Day02回顾

请求模块(requests)

```
html = requests.get(url=url,headers=headers).text
html = requests.get(url=url,headers=headers).content.decode('utf-8')

with open('xxx.txt','w',encoding='utf-8') as f:
    f.write(html)
```

编码模块(urllib.parse)

```
1、urlencode({dict})
   urlencode({'wd':'美女','pn':'20'})
   编码后 : 'wd=%E8%D5XXX&pn=20'

2、quote(string)
   quote('织女')
   编码后 : '%D3%F5XXX'

3、unquote('%D3%F5XXX')
```

解析模块(re)

- **使用流程**

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

- **贪婪匹配和非贪婪匹配**

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

- **正则表达式分组**

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

抓取步骤

```
【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】数据存在: 查看URL地址规律
【3】写正则表达式,来匹配数据
【4】程序结构
    a>每爬取1个页面后随机休眠一段时间
```

```python
# 程序结构
class xxxSpider(object):
    def __init__(self):
        # 定义常用变量,url,headers及计数等

    def get_html(self):
        # 获取响应内容函数,使用随机User-Agent

    def parse_html(self):
        # 使用正则表达式来解析页面，提取数据

    def save_html(self):
        # 将提取的数据按要求保存，csv、MySQL数据库等

    def run(self):
        # 程序入口函数，用来控制整体逻辑

if __name__ == '__main__':
    # 程序开始运行时间戳
    start = time.time()
    spider = xxxSpider()
    spider.run()
    # 程序运行结束时间戳
    end = time.time()
    print('执行时间:%.2f' % (end-start))
```

数据持久化-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()
```

spider-day03笔记

瓜子二手车数据抓取 - 二级页面

- **领取任务**

```
【1】爬取地址
    瓜子网 - 我要买车
    https://www.guazi.com/bj/buy/

【2】爬取目标
    所有汽车的 汽车名称、行驶里程、排量、变速箱、价格

【3】爬取分析
    *********一级页面需抓取***********
        1、车辆详情页的链接

    *********二级页面需抓取***********
        1、汽车名称
        2、行驶里程
        3、排量
        4、变速箱
        5、价格
```

- **实现步骤**

```
【1】确定响应内容中是否存在所需抓取数据 - 存在

【2】找URL地址规律
    第1页: https://www.guazi.com/bj/buy/o1/#bread
    第2页: https://www.guazi.com/bj/buy/o2/#bread
    第n页: https://www.guazi.com/bj/buy/o{}/#bread

【3】 写正则表达式
    一级页面正则表达式:<li data-scroll-track=.*?href="(.*?)"

    二级页面正则表达式:<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>

【4】代码实现
```

- **代码实现**

```
import requests
import re
import time
import random

class GuaziSpider:
    def __init__(self):
        self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
        self.headers = {
            'Cookie':'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=985883572598922319783l; 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%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',
            'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
        }

    def get_html(self, url):
        """请求功能函数: 获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('utf-8', 'ignore')

        return html

    def re_func(self, regex, html):
```

```
        """解析功能函数：正则解析得到列表"""
        pattern = re.compile(regex, re.S)
        r_list = pattern.findall(html)

        return r_list

    def parse_html(self, one_url):
        """爬虫逻辑函数"""
        one_html = self.get_html(url=one_url)
        one_regex = '<li data-scroll-track=.*?href="(.*?)"'
        href_list = self.re_func(regex=one_regex, html=one_html)
        for href in href_list:
            two_url = 'https://www.guazi.com' + href
            # 获取一辆汽车详情页的具体数据
            self.get_one_car_info(two_url)
            # 控制数据抓取的频率
            time.sleep(random.uniform(0, 1))

    def get_one_car_info(self, two_url):
        """获取一辆汽车的具体数据"""
        # 名称、行驶里程、排量、变速箱、价格
        two_html = self.get_html(url=two_url)
        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>'
        car_info_list = self.re_func(regex=two_regex, html=two_html)
        # 获取具体数据
        item = {}
        item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
        item['km'] = car_info_list[0][1].strip()
        item['displace'] = car_info_list[0][2].strip()
        item['type'] = car_info_list[0][3].strip()
        item['price'] = car_info_list[0][4].strip()
        print(item)

    def run(self):
        for o in range(1, 6):
            one_url = self.url.format(o)
            self.parse_html(one_url=one_url)

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

- **练习 - 将数据存入MySQL数据库**

```
create database guazidb charset utf8;
use guazidb;
create table guazitab(
name varchar(200),
km varchar(100),
displace varchar(100),
type varchar(100),
price varchar(100)
)charset=utf8;
```

- **使用redis实现增量爬虫-redis集合实现**

```
import requests
import re
import time
import random
import pymysql
import redis
import sys
from hashlib import md5


class GuaziSpider:
    def __init__(self):
        self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
        self.headers = {
            'Cookie':'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=985883572598922319783l; 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%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',
            'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
        }
        # 连接MySQL
        self.db = pymysql.connect('localhost', 'root', '123456', 'guazidb', charset='utf8')
        self.cur = self.db.cursor()
        # 连接redis
        self.r = redis.Redis(host='localhost', port=6379, db=0)

    def get_html(self, url):
        """请求功能函数：获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('utf-8', 'ignore')

        return html

    def re_func(self, regex, html):
        """解析功能函数：正则解析得到列表"""
        pattern = re.compile(regex, re.S)
        r_list = pattern.findall(html)

        return r_list

```

```python
    def md5_url(self, url):
        """功能函数3：md5加密生成指纹"""
        s = md5()
        s.update(url.encode())

        return s.hexdigest()

    def parse_html(self, one_url):
        """爬虫逻辑函数"""
        one_html = self.get_html(url=one_url)
        one_regex = '<li data-scroll-track=.*?href="(.*?)"'
        href_list = self.re_func(regex=one_regex, html=one_html)
        for href in href_list:
            two_url = 'https://www.guazi.com' + href
            # 生成指纹
            finger = self.md5_url(url=two_url)
            # 如果添加成功说明之前未抓取过
            if self.r.sadd('guazi:spider', finger) == 1:
                # 获取一辆汽车详情页的具体数据
                self.get_one_car_info(two_url)
                # 控制数据抓取的频率
                time.sleep(random.uniform(0, 1))
            else:
                # 否则结束程序
                sys.exit('更新完成')

    def get_one_car_info(self, two_url):
        """获取一辆汽车的具体数据"""
        # 名称、行驶里程、排量、变速箱、价格
        two_html = self.get_html(url=two_url)
        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>'
        car_info_list = self.re_func(regex=two_regex, html=two_html)
        # 获取具体数据
        item = {}
        item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
        item['km'] = car_info_list[0][1].strip()
        item['displace'] = car_info_list[0][2].strip()
        item['type'] = car_info_list[0][3].strip()
        item['price'] = car_info_list[0][4].strip()
        print(item)

        li = [item['name'], item['km'], item['displace'], item['type'], item['price']]
        ins = 'insert into guazitab values(%s,%s,%s,%s,%s)'
        self.cur.execute(ins, li)
        self.db.commit()


    def run(self):
        for o in range(1, 2):
            one_url = self.url.format(o)
            self.parse_html(one_url=one_url)
        # 断开数据库
        self.cur.close()
        self.db.close()

```

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

汽车之家数据抓取 - 二级页面

- **领取任务**

```
【1】爬取地址
    汽车之家 - 二手车 - 价格从低到高
    https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp1exx0/


【2】爬取目标
    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

【3】爬取分析
    *********一级页面需抓取***********
        1、车辆详情页的链接

    *********二级页面需抓取***********
        1、名称
        2、行驶里程
        3、上牌时间
        4、档位
        5、排量
        6、车辆所在地
        7、价格
```

- **实现步骤**

```
【1】确定响应内容中是否存在所需抓取数据 - 存在

【2】找URL地址规律
    第1页: https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp1exx0/
    第2页: https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp2exx0/
    第n页: https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp{}exx0/

【3】 写正则表达式
    一级页面正则表达式:<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>
    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>

【4】代码实现
<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**

```
"""
```

```
汽车之家二手车数据抓取
分析:
    1. 一级页面: 每辆汽车详情页的链接
    2. 二级页面: 每辆汽车具体的数据
"""
import requests
import re
import time
import random
from fake_useragent import UserAgent
import pymongo

class CarSpider:
    def __init__(self):
        self.url = 'https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp{}exx0/?pvareaid=102179#currengpostion'
        # 3个对象
        self.conn = pymongo.MongoClient('localhost', 27017)
        self.db = self.conn['cardb']
        self.myset = self.db['carset']

    def get_html(self, url):
        """功能函数1: 请求功能函数"""
        headers = {'User-Agent': UserAgent().random}
        # ignore参数: 解码时遇到不识别的字符直接忽略掉
        html = requests.get(url=url, headers=headers).content.decode('gb2312', 'ignore')

        return html

    def re_func(self, regex, html):
        """功能函数2: 解析功能函数"""
        pattern = re.compile(regex, re.S)
        r_list = pattern.findall(html)

        return r_list

    def parse_html(self, one_url):
        """爬虫逻辑函数"""
        one_html = self.get_html(url=one_url)
        one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
        # href_list: ['/declear/xxx.html', '', '', '', ...]
        href_list = self.re_func(one_regex, one_html)
        for href in href_list:
            # 拼接完整URL地址,发请求提取具体汽车信息
            self.get_one_car_info(href)
            time.sleep(random.randint(1, 2))

    def get_one_car_info(self, href):
        """提取一辆汽车的具体信息"""
        two_url = 'https://www.che168.com' + href
        two_html = self.get_html(url=two_url)
        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>'
        # car_info_list:
        # [('宝马','12万公里','2004年','自动/2.5L','北京','4.20')]
```

```
        car_info_list = self.re_func(two_regex, two_html)
        item = {}
        item['name'] = car_info_list[0][0].strip()
        item['km'] = car_info_list[0][1].strip()
        item['time'] = car_info_list[0][2].strip()
        item['type'] = car_info_list[0][3].split('/')[0].strip()
        item['displace'] = car_info_list[0][3].split('/')[1].strip()
        item['address'] = car_info_list[0][4].strip()
        item['price'] = car_info_list[0][5].strip()

        print(item)
        # 数据存入到MongoDB数据库
        self.myset.insert_one(item)

    def run(self):
        for page in range(1, 5):
            page_url = self.url.format(page)
            self.parse_html(page_url)

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

- **练习 - 将数据存入MySQL数据库**

```
create database cardb charset utf8;
use cardb;
create table cartab(
name varchar(100),
km varchar(50),
years varchar(50),
type varchar(50),
displacement varchar(50),
city varchar(50),
price varchar(50)
)charset=utf8;
```

- **使用redis实现增量爬虫**

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

```python
import time
import random
from fake_useragent import UserAgent
import pymongo
import redis
from hashlib import md5
import sys

class CarSpider:
    def __init__(self):
        self.url = 'https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp{}exx0/?pvareaid=102179#currengpostion'
        # mongodb3个对象
        self.conn = pymongo.MongoClient('localhost', 27017)
        self.db = self.conn['cardb']
        self.myset = self.db['carset']
        # 连接到redis
        self.r = redis.Redis(host='localhost', port=6379, db=0)

    def get_html(self, url):
        """功能函数1: 请求功能函数"""
        headers = {'User-Agent': UserAgent().random}
        # ignore参数: 解码时遇到不识别的字符直接忽略掉
        html = requests.get(url=url, headers=headers).content.decode('gb2312', 'ignore')

        return html

    def re_func(self, regex, html):
        """功能函数2: 解析功能函数"""
        pattern = re.compile(regex, re.S)
        r_list = pattern.findall(html)

        return r_list

    def md5_url(self, url):
        """功能函数: 对url进行md5加密"""
        s = md5()
        s.update(url.encode())

        return s.hexdigest()

    def parse_html(self, one_url):
        """爬虫逻辑函数"""
        one_html = self.get_html(url=one_url)
        one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
        # href_list: ['/declear/xxx.html', '', '', '', ...]
        href_list = self.re_func(one_regex, one_html)
        for href in href_list:
            finger = self.md5_url(href)
            # 返回值1:之前没抓过
            if self.r.sadd('car:spiders', finger) == 1:
                # 拼接完整URL地址,发请求提取具体汽车信息
                self.get_one_car_info(href)
                time.sleep(random.randint(1, 2))
            else:
                # 一旦发现之前抓过的,则彻底终止程序
```

```
                sys.exit('更新完成')

    def get_one_car_info(self, href):
        """提取一辆汽车的具体信息"""
        two_url = 'https://www.che168.com' + href
        two_html = self.get_html(url=two_url)
        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>'
        # car_info_list:
        # [('宝马','12万公里','2004年','自动/2.5L','北京','4.20')]
        car_info_list = self.re_func(two_regex, two_html)
        item = {}
        item['name'] = car_info_list[0][0].strip()
        item['km'] = car_info_list[0][1].strip()
        item['time'] = car_info_list[0][2].strip()
        item['type'] = car_info_list[0][3].split('/')[0].strip()
        item['displace'] = car_info_list[0][3].split('/')[1].strip()
        item['address'] = car_info_list[0][4].strip()
        item['price'] = car_info_list[0][5].strip()

        print(item)
        # 数据存入到MongoDB数据库
        self.myset.insert_one(item)

    def run(self):
        for page in range(1, 5):
            page_url = self.url.format(page)
            self.parse_html(page_url)

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

数据持久化 - MySQL

- **瓜子二手车数据存入MySQL数据库**

```
import requests
import re
import time
import random
import pymysql

class GuaziSpider:
    def __init__(self):
        self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
        self.headers = {
```

```
            'Cookie':'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=985883572598922319783 1; 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%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',
            'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
        }
        self.db = pymysql.connect('localhost', 'root', '123456', 'guazidb', charset='utf8')
        self.cur = self.db.cursor()

    def get_html(self, url):
        """请求功能函数：获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('utf-8', 'ignore')

        return html

    def re_func(self, regex, html):
        """解析功能函数：正则解析得到列表"""
        pattern = re.compile(regex, re.S)
        r_list = pattern.findall(html)

        return r_list

    def parse_html(self, one_url):
        """爬虫逻辑函数"""
        one_html = self.get_html(url=one_url)
        one_regex = '<li data-scroll-track=.*?href="(.*?)"'
        href_list = self.re_func(regex=one_regex, html=one_html)
        for href in href_list:
            two_url = 'https://www.guazi.com' + href
            # 获取一辆汽车详情页的具体数据
            self.get_one_car_info(two_url)
            # 控制数据抓取的频率
            time.sleep(random.uniform(0, 1))

    def get_one_car_info(self, two_url):
        """获取一辆汽车的具体数据"""
        # 名称、行驶里程、排量、变速箱、价格
        two_html = self.get_html(url=two_url)
        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>'
        car_info_list = self.re_func(regex=two_regex, html=two_html)
```

```
        # 获取具体数据
        item = {}
        item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
        item['km'] = car_info_list[0][1].strip()
        item['displace'] = car_info_list[0][2].strip()
        item['type'] = car_info_list[0][3].strip()
        item['price'] = car_info_list[0][4].strip()
        print(item)

        li = [item['name'], item['km'], item['displace'], item['type'], item['price']]
        ins = 'insert into guazitab values(%s,%s,%s,%s,%s)'
        self.cur.execute(ins, li)
        self.db.commit()


    def run(self):
        for o in range(1, 3):
            one_url = self.url.format(o)
            self.parse_html(one_url=one_url)
        # 断开数据库
        self.cur.close()
        self.db.close()

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

数据持久化 - csv

- **csv描述**

```
【1】作用
   将爬取的数据存放到本地的csv文件中

【2】使用流程
    2.1> 打开csv文件
    2.2> 初始化写入对象
    2.3> 写入数据(参数为列表)

【3】示例代码
    import csv
    with open('sky.csv','w') as f:
        writer = csv.writer(f)
        writer.writerow([])
```

- **示例**

```
【1】题目描述
   创建 test.csv 文件，在文件中写入数据

【2】数据写入 - writerow([])方法
   import csv
   with open('test.csv','w') as f:
       writer = csv.writer(f)
       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】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】在 save_html() 中将所抓取的数据处理成列表，使用writerow()方法写入
【3】在run() 中等数据抓取完成后关闭文件
```

- **代码实现**

```
import requests
import re
import time
import random
import csv

class GuaziSpider:
    def __init__(self):
        self.url = 'https://www.guazi.com/bj/buy/o{}/#bread'
        self.headers = {
            'Cookie':'antipas=B643vU290N4423L56048105H5340; uuid=1c286513-a2e1-4d4d-e9d5-231da3e8ee16; clueSourceCode=%2A%2300; ganji_uuid=985883572598922319783l; 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%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',
            'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36',
        }
        # 打开文件、创建csv写入对象
        self.f = open('guazi.csv', 'w', newline='')
        self.writer = csv.writer(self.f)

    def get_html(self, url):
        """请求功能函数：获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('utf-8', 'ignore')

```

```
        return html

    def re_func(self, regex, html):
        """解析功能函数：正则解析得到列表"""
        pattern = re.compile(regex, re.S)
        r_list = pattern.findall(html)

        return r_list

    def parse_html(self, one_url):
        """爬虫逻辑函数"""
        one_html = self.get_html(url=one_url)
        one_regex = '<li data-scroll-track=.*?href="(.*?)"'
        href_list = self.re_func(regex=one_regex, html=one_html)
        for href in href_list:
            two_url = 'https://www.guazi.com' + href
            # 获取一辆汽车详情页的具体数据
            self.get_one_car_info(two_url)
            # 控制数据抓取的频率
            time.sleep(random.uniform(0, 1))

    def get_one_car_info(self, two_url):
        """获取一辆汽车的具体数据"""
        # 名称、行驶里程、排量、变速箱、价格
        two_html = self.get_html(url=two_url)
        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>'
        car_info_list = self.re_func(regex=two_regex, html=two_html)
        # 获取具体数据
        item = {}
        item['name'] = car_info_list[0][0].strip().split('\r\n')[0].strip()
        item['km'] = car_info_list[0][1].strip()
        item['displace'] = car_info_list[0][2].strip()
        item['type'] = car_info_list[0][3].strip()
        item['price'] = car_info_list[0][4].strip()
        print(item)

        # 将数据处理成列表,并存入csv文件
        li = [item['name'], item['km'], item['displace'], item['type'], item['price']]
        self.writer.writerow(li)

    def run(self):
        for o in range(1, 2):
            one_url = self.url.format(o)
            self.parse_html(one_url=one_url)

        # 关闭文件
        self.f.close()

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

Chrome浏览器安装插件

- **安装方法**

```
【1】在线安装
    1.1> 下载插件 - google访问助手
    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)
    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

【2】离线安装
    2.1> 网上下载插件 - xxx.crx 重命名为 xxx.zip
    2.2> Chrome浏览器-设置-更多工具-扩展程序-开发者模式
    2.3> 拖拽 插件(或者解压后文件夹) 到浏览器中
    2.4> 重启浏览器，使插件生效
```

- **爬虫常用插件**

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

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()
```

作业

```
【1】正则抓取豆瓣图书top250书籍信息
    地址：https://book.douban.com/top250?icn=index-book250-all
    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

【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
    书籍名称：
    书籍信息：
    书籍评分：
    评论人数：
    书籍描述：
```