badbb0a5262ec103d.jpeg'
9 headers = {'User-Agent': 'Mozilla/5.0'}
10
11 html = requests.get(url=url,headers=headers).content
12 with open('花千骨.jpg','wb') as f:
13     f.write(html)

```

#### ■ 课堂练习

```

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 【2】 注意

```

```
7 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)
8
9 【3】 颠覆前两天课程认知的一个现实
10 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
11 3.2> 原因1: 可能会有部分数据为动态加载的
12 原因2: 响应内容中存在JavaScript, 对页面结构做了一定调整
13
14 【4】 那我们写正则表达式时要以谁为准?
15 4.1> 必须以响应内容为准!!!!!! -> 右键, 查看网页源代码为准
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
19 @@ 重要的事情说三遍, 必须以响应内容为准 @@
```

## ■ 百度图片抓取实现步骤

```
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
```

## ■ 百度图片代码实现

```
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 【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
6   //ul[@class="CarList"]/li[2]/p
7 【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
8   //ul[@class="CarList"]/li/@href
9
10 【注意】
11   1> 只要涉及到条件,加 []
12   2> 只要获取属性值,加 @
```

## ▪ 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
11   3.2> 匹配电影主演
12
```

```
13 3.3> 匹配上映时间
14
15 3.4> 匹配电影链接
16
```

## ■ 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 【2】 匹配汽车之家-汽车详情页中,汽车的
3 2.1)名称:
4 2.2)里程:
5 2.3)时间:
6 2.4)挡位+排量:
7 2.5)所在地:
8 2.6)价格:
```

# lxml解析库

## ■ 安装

```
1 | sudo pip3 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 运行此代码前, 请执行如下操作:
3 1、清除浏览器缓存
4 2、输入地址: https://maoyan.com/board/4?offset=0
5 3、如果出现验证, 则手动滑动, 跳过美团滑块验证
6 """
7 from urllib import request
8 from lxml import etree
9 import time
10 import random
11
12 class MaoyanSpider(object):
13     def __init__(self):
14         self.url = 'https://maoyan.com/board/4?offset={}'
15         self.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)'}
16         # 计数变量
17         self.i = 0
18
19     def get_html(self, url):
20         req = request.Request(url=url, headers=self.headers)
21         resp = request.urlopen(req)
22         html = resp.read().decode()
23         # 直接调用解析函数
24         self.parse_html(html)
25
26     def parse_html(self, html):
27         p = etree.HTML(html)
28         item = {}
29         # 1. 基准xpath: dd节点对象列表 [dd1, dd2, dd3]
30         dd_list = p.xpath('//dl[@class="board-wrapper"]/dd')
31         # 2. for循环遍历, 依次提取每个电影信息

```

```

32         for dd in dd_list:
33             item['name'] = dd.xpath('.//p[@class="name"]/a/text()')[0]
34             item['star'] = dd.xpath('.//p[@class="star"]/text()')[0].strip()
35             item['releasetime'] = dd.xpath('.//p[@class="releasetime"]/text()')[0].strip()
[5:15]
36
37             print(item)
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]

```



```

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()
```

## ■ 练习

1 | 把抓取百度贴吧的案例改为 params 参数实现

# 作业 - 百度贴吧图片抓取

## 目标思路

## ■ 目标

1 | 抓取指定贴吧所有图片

## ■ 思路

```
1 【1】获取贴吧主页URL,下一页,找到不同页的URL规律
2 【2】获取1页中所有帖子URL地址: [帖子链接1,帖子链接2,...]
3 【3】对每个帖子链接发请求,获取图片URL列表: [图片链接1,图片链接2,...]
4 【4】依次向图片的URL发请求,以wb方式写入本地文件
```

## 实现步骤

## ■ 贴吧URL规律

1 | http://tieba.baidu.com/f?kw=?&pn=50

## ■ xpath表达式

```
1  【1】帖子链接xpath
2      //div[@class="t_con cleafix"]/div/div/div/a/@href
3
4  【2】图片链接xpath
5      //div[@class="d_post_content j_d_post_content clearfix"]/img[@class="BDE_Image"]/@src
6
7  【3】视频链接xpath
8      //div[@class="video_src_wrapper"]/embed/@data-video
9
10 【4】注意
11     4.1) 务必务必使用IE的User-Agent
12     4.2) 一切以响应内容为主
13     4.3) 视频链接前端对响应内容做了处理,需要查看网页源代码来查看, 复制HTML代码在线格式化
```