

# Day02回顾

## 请求模块(requests)

```
1 | html = requests.get(url=url,headers=headers).text
2 | html = requests.get(url=url,headers=headers).content.decode('utf-8')
3 |
4 | with open('xxx.txt','w',encoding='utf-8') as f:
5 |     f.write(html)
```

## 编码模块(urllib.parse)

```
1 | 1、urlencode({dict})
2 |     urlencode({'wd':'美女','pn':'20'})
3 |     编码后 : 'wd=%E8%D5XXX&pn=20'
4 |
5 | 2、quote(string)
6 |     quote('织女')
7 |     编码后 : '%D3%F5XXX'
8 |
9 | 3、unquote('%D3%F5XXX')
```

## 解析模块(re)

### ■ 使用流程

```
1 | pattern = re.compile('正则表达式',re.S)
2 | r_list = pattern.findall(html)
```

### ■ 贪婪匹配和非贪婪匹配

```
1 | 贪婪匹配(默认) : .*
2 | 非贪婪匹配      : .*?
```

### ■ 正则表达式分组

```
1  【1】 想要什么内容在正则表达式中加()  
2  【2】 多个分组,先按整体正则匹配,然后再提取()中数据。  
3      结果1(未匹配到数据): []  
4      结果2(只有一个分组): [' ', ' ', ' ']  
5      结果3(正则多个分组): [(), (), ()]
```

## 抓取步骤

```
1  【1】 确定所抓取数据在响应中是否存在 (右键 - 查看网页源码 - 搜索关键字)  
2  【2】 数据存在: 查看URL地址规律  
3  【3】 写正则表达式,来匹配数据  
4  【4】 程序结构  
5      a>每爬取1个页面后随机休眠一段时间
```

```
1  # 程序结构  
2  class xxxSpider(object):  
3      def __init__(self):  
4          # 定义常用变量,url,headers及计数等  
5  
6      def get_html(self):  
7          # 获取响应内容函数,使用随机User-Agent  
8  
9      def parse_html(self):  
10         # 使用正则表达式来解析页面,提取数据  
11  
12         def save_html(self):  
13             # 将提取的数据按要求保存, csv、MySQL数据库等  
14  
15         def run(self):  
16             # 程序入口函数,用来控制整体逻辑  
17  
18  if __name__ == '__main__':  
19     # 程序开始运行时间戳  
20     start = time.time()  
21     spider = xxxSpider()  
22     spider.run()  
23     # 程序运行结束时间戳  
24     end = time.time()  
25     print('执行时间:%.2f' % (end-start))
```

## 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.db.commit()
10
11 # run(self):
12     self.cursor.close()
13     self.db.close()

```

# spider-day03笔记

## 瓜子二手车数据抓取 - 二级页面

### ▪ 领取任务

```

1 【1】 爬取地址
2     瓜子网 - 我要买车
3     https://www.guazi.com/bj/buy/
4
5 【2】 爬取目标
6     所有汽车的 汽车名称、行驶里程、排量、变速箱、价格
7
8 【3】 爬取分析
9     *****一级页面需抓取*****
10    1、车辆详情页的链接
11
12    *****二级页面需抓取*****
13    1、汽车名称
14    2、行驶里程
15    3、排量
16    4、变速箱
17    5、价格

```

### ▪ 实现步骤

```

1  【1】确定响应内容中是否存在所需抓取数据 - 存在
2
3  【2】找URL地址规律
4      第1页: https://www.guazi.com/bj/buy/o1/#bread
5      第2页: https://www.guazi.com/bj/buy/o2/#bread
6      第n页: https://www.guazi.com/bj/buy/o{}/#bread
7
8  【3】写正则表达式
9      一级页面正则表达式:<li data-scroll-track=.*?href="(.*?)>
10
11     二级页面正则表达式:<div class="product-textbox">.*?<h2 class="titlebox">(.*?)</h2>.*?
12     <li class="two"><span>(.*?)</span>.*?<li class="three"><span>(.*?)</span>.*?<li
13     class="last"><span>(.*?)</span>.*?<span class="price-num">(.*?)</span>

```

## ■ 代码实现

```

1  import requests
2  import re
3  import time
4  import random
5
6  class GuaziSpider:
7      def __init__(self):
8          self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
9          self.headers = {
10              'Cookie': 'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=9858835725989223197831; sessionid=5c4c7246-25a1-4a16-8adb-678af786a472; lg=1; lng_lat=116.84757_39.8668; gps_type=1; close_finance_popup=2020-10-13; cainfo=%7B%22ca_a%22%3A%22-%22%2C%22ca_b%22%3A%22-%22%2C%22ca_s%22%3A%22self%22%2C%22ca_n%22%3A%22self%22%2C%22ca_medium%22%3A%22-%22%2C%22ca_term%22%3A%22-%22%2C%22ca_content%22%3A%22-%22%2C%22ca_campaign%22%3A%22-%22%2C%22ca_kw%22%3A%22-%22%2C%22ca_i%22%3A%22-%22%2C%22scode%22%3A%22-%22%2C%22keyword%22%3A%22-%22%2C%22ca_keywordid%22%3A%22-%22%2C%22display_finance_flag%22%3A%22-%22%2C%22platform%22%3A%221%22%2C%22version%22%3A%221%22%2C%22client_ab%22%3A%22-%22%2C%22guid%22%3A%221c286513-a2e1-4d4d-e9d5-231da3e8ee16%22%2C%22ca_city%22%3A%22langfang%22%2C%22sessionid%22%3A%225c4c7246-25a1-4a16-8adb-678af786a472%22%7D; cityDomain=bj; user_city_id=12; preTime=%7B%22last%22%3A%221602604364%22%22this%22%3A%221602604337%22%22pre%22%3A%221602604337%22%7D',
11              'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
12          }
13
14      def get_html(self, url):
15          """请求功能函数: 获取html"""
16          html = requests.get(url=url, headers=self.headers).content.decode('utf-8', 'ignore')
17
18          return html
19
20      def re_func(self, regex, html):

```

```

21     """解析功能函数：正则解析得到列表"""
22     pattern = re.compile(regex, re.S)
23     r_list = pattern.findall(html)
24
25     return r_list
26
27 def parse_html(self, one_url):
28     """爬虫逻辑函数"""
29     one_html = self.get_html(url=one_url)
30     one_regex = '<li data-scroll-track=.*?href="(.*?)"'
31     href_list = self.re_func(regex=one_regex, html=one_html)
32     for href in href_list:
33         two_url = 'https://www.guazi.com' + href
34         # 获取一辆汽车详情页的具体数据
35         self.get_one_car_info(two_url)
36         # 控制数据抓取的频率
37         time.sleep(random.uniform(0, 1))
38
39 def get_one_car_info(self, two_url):
40     """获取一辆汽车的具体数据"""
41     # 名称、行驶里程、排量、变速箱、价格
42     two_html = self.get_html(url=two_url)
43     two_regex = '<div class="product-textbox">.*?<h2 class="titlebox">(.*?)</h2>.*?<li class="two"><span>(.*?)</span>.*?<li class="three"><span>(.*?)</span>.*?<li class="last"><span>(.*?)</span>.*?<span class="price-num">(.*?)</span>'
44     car_info_list = self.re_func(regex=two_regex, html=two_html)
45     # 获取具体数据
46     item = {}
47     item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
48     item['km'] = car_info_list[0][1].strip()
49     item['displace'] = car_info_list[0][2].strip()
50     item['type'] = car_info_list[0][3].strip()
51     item['price'] = car_info_list[0][4].strip()
52     print(item)
53
54 def run(self):
55     for o in range(1, 6):
56         one_url = self.url.format(o)
57         self.parse_html(one_url=one_url)
58
59 if __name__ == '__main__':
60     spider = GuaziSpider()
61     spider.run()

```

## ■ 练习 - 将数据存入MySQL数据库

```

1 create database guazidb charset utf8;
2 use guazidb;
3 create table guazitab(
4     name varchar(200),
5     km varchar(100),
6     displace varchar(100),
7     type varchar(100),
8     price varchar(100)
9 )charset=utf8;

```

## ■ 使用redis实现增量爬虫-redis集合实现

```
1 import requests
2 import re
3 import time
4 import random
5 import pymysql
6 import redis
7 import sys
8 from hashlib import md5
9
10
11 class GuaziSpider:
12     def __init__(self):
13         self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
14         self.headers = {
15             'Cookie': 'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=9858835725989223197831; sessionid=5c4c7246-25a1-4a16-8adb-678af786a472; lg=1; lng_lat=116.84757_39.86668; gps_type=1; close_finance_popup=2020-10-13; cainfo=%7B%22ca_a%22%3A%22-%22%2C%22ca_b%22%3A%22-%22%2C%22ca_s%22%3A%22self%22%2C%22ca_n%22%3A%22self%22%2C%22ca_medium%22%3A%22-%22%2C%22ca_term%22%3A%22-%22%2C%22ca_content%22%3A%22-%22%2C%22ca_campaign%22%3A%22-%22%2C%22ca_kw%22%3A%22-%22%2C%22ca_i%22%3A%22-%22%2C%22scode%22%3A%22-%22%2C%22keyword%22%3A%22-%22%2C%22ca_keywordid%22%3A%22-%22%2C%22display_finance_flag%22%3A%22-%22%2C%22platform%22%3A%221%22%2C%22version%22%3A1%2C%22client_ab%22%3A%22-%22%2C%22guid%22%3A%221c286513-a2e1-4d4d-e9d5-231da3e8ee16%22%2C%22ca_city%22%3A%22langfang%22%2C%22sessionid%22%3A%225c4c7246-25a1-4a16-8adb-678af786a472%22%7D; cityDomain=bj; user_city_id=12; preTime=%7B%22last%22%3A1602604364%2C%22this%22%3A1602604337%2C%22pre%22%3A1602604337%7D',
16             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
17         }
18         # 连接MySQL
19         self.db = pymysql.connect('localhost', 'root', '123456', 'guazidb', charset='utf8')
20         self.cur = self.db.cursor()
21         # 连接redis
22         self.r = redis.Redis(host='localhost', port=6379, db=0)
23
24     def get_html(self, url):
25         """请求功能函数：获取html"""
26         html = requests.get(url=url, headers=self.headers).content.decode('utf-8', 'ignore')
27
28         return html
29
30     def re_func(self, regex, html):
31         """解析功能函数：正则解析得到列表"""
32         pattern = re.compile(regex, re.S)
33         r_list = pattern.findall(html)
34
35         return r_list
36
```

```

37 def md5_url(self, url):
38     """功能函数3: md5加密生成指纹"""
39     s = md5()
40     s.update(url.encode())
41
42     return s.hexdigest()
43
44 def parse_html(self, one_url):
45     """爬虫逻辑函数"""
46     one_html = self.get_html(url=one_url)
47     one_regex = '<li data-scroll-track=.*?href="(.*?)"'
48     href_list = self.re_func(regex=one_regex, html=one_html)
49     for href in href_list:
50         two_url = 'https://www.guazi.com' + href
51         # 生成指纹
52         finger = self.md5_url(url=two_url)
53         # 如果添加成功说明之前未抓取过
54         if self.r.sadd('guazi:spider', finger) == 1:
55             # 获取一辆汽车详情页的具体数据
56             self.get_one_car_info(two_url)
57             # 控制数据抓取的频率
58             time.sleep(random.uniform(0, 1))
59         else:
60             # 否则结束程序
61             sys.exit('更新完成')
62
63 def get_one_car_info(self, two_url):
64     """获取一辆汽车的具体数据"""
65     # 名称、行驶里程、排量、变速箱、价格
66     two_html = self.get_html(url=two_url)
67     two_regex = '<div class="product-textbox">.*?<h2 class="titlebox">(.*?)</h2>.*?<li class="two"><span>(.*?)</span>.*?<li class="three"><span>(.*?)</span>.*?<li class="last"><span>(.*?)</span>.*?<span class="price-num">(.*?)</span>'
68     car_info_list = self.re_func(regex=two_regex, html=two_html)
69     # 获取具体数据
70     item = {}
71     item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
72     item['km'] = car_info_list[0][1].strip()
73     item['displace'] = car_info_list[0][2].strip()
74     item['type'] = car_info_list[0][3].strip()
75     item['price'] = car_info_list[0][4].strip()
76     print(item)
77
78     li = [item['name'], item['km'], item['displace'], item['type'], item['price']]
79     ins = 'insert into guazitab values(%s,%s,%s,%s,%s)'
80     self.cur.execute(ins, li)
81     self.db.commit()
82
83
84 def run(self):
85     for o in range(1, 2):
86         one_url = self.url.format(o)
87         self.parse_html(one_url=one_url)
88     # 断开数据库
89     self.cur.close()
90     self.db.close()
91

```

```
92 if __name__ == '__main__':
93     spider = GuaziSpider()
94     spider.run()
```

## 汽车之家数据抓取 - 二级页面

### ■ 领取任务

```
1  【1】 爬取地址
2      汽车之家 - 二手车 - 价格从低到高
3      https://www.che168.com/beijing/a0_0msdgsncngpi1lto1csp1exx0/
4
5
6  【2】 爬取目标
7      所有汽车的 型号、行驶里程、上牌时间、档位、排量、车辆所在地、价格
8
9  【3】 爬取分析
10     *****一级页面需抓取*****
11         1、车辆详情页的链接
12
13     *****二级页面需抓取*****
14         1、名称
15         2、行驶里程
16         3、上牌时间
17         4、档位
18         5、排量
19         6、车辆所在地
20         7、价格
```

### ■ 实现步骤

```
1  【1】 确定响应内容中是否存在所需抓取数据 - 存在
2
3  【2】 找URL地址规律
4      第1页: https://www.che168.com/beijing/a0_0msdgsncngpi1lto1csp1exx0/
5      第2页: https://www.che168.com/beijing/a0_0msdgsncngpi1lto1csp2exx0/
6      第n页: https://www.che168.com/beijing/a0_0msdgsncngpi1lto1csp{ }exx0/
7
8  【3】 写正则表达式
9      一级页面正则表达式: <li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>
10     二级页面正则表达式: <div class="car-box">.*?<h3 class="car-brand-name">(.*?)</h3>.*?
    <ul class="brand-unit-item fn-clear">.*?<li>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>
    (.*?)</h4>.*?<h4>(.*?)</h4>.*?<span class="price" id="overlayPrice">¥(.*?)<b>
11
12  【4】 代码实现
13  <div class="car-box">.*?<h3 class="car-brand-name">(.*?)</h3>.*?<h4>(.*?)</h4>.*?<h4>
    (.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<span class="price" id="overlayPrice">¥
    (.*?)<b>
```

### ■ 代码实现

```
1  """
```



```

2 汽车之家二手车数据抓取
3 分析:
4     1. 一级页面: 每辆汽车详情页的链接
5     2. 二级页面: 每辆汽车具体的数据
6     """
7  import requests
8  import re
9  import time
10 import random
11 from fake_useragent import UserAgent
12 import pymongo
13
14 class CarSpider:
15     def __init__(self):
16         self.url = 'https://www.che168.com/beijing/a0_0msdgcncgpi1lto1csp{}exx0/?
pvareaid=102179#currngpostion'
17         # 3个对象
18         self.conn = pymongo.MongoClient('localhost', 27017)
19         self.db = self.conn['cardb']
20         self.myset = self.db['carset']
21
22     def get_html(self, url):
23         """功能函数1: 请求功能函数"""
24         headers = {'User-Agent': UserAgent().random}
25         # ignore参数: 解码时遇到不识别的字符直接忽略掉
26         html = requests.get(url=url, headers=headers).content.decode('gb2312',
'ignore')
27
28         return html
29
30     def re_func(self, regex, html):
31         """功能函数2: 解析功能函数"""
32         pattern = re.compile(regex, re.S)
33         r_list = pattern.findall(html)
34
35         return r_list
36
37     def parse_html(self, one_url):
38         """爬虫逻辑函数"""
39         one_html = self.get_html(url=one_url)
40         one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
41         # href_list: ['/declare/xxx.html', '', '', '', ...]
42         href_list = self.re_func(one_regex, one_html)
43         for href in href_list:
44             # 拼接完整URL地址,发请求提取具体汽车信息
45             self.get_one_car_info(href)
46             time.sleep(random.randint(1, 2))
47
48     def get_one_car_info(self, href):
49         """提取一辆汽车的具体信息"""
50         two_url = 'https://www.che168.com' + href
51         two_html = self.get_html(url=two_url)
52         two_regex = '<div class="car-box">.*?<h3 class="car-brand-name">(.*?)</h3>.*?
<ul class="brand-unit-item fn-clear">.*?<li>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>
(.*?)</h4>.*?<h4>(.*?)</h4>.*?<span class="price" id="overlayPrice">¥(.*?)<b>'
53         # car_info_list:
54         # [('宝马', '12万公里', '2004年', '自动/2.5L', '北京', '4.20')]

```

```

55         car_info_list = self.re_func(two_regex, two_html)
56         item = {}
57         item['name'] = car_info_list[0][0].strip()
58         item['km'] = car_info_list[0][1].strip()
59         item['time'] = car_info_list[0][2].strip()
60         item['type'] = car_info_list[0][3].split('/')[0].strip()
61         item['displace'] = car_info_list[0][3].split('/')[1].strip()
62         item['address'] = car_info_list[0][4].strip()
63         item['price'] = car_info_list[0][5].strip()
64
65         print(item)
66         # 数据存入到MongoDB数据库
67         self.myset.insert_one(item)
68
69     def run(self):
70         for page in range(1, 5):
71             page_url = self.url.format(page)
72             self.parse_html(page_url)
73
74 if __name__ == '__main__':
75     spider = CarSpider()
76     spider.run()

```

#### ■ 练习 - 将数据存入MySQL数据库

```

1  create database cardb charset utf8;
2  use cardb;
3  create table cartab(
4  name varchar(100),
5  km varchar(50),
6  years varchar(50),
7  type varchar(50),
8  displacement varchar(50),
9  city varchar(50),
10 price varchar(50)
11 )charset=utf8;

```

#### ■ 使用redis实现增量爬虫

```

1  """
2  汽车之家二手车数据抓取
3  一、分析：
4      1. 一级页面：每辆汽车详情页的链接
5      2. 二级页面：每辆汽车具体的数据
6  二、建立自己的User-Agent池：
7      1. sudo pip3 install fake_useragent
8      2. from fake_useragent import UserAgent
9         UserAgent().random
10 三、使用redis中集合实现增量爬虫
11     原理：根据sadd()的返回值来确定之前是否抓取过
12         返回值为1：说明之前没有抓取过
13         返回值为0：说明之前已经抓取过，程序结束
14 """
15 import requests
16 import re

```

```

17 import time
18 import random
19 from fake_useragent import UserAgent
20 import pymongo
21 import redis
22 from hashlib import md5
23 import sys
24
25 class CarSpider:
26     def __init__(self):
27         self.url = 'https://www.che168.com/beijing/a0_0msdgscncgpillto1csp{}exx0/?
pvareaid=102179#currngpostion'
28         # mongodb3个对象
29         self.conn = pymongo.MongoClient('localhost', 27017)
30         self.db = self.conn['cardb']
31         self.myset = self.db['carset']
32         # 连接到redis
33         self.r = redis.Redis(host='localhost', port=6379, db=0)
34
35     def get_html(self, url):
36         """功能函数1: 请求功能函数"""
37         headers = {'User-Agent': UserAgent().random}
38         # ignore参数: 解码时遇到不识别的字符直接忽略掉
39         html = requests.get(url=url, headers=headers).content.decode('gb2312',
'ignore')
40
41         return html
42
43     def re_func(self, regex, html):
44         """功能函数2: 解析功能函数"""
45         pattern = re.compile(regex, re.S)
46         r_list = pattern.findall(html)
47
48         return r_list
49
50     def md5_url(self, url):
51         """功能函数: 对url进行md5加密"""
52         s = md5()
53         s.update(url.encode())
54
55         return s.hexdigest()
56
57     def parse_html(self, one_url):
58         """爬虫逻辑函数"""
59         one_html = self.get_html(url=one_url)
60         one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
61         # href_list: ['/declear/xxx.html', '', '', '', ...]
62         href_list = self.re_func(one_regex, one_html)
63         for href in href_list:
64             finger = self.md5_url(href)
65             # 返回值1:之前没抓过
66             if self.r.sadd('car:spiders', finger) == 1:
67                 # 拼接完整URL地址,发请求提取具体汽车信息
68                 self.get_one_car_info(href)
69                 time.sleep(random.randint(1, 2))
70             else:
71                 # 一旦发现之前抓过的,则彻底终止程序

```

```

72         sys.exit('更新完成')
73
74     def get_one_car_info(self, href):
75         """提取一辆汽车的具体信息"""
76         two_url = 'https://www.che168.com' + href
77         two_html = self.get_html(url=two_url)
78         two_regex = '<div class="car-box">.*?<h3 class="car-brand-name">(.*?)</h3>.*?<ul class="brand-unit-item fn-clear">.*?<li>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<span class="price" id="overlayPrice">¥(.*?)<b>'
79         # car_info_list:
80         # [('宝马', '12万公里', '2004年', '自动/2.5L', '北京', '4.20')]
81         car_info_list = self.re_func(two_regex, two_html)
82         item = {}
83         item['name'] = car_info_list[0][0].strip()
84         item['km'] = car_info_list[0][1].strip()
85         item['time'] = car_info_list[0][2].strip()
86         item['type'] = car_info_list[0][3].split('/')[0].strip()
87         item['displace'] = car_info_list[0][3].split('/')[1].strip()
88         item['address'] = car_info_list[0][4].strip()
89         item['price'] = car_info_list[0][5].strip()
90
91         print(item)
92         # 数据存入到MongoDB数据库
93         self.myset.insert_one(item)
94
95     def run(self):
96         for page in range(1, 5):
97             page_url = self.url.format(page)
98             self.parse_html(page_url)
99
100 if __name__ == '__main__':
101     spider = CarSpider()
102     spider.run()

```

## 数据持久化 - MySQL

### ■ 瓜子二手车数据存入MySQL数据库

```

1  import requests
2  import re
3  import time
4  import random
5  import pymysql
6
7  class GuaziSpider:
8      def __init__(self):
9          self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
10         self.headers = {

```

```

11         'Cookie': 'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-
e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=9858835725989223197831;
sessionid=5c4c7246-25a1-4a16-8adb-678af786a472; lg=1; lng_lat=116.84757_39.8668;
gps_type=1; close_finance_popup=2020-10-13; cainfo=%7B%22ca_a%22%3A%22-
%22%2C%22ca_b%22%3A%22-
%22%2C%22ca_s%22%3A%22self%22%2C%22ca_n%22%3A%22self%22%2C%22ca_medium%22%3A%22-
%22%2C%22ca_term%22%3A%22-%22%2C%22ca_content%22%3A%22-%22%2C%22ca_campaign%22%3A%22-
%22%2C%22ca_kw%22%3A%22-%22%2C%22ca_i%22%3A%22-%22%2C%22score%22%3A%22-
%22%2C%22keyword%22%3A%22-%22%2C%22ca_keywordid%22%3A%22-
%22%2C%22display_finance_flag%22%3A%22-
%22%2C%22platform%22%3A%221%22%2C%22version%22%3A%221%22%2C%22client_ab%22%3A%22-
%22%2C%22guid%22%3A%221c286513-a2e1-4d4d-e9d5-
231da3e8ee16%22%2C%22ca_city%22%3A%22langfang%22%2C%22sessionid%22%3A%225c4c7246-25a1-
4a16-8adb-678af786a472%22%7D; cityDomain=bj; user_city_id=12;
preTime=%7B%22last%22%3A%221602604364%22%22this%22%3A%221602604337%22%22pre%22%3A%221602604337%7
D',
12         'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
13     }
14     self.db = pymysql.connect('localhost', 'root', '123456', 'guazidb',
charset='utf8')
15     self.cur = self.db.cursor()
16
17     def get_html(self, url):
18         """请求功能函数：获取html"""
19         html = requests.get(url=url, headers=self.headers).content.decode('utf-8',
'ignore')
20
21         return html
22
23     def re_func(self, regex, html):
24         """解析功能函数：正则解析得到列表"""
25         pattern = re.compile(regex, re.S)
26         r_list = pattern.findall(html)
27
28         return r_list
29
30     def parse_html(self, one_url):
31         """爬虫逻辑函数"""
32         one_html = self.get_html(url=one_url)
33         one_regex = '<li data-scroll-track=.*?href="(.*?)"'
34         href_list = self.re_func(regex=one_regex, html=one_html)
35         for href in href_list:
36             two_url = 'https://www.guazi.com' + href
37             # 获取一辆汽车详情页的具体数据
38             self.get_one_car_info(two_url)
39             # 控制数据抓取的频率
40             time.sleep(random.uniform(0, 1))
41
42     def get_one_car_info(self, two_url):
43         """获取一辆汽车的具体数据"""
44         # 名称、行驶里程、排量、变速箱、价格
45         two_html = self.get_html(url=two_url)
46         two_regex = '<div class="product-textbox">.*?<h2 class="titlebox">(.*?)</h2>.*?
<li class="two"><span>(.*?)</span>.*?<li class="three"><span>(.*?)</span>.*?<li
class="last"><span>(.*?)</span>.*?<span class="price-num">(.*?)</span>'
47         car_info_list = self.re_func(regex=two_regex, html=two_html)

```

```

48         # 获取具体数据
49         item = {}
50         item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
51         item['km'] = car_info_list[0][1].strip()
52         item['displace'] = car_info_list[0][2].strip()
53         item['type'] = car_info_list[0][3].strip()
54         item['price'] = car_info_list[0][4].strip()
55         print(item)
56
57         li = [item['name'], item['km'], item['displace'], item['type'], item['price']]
58         ins = 'insert into guazitab values(%s,%s,%s,%s,%s)'
59         self.cur.execute(ins, li)
60         self.db.commit()
61
62
63     def run(self):
64         for o in range(1, 3):
65             one_url = self.url.format(o)
66             self.parse_html(one_url=one_url)
67         # 断开数据库
68         self.cur.close()
69         self.db.close()
70
71 if __name__ == '__main__':
72     spider = GuaziSpider()
73     spider.run()

```

## 数据持久化 - csv

### ■ csv描述

```

1  【1】作用
2      将爬取的数据存放到本地的csv文件中
3
4  【2】使用流程
5      2.1> 打开csv文件
6      2.2> 初始化写入对象
7      2.3> 写入数据(参数为列表)
8
9  【3】示例代码
10     import csv
11     with open('sky.csv','w') as f:
12         writer = csv.writer(f)
13         writer.writerow([])

```

### ■ 示例

```

1  【1】 题目描述
2      创建 test.csv 文件，在文件中写入数据
3
4  【2】 数据写入 - writerow([])方法
5      import csv
6      with open('test.csv','w') as f:
7          writer = csv.writer(f)
8          writer.writerow(['超哥哥','25'])

```

#### ■ 练习 - 使用 writerow() 方法将瓜子二手车数据存入本地 guazi.csv 文件

```

1  【1】 在 __init__() 中打开csv文件，因为csv文件只需要打开和关闭1次即可
2  【2】 在 save_html() 中将所抓取的数据处理成列表，使用writerow()方法写入
3  【3】 在run() 中等数据抓取完成后关闭文件

```

#### ■ 代码实现

```

1  import requests
2  import re
3  import time
4  import random
5  import csv
6
7  class GuaziSpider:
8      def __init__(self):
9          self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
10         self.headers = {
11             'Cookie': 'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=9858835725989223197831; sessionid=5c4c7246-25a1-4a16-8adb-678af786a472; lg=1; lng_lat=116.84757_39.8668; gps_type=1; close_finance_popup=2020-10-13; cainfo=%7B%22ca_a%22%3A%22-%22%2C%22ca_b%22%3A%22-%22%2C%22ca_s%22%3A%22self%22%2C%22ca_n%22%3A%22self%22%2C%22ca_medium%22%3A%22-%22%2C%22ca_term%22%3A%22-%22%2C%22ca_content%22%3A%22-%22%2C%22ca_campaign%22%3A%22-%22%2C%22ca_kw%22%3A%22-%22%2C%22ca_i%22%3A%22-%22%2C%22scode%22%3A%22-%22%2C%22keyword%22%3A%22-%22%2C%22ca_keywordid%22%3A%22-%22%2C%22display_finance_flag%22%3A%22-%22%2C%22platform%22%3A%221%22%2C%22version%22%3A%221%22%2C%22client_ab%22%3A%22-%22%2C%22guid%22%3A%221c286513-a2e1-4d4d-e9d5-231da3e8ee16%22%2C%22ca_city%22%3A%22langfang%22%2C%22sessionid%22%3A%225c4c7246-25a1-4a16-8adb-678af786a472%22%7D; cityDomain=bj; user_city_id=12; preTime=%7B%22last%22%3A%221602604364%22%22this%22%3A%221602604337%22%22pre%22%3A%221602604337%22%7D',
12             'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
13         }
14         # 打开文件、创建csv写入对象
15         self.f = open('guazi.csv', 'w', newline='')
16         self.writer = csv.writer(self.f)
17
18         def get_html(self, url):
19             """请求功能函数：获取html"""
20             html = requests.get(url=url, headers=self.headers).content.decode('utf-8', 'ignore')
21

```

```

22         return html
23
24     def re_func(self, regex, html):
25         """解析功能函数：正则解析得到列表"""
26         pattern = re.compile(regex, re.S)
27         r_list = pattern.findall(html)
28
29         return r_list
30
31     def parse_html(self, one_url):
32         """爬虫逻辑函数"""
33         one_html = self.get_html(url=one_url)
34         one_regex = '<li data-scroll-track=.*?href="(.*?)>'
35         href_list = self.re_func(regex=one_regex, html=one_html)
36         for href in href_list:
37             two_url = 'https://www.guazi.com' + href
38             # 获取一辆汽车详情页的具体数据
39             self.get_one_car_info(two_url)
40             # 控制数据抓取的频率
41             time.sleep(random.uniform(0, 1))
42
43     def get_one_car_info(self, two_url):
44         """获取一辆汽车的具体数据"""
45         # 名称、行驶里程、排量、变速箱、价格
46         two_html = self.get_html(url=two_url)
47         two_regex = '<div class="product-textbox">.*?<h2 class="titlebox">(.*?)</h2>.*?<li class="two"><span>(.*?)</span>.*?<li class="three"><span>(.*?)</span>.*?<li class="last"><span>(.*?)</span>.*?<span class="price-num">(.*?)</span>'
48         car_info_list = self.re_func(regex=two_regex, html=two_html)
49         # 获取具体数据
50         item = {}
51         item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
52         item['km'] = car_info_list[0][1].strip()
53         item['displace'] = car_info_list[0][2].strip()
54         item['type'] = car_info_list[0][3].strip()
55         item['price'] = car_info_list[0][4].strip()
56         print(item)
57
58         # 将数据处理成列表,并存入csv文件
59         li = [item['name'], item['km'], item['displace'], item['type'], item['price']]
60         self.writer.writerow(li)
61
62     def run(self):
63         for o in range(1, 2):
64             one_url = self.url.format(o)
65             self.parse_html(one_url=one_url)
66
67         # 关闭文件
68         self.f.close()
69
70 if __name__ == '__main__':
71     spider = GuaziSpider()
72     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> 网上下载插件 - 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
```

## ■ 匹配演示 - 猫眼电影top100

```
1  【1】查找所有的dd节点
2      //dd
3  【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
4      //p[@class="name"]/a
5  【3】获取d1节点下第2个dd节点的电影节点
6      //d1[@class="board-wrapper"]/dd[2]
7  【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
8      //p[@class="name"]/a/@href
9
10  【注意】
11     1> 只要涉及到条件,加 [] : //d1[@class="xxx"]    //d1/dd[2]
12     2> 只要获取属性值,加 @ : //d1[@class="xxx"]    //p/a/@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/@title
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】text() : 获取节点的文本内容
2      xpath表达式末尾不加 /text() : 则得到的结果为节点对象
3      xpath表达式末尾加 /text() 或者 /@href : 则得到结果为字符串
4
5  【2】contains() : 匹配属性值中包含某些字符串节点
6      匹配class属性值中包含 'movie-item' 这个字符串的 div 节点
7      //div[contains(@class,"movie-item")]

```

## ■ 终极总结

```

1  【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
3  【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'
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()

```

# 作业

---

- 1   **【1】正则抓取豆瓣图书top250书籍信息**
- 2       地址: <https://book.douban.com/top250?icn=index-book250-all>
- 3       抓取目标: 书籍名称、书籍信息、书籍评分、书籍评论人数、书籍描述
- 4
- 5   **【2】使用xpath helper在页面中匹配豆瓣图书top250的信息, 写出对应的xpath表达式**
- 6       书籍名称:
- 7       书籍信息:
- 8       书籍评分:
- 9       评论人数:
- 10      书籍描述: