

Day01回顾

请求模块(urllib.request)

```
1 req = request.Request(url,headers=headers)
2 res = request.urlopen(req)
3 html = res.read().decode('utf-8','ignore')
```

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 p = re.compile('正则表达式',re.S)
2 r_list = p.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。结果: [(,),(),(),(),()]
```

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
- 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        # 主函数,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.main()
23     # 程序运行结束时间戳
24     end = time.time()
25     print('执行时间:%.2f' % (end-start))
```

spider-day02笔记

猫眼电影top100抓取案例

爬虫需求

```
1 # 1. 确定URL地址
2 百度搜索 - 猫眼电影 - 榜单 - top100榜
3
4 # 2. 爬取目标
5 电影名称、主演、上映时间
```

爬虫实现

```

1 # 1. 查看网页源码, 确认数据来源
2 响应内容中存在所需抓取数据 - 电影名称、主演、上映时间
3
4 # 2. 翻页寻找URL地址规律
5 第1页: https://maoyan.com/board/4?offset=0
6 第2页: https://maoyan.com/board/4?offset=10
7 第n页: offset=(n-1)*10
8
9 # 3. 编写正则表达式
10 <div class="movie-item-info">.*?title="(.*?)".*?class="star">(.*?)</p>.*?releasetime">(.*?)
    </p>

```

■ 代码实现 - 初始代码

```

1 from urllib import request
2 import re
3 import time
4 import random
5
6
7 class MaoyanSpider(object):
8     def __init__(self):
9         self.url = 'https://maoyan.com/board/4?offset={}'
10        self.i = 0
11
12        def get_html(self,url):
13            headers = {
14                'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML,
15                like Gecko) Chrome/14.0.835.163 Safari/535.1',
16                'Cookie': '__mta=219092646.1582029073016.1582029104477.1582029107258.4;
17                uuid_n_v=v1; uuid=86F5E1A0524A11EA968A372EE6B0620493363A49A0D9439A84A14933EDB51143;
18                _csrf=5029ad27fba276f31795f6351789d9c6752f33517e8a61236b65d2eca7885032;
19                Hm_lvt_703e94591e87be68cc8da0da7cbd0be2=1582029064;
20                _lx_utm=utm_source%3DBaidu%26utm_medium%3Dorganic; _lxsdk_cuid=1705847c8c1c8-
21                00072237f76da3-3f6b4d00-100200-1705847c8c1c8;
22                _lxsdk=86F5E1A0524A11EA968A372EE6B0620493363A49A0D9439A84A14933EDB51143; mojo-
23                uuid=c8bce593970a7f39b7675f826ca3103b; mojo-session-id=
24                {"id":"62ea6d2e0b68f4a245fee15d84e88693","time":1582029073083};
25                __mta=219092646.1582029073016.1582029102826.1582029104477.3; mojo-trace-id=6;
26                Hm_lpv7_703e94591e87be68cc8da0da7cbd0be2=1582029107; _lxsdk_s=1705847c8c4-e3c-f08-
27                93d%7C%7C10'
28            }
29            req = request.Request(url=url,headers=headers)
30            resp= request.urlopen(req)
31            html= resp.read().decode()
32            # 直接调用解析函数
33            self.parse_html(html)
34
35        def parse_html(self,html):
36            """正则解析函数"""
37            regex = '<div class="movie-item-info">.*?title="(.*?)".*?class="star">(.*?)</p>.*?
38            releasetime">(.*?)</p>'
39            pattern = re.compile(regex,re.S)
40            # dd_list: [(),(),())
41            dd_list = pattern.findall(html)

```

```

29         self.save_html(dd_list)
30
31     def save_html(self, dd_list):
32         """保存数据函数"""
33         item = {}
34         for dd in dd_list:
35             item['name'] = dd[0].strip()
36             item['star'] = dd[1].strip()[3:]
37             item['time'] = dd[2].strip()[5:15]
38             print(item)
39             self.i += 1
40
41     def run(self):
42         for offset in range(0, 91, 10):
43             url = self.url.format(offset)
44             self.get_html(url)
45             time.sleep(random.randint(1, 3))
46             print('电影数量: ', self.i)
47
48 if __name__ == '__main__':
49     start_time = time.time()
50     spider = MaoyanSpider()
51     spider.run()
52     end_time = time.time()
53     print('执行时间: %.2f' % (end_time - start_time))

```

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】单行写入 - writerow([])方法
5      import csv
6      with open('test.csv', 'w') as f:

```

```

7     writer = csv.writer(f)
8     writer.writerow(['步惊云', '36'])
9     writer.writerow(['超哥哥', '25'])
10
11 【3】多行写入 - writerows([(),(),()]) 方法
12     import csv
13     with open('test.csv','w') as f:
14         writer = csv.writer(f)
15         writer.writerows([('聂风', '36'),('秦霜', '25'),('孔慈', '30')])

```

■ 练习 - 猫眼电影数据存入本地 maoyanfilm.csv 文件

```

1 def save_html(self,r_list):
2     with open('maoyanfilm.csv','a') as f:
3         writer = csv.writer(f)
4         for rt in r_list:
5             one_film_list = [rt[0].strip(),rt[1].strip(),rt[2].strip()]
6             writer.writerow(one_film_list)

```

数据持久化 - MySQL

■ pymysql回顾

```

1 # 1. 单条插入表记录 - excute()方法
2 # 2. 多条插入表记录 - executemany()方法
3
4 # 示例代码如下:
5 import pymysql
6
7 db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')
8 cursor = db.cursor()
9
10 ins = 'insert into filmtab values(%s,%s,%s)'
11 # 1. 单条插入表记录之 excute() 方法
12 cursor.execute(ins,['霸王别姬','张国荣','1993'])
13 # 2. 多条插入表记录之 executemany() 方法 - 高效且节省资源
14 cursor.executemany(ins,[('大话1','周','1993'),('大话2','周','1994')])
15
16 db.commit()
17 cursor.close()
18 db.close()

```

■ 练习 - 将电影信息存入MySQL数据库

```

1 【1】提前建库建表
2 mysql -h127.0.0.1 -uroot -p123456
3 create database maoyandb charset utf8;
4 use maoyandb;
5 create table maoyantab(
6     name varchar(100),
7     star varchar(300),
8     time varchar(100)

```

```

9 )charset=utf8;
10
11 【2】 使用execute()方法将数据存入数据库 - 在初始代码基础上做如下改动
12 import pymysql
13
14 def __init__(self):
15     # 添加代码
16     self.db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')
17     self.cursor = self.db.cursor()
18
19 def save_html(self,dd_list):
20     # 覆盖原来代码
21     ins = 'insert into maoyantab values(%s,%s,%s)'
22     for dd in dd_list:
23         # 将每个电影信息处理成列表
24         dd_li = [dd[0].strip(),dd[1].strip(),dd[2].strip()]
25         self.cursor.execute(ins,dd_li)
26         self.db.commit()
27         print(dd_li)
28         self.i += 1
29
30 def run(self):
31     # 添加代码
32     self.cursor.close()
33     self.db.close()
34
35 【3】 使用executemany()方法将数据存入数据库 - 在初始代码基础上做如下改动
36 import pymysql
37
38 def __init__(self):
39     # 添加代码
40     self.db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')
41     self.cursor = self.db.cursor()
42     # 存放所有电影信息的大列表
43     self.all_film_list = []
44
45 def save_html(self,dd_list):
46     # 覆盖原来代码
47     for dd in dd_list:
48         dd_tuple = (dd[0].strip(),dd[1].strip(),dd[2].strip())
49         self.all_film_list.append(dd_tuple)
50         self.i += 1
51
52 def run(self):
53     # 添加代码
54     ins = 'insert into maoyantab values(%s,%s,%s)'
55     self.cursor.executemany(ins,self.all_film_list)
56     self.db.commit()
57     self.cursor.close()
58     self.db.close()

```

■ 4、练习 - SQL查询

```

1 1、查询20年以前的电影的名字和上映时间
2   select name,time from film where time<=(now()-interval 20 year);
3 2、查询1990-2000年的电影名字和上映时间
4   select name,time from film where time>='1990-01-01' and time<='2000-12-31';

```

数据持久化 - MongoDB

■ MongoDB特点

```

1 1、非关系型数据库,数据以键值对方式存储
2 2、MongoDB基于磁盘存储
3 3、MongoDB数据类型单一,值为JSON文档,而Redis基于内存,数据类型丰富
4 4、MongoDB: 库 -> 集合 -> 文档
5   MySQL  : 库 -> 表 -> 表记录

```

■ MongoDB常用命令

```

1 Linux进入: mongo
2 >show dbs           - 查看所有库
3 >use 库名           - 切换库
4 >show collections   - 查看当前库中所有集合
5 >db.集合名.find().pretty() - 查看集合中文档
6 >db.集合名.count()   - 统计文档条数
7 >db.集合名.drop()    - 删除集合
8 >db.dropDatabase()  - 删除当前库

```

■ pymongo回顾

```

1 import pymongo
2
3 # 1.连接对象
4 conn = pymongo.MongoClient(host = 'localhost',port = 27017)
5 # 2.库对象
6 db = conn['maoyandb']
7 # 3.集合对象
8 myset = db['maoyanset']
9 # 4.插入数据库
10 myset.insert_one({'name':'赵敏'})
11 myset.insert_many([{'name':'小昭'},{'age':'30'}])

```

■ 练习 - 将电影信息存入MongoDB数据库

```

1 """在初始代码基础上做如下更改"""
2 import pymongo
3
4 def __init__(self):
5     # 添加
6     self.conn = pymongo.MongoClient('localhost',27017)
7     self.db = self.conn['maoyandb']
8     self.myset = self.db['maoyanset']

```

```

9
10 def save_html(self,r_list):
11     # 覆盖
12     # 将数据处理为字典,执行insert_one()
13     item = {}
14     for r in r_list:
15         item['name'] = r[0].strip()
16         item['star'] = r[1].strip()
17         item['time'] = r[2].strip()
18         self.myset.insert_one(item)

```

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_0msdgsncncgpi1lto1csp1exx0/
4
5
6 # 2、爬取目标
7 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、车辆详情页的连接
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_0msdgsncncgpi1lto1csp1exx0/
5 第2页: https://www.che168.com/beijing/a0_0msdgsncncgpi1lto1csp2exx0/
6 第n页: https://www.che168.com/beijing/a0_0msdgsncncgpi1lto1csp{ }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

```


■ 代码实现

```
1 from urllib import request
2 import re
3 import time
4 import random
5
6
7 class CarSpider(object):
8     def __init__(self):
9         self.url = 'https://www.che168.com/beijing/a0_0msdgsncgpi1to1csp{}exx0/'
10        self.headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1
11        (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}
12
13        # 功能函数1 - 获取响应内容
14        def get_html(self, url):
15            req = request.Request(url=url, headers=self.headers)
16            try:
17                res = request.urlopen(req, timeout=3)
18                html = res.read().decode('gb2312', 'ignore')
19                return html
20            except Exception as e:
21                print(e)
22
23        # 功能函数2 - 正则解析
24        def re_func(self, regex, html):
25            pattern = re.compile(regex, re.S)
26            r_list = pattern.findall(html)
27
28            return r_list
29
30        # 爬虫函数开始
31        def parse_html(self, one_url):
32            one_html = self.get_html(one_url)
33            one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
34            href_list = self.re_func(one_regex, one_html)
35            for href in href_list:
36                # 每便利一个汽车信息，必须要把此辆汽车所有数据提取完成后再提取下一辆汽车信息
37                url = 'https://www.che168.com' + href
38
39                # 获取一辆汽车的信息
40                self.get_data(url)
41                time.sleep(random.randint(1, 2))
42
43        # 获取一辆汽车信息
44        def get_data(self, url):
45            two_html = self.get_html(url)
46            two_regex = '<div class="car-box">.*?<h3 class="car-brand-name">(.*?)</h3>.*?<ul
47            class="brand-unit-item fn-clear">.*?<li>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)
48            </h4>.*?<h4>(.*?)</h4>.*?<span class="price" id="overlayPrice">¥(.*?)<b'
49
50            item = {}
51            car_info_list = self.re_func(two_regex, two_html)
52            item['name'] = car_info_list[0][0]
53            item['km'] = car_info_list[0][1]
54            item['year'] = car_info_list[0][2]
```

```

52         item['type'] = car_info_list[0][3].split('/')[0]
53         item['displacement'] = car_info_list[0][3].split('/')[1]
54         item['city'] = car_info_list[0][4]
55         item['price'] = car_info_list[0][5]
56         print(item)
57
58     def run(self):
59         for p in range(1,11):
60             url = self.url.format(p)
61             self.parse_html(url)
62
63 if __name__ == '__main__':
64     spider = CarSpider()
65     spider.run()

```

■ 扩展 - 增量爬取

```

1  # 将数据存入MySQL数据库 - 增量爬取
2  【1】思路
3      1.1 MySQL中新建表 urltab,存储所有爬取过的链接的指纹
4      1.2 在爬取之前,先判断该指纹是否爬取过,如果爬取过,则不再继续爬取
5
6  【2】建库建表
7  create database cardb charset utf8;
8  use cardb;
9  create table request_finger(
10     finger char(32)
11 )charset=utf8;
12 create table cartab(
13     name varchar(100),
14     km varchar(50),
15     years varchar(50),
16     type varchar(50),
17     displacement varchar(50),
18     city varchar(50),
19     price varchar(50)
20 )charset=utf8;

```

■ 增量爬取 - MySQL

```

1  from urllib import request
2  import re
3  import time
4  import random
5  import pymysql
6  from hashlib import md5
7  import sys
8
9
10 class CarSpider(object):
11     def __init__(self):
12         self.url = 'https://www.che168.com/beijing/a0_0msdgscncgpillto1csp{}exx0/'
13         self.headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}
14         self.db = pymysql.connect('localhost', 'root', '123456', 'cardb', charset='utf8')

```

```

15         self.cursor = self.db.cursor()
16
17     # 功能函数1 - 获取响应内容
18     def get_html(self,url):
19         req = request.Request(url=url,headers=self.headers)
20         try:
21             res = request.urlopen(req,timeout=3)
22             html = res.read().decode('gb2312','ignore')
23             return html
24         except Exception as e:
25             print(e)
26
27     # 功能函数2 - 正则解析
28     def re_func(self,regex,html):
29         pattern = re.compile(regex,re.S)
30         r_list = pattern.findall(html)
31
32         return r_list
33
34     # 爬虫函数开始
35     def parse_html(self,one_url):
36         one_html = self.get_html(one_url)
37         one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
38         href_list = self.re_func(one_regex,one_html)
39         for href in href_list:
40             # 加密指纹
41             s = md5()
42             s.update(href.encode())
43             finger = s.hexdigest()
44             # 如果指纹表中不存在
45             if self.go_spider(finger):
46                 # 每便利一个汽车信息，必须要把此辆汽车所有数据提取完成后再提取下一辆汽车信息
47                 url = 'https://www.che168.com' + href
48
49                 # 获取一辆汽车的信息
50                 self.get_data(url)
51                 ins = 'insert into request_finger values(%s)'
52                 self.cursor.execute(ins,[finger])
53                 self.db.commit()
54                 time.sleep(random.randint(1,2))
55             else:
56                 sys.exit('抓取结束')
57
58     # 判断是否存在：存在返回False，不存在返回True
59     def go_spider(self,finger):
60         sel = 'select * from request_finger where finger=%s'
61         result = self.cursor.execute(sel,[finger])
62         if result:
63             return False
64         return True
65
66     # 获取一辆汽车信息
67     def get_data(self,url):
68         two_html = self.get_html(url)
69         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'

```

```

70         item = {}
71         car_info_list = self.re_func(two_regex,two_html)
72         item['name'] = car_info_list[0][0]
73         item['km'] = car_info_list[0][1]
74         item['year'] = car_info_list[0][2]
75         item['type'] = car_info_list[0][3].split('/')[0]
76         item['displacement'] = car_info_list[0][3].split('/')[1]
77         item['city'] = car_info_list[0][4]
78         item['price'] = car_info_list[0][5]
79         print(item)
80
81         one_car_list = [
82             item['name'],
83             item['km'],
84             item['year'],
85             item['type'],
86             item['displacement'],
87             item['city'],
88             item['price']
89         ]
90         ins = 'insert into cartab values(%s,%s,%s,%s,%s,%s,%s,%s)'
91         self.cursor.execute(ins,one_car_list)
92         self.db.commit()
93
94     def run(self):
95         for p in range(1,2):
96             url = self.url.format(p)
97             self.parse_html(url)
98
99         # 断开数据库链接
100        self.cursor.close()
101        self.db.close()
102
103    if __name__ == '__main__':
104        spider = CarSpider()
105        spider.run()

```

■ 能不能使用redis来实现增量

```

1  """
2      提示：使用redis中的集合,sadd()方法,添加成功返回1,否则返回0
3      请各位大佬忽略掉下面代码,自己独立实现
4  """
5
6  from urllib import request
7  import re
8  import time
9  import random
10 import pymysql
11 from hashlib import md5
12 import sys
13 import redis
14
15
16 class CarSpider(object):
17     def __init__(self):

```

```

18     self.url = 'https://www.che168.com/beijing/a0_0msdgsncngpillto1csp{}exx0/'
19     self.headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.835.163 Safari/535.1'}
20     self.db = pymysql.connect('localhost', 'root', 'attack', 'cardb', charset='utf8')
21     self.cursor = self.db.cursor()
22     # 连接redis去重
23     self.r = redis.Redis(host='localhost', port=6379, db=0)
24
25     # 功能函数1 - 获取响应内容
26     def get_html(self, url):
27         req = request.Request(url=url, headers=self.headers)
28         res = request.urlopen(req)
29         html = res.read().decode('gb2312', 'ignore')
30
31         return html
32
33     # 功能函数2 - 正则解析
34     def re_func(self, regex, html):
35         pattern = re.compile(regex, re.S)
36         r_list = pattern.findall(html)
37
38         return r_list
39
40     # 爬虫函数开始
41     def parse_html(self, one_url):
42         one_html = self.get_html(one_url)
43         one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
44         href_list = self.re_func(one_regex, one_html)
45         for href in href_list:
46             # 加密指纹
47             s = md5()
48             s.update(href.encode())
49             finger = s.hexdigest()
50             # 如果指纹表中不存在
51             if self.r.sadd('car:urls', finger):
52                 # 每便利一个汽车信息，必须要把此辆汽车所有数据提取完成后再提取下一辆汽车信息
53                 url = 'https://www.che168.com' + href
54
55                 # 获取一辆汽车的信息
56                 self.get_data(url)
57                 time.sleep(random.randint(1, 2))
58             else:
59                 sys.exit('抓取结束')
60
61     # 判断是否存在：存在返回False，不存在返回True
62     def go_spider(self, finger):
63         sel = 'select * from request_finger where finger=%s'
64         result = self.cursor.execute(sel, [finger])
65         if result:
66             return False
67         return True
68
69     # 获取一辆汽车信息
70     def get_data(self, url):
71         two_html = self.get_html(url)

```

```

72         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'
73         item = {}
74         car_info_list = self.re_func(two_regex,two_html)
75         item['name'] = car_info_list[0][0]
76         item['km'] = car_info_list[0][1]
77         item['year'] = car_info_list[0][2]
78         item['type'] = car_info_list[0][3].split('/')[0]
79         item['displacement'] = car_info_list[0][3].split('/')[1]
80         item['city'] = car_info_list[0][4]
81         item['price'] = car_info_list[0][5]
82         print(item)
83
84         one_car_list = [
85             item['name'],
86             item['km'],
87             item['year'],
88             item['type'],
89             item['displacement'],
90             item['city'],
91             item['price']
92         ]
93         ins = 'insert into cartab values(%s,%s,%s,%s,%s,%s,%s)'
94         self.cursor.execute(ins,one_car_list)
95         self.db.commit()
96
97     def run(self):
98         for p in range(1,2):
99             url = self.url.format(p)
100             self.parse_html(url)
101
102         # 断开数据库链接
103         self.cursor.close()
104         self.db.close()
105
106 if __name__ == '__main__':
107     spider = CarSpider()
108     spider.run()

```

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、encoding : 响应字符编码
12     res.encoding = 'utf-8'
13 2、text : 字符串
14 3、content : 字节流
15 4、status_code : HTTP响应码
16 5、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)
```