

# Day01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 p = re.compile('正则表达式',re.S)
2 r_list = p.findall(html)
```

不使用re.S时，则只在每一行内进行匹配，如果存在一行没有，就换下一行重新开始，使用re.S参数以后，正则表达式会将这个字符串看做整体，在整体中进行匹配

### ■ 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 【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))
```

# spider-day02笔记

## 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1 - 使用 writerow() 方法将猫眼电影数据存入本地 maoyan.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

```

## ■ 练习2 - 使用 writerows() 方法将猫眼电影数据存入本地 maoyan.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 __init__() 中定义存储所有电影信息的空列表，用于后序写入文件
3  【3】在 save_html() 中将所抓取的数据处理成元组，并添加到总列表中
4  【4】在run() 中等数据抓取完成一次性使用writerows()写入后关闭文件

```

所有数据抓取完成后，一次性写入本地磁盘，并关闭文件，减少磁盘io次数

# 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 2.1) 在 __init__() 中连接数据库并创建游标对象
13 2.2) 在 save_html() 中将所抓取的数据处理成列表, 使用execute()方法写入
14 2.3) 在run() 中等数据抓取完成后关闭游标及断开数据库连接
15
16 import pymysql
17
18 def __init__(self):
19     # 添加代码
20     self.db = pymysql.connect('localhost', 'root', '123456', 'maoyandb', charset='utf8')
21     self.cursor = self.db.cursor()
22
23 def save_html(self, dd_list):
24     # 覆盖原来代码
25     ins = 'insert into maoyantab values(%s,%s,%s)'
26     for dd in dd_list:
27         # 将每个电影信息处理成列表
28         dd_li = [dd[0].strip(), dd[1].strip(), dd[2].strip()]
29         self.cursor.execute(ins, dd_li)
30         self.db.commit()
31         print(dd_li)
32         self.i += 1
33
34 def run(self):
35     # 添加代码
36     self.cursor.close()

```

```

37     self.db.close()
38
39 【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
40 3.1) 在 __init__() 中连接数据库及创建游标对象
41 3.2) 在 __init__() 中定义存储所有电影信息的空列表, 用于后序存入数据库
42 3.3) 在 save_html() 中将所抓取的数据处理成元组, 并添加到总列表中
43 3.4) 在run() 中等数据抓取完成一次性使用executemany()写入后断开数据库
44
45 import pymysql
46
47 def __init__(self):
48     # 添加代码
49     self.db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')
50     self.cursor = self.db.cursor()
51     # 存放所有电影信息的大列表
52     self.all_film_list = []
53
54 def save_html(self,dd_list):
55     # 覆盖原来代码
56     for dd in dd_list:
57         dd_tuple = (dd[0].strip(),dd[1].strip(),dd[2].strip())
58         self.all_film_list.append(dd_tuple)
59         self.i += 1
60
61 def run(self):
62     # 添加代码
63     ins = 'insert into maoyantab values(%s,%s,%s)'
64     self.cursor.executemany(ins,self.all_film_list)
65     self.db.commit()
66     self.cursor.close()
67     self.db.close()

```

## 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 端口 : 27017
- 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,
- 4 3.1> MySQL数据类型: 数值类型、字符类型、日期时间类型、枚举类型
- 5 3.2> Redis数据类型: 字符串、列表、哈希、集合、有序集合
- 6 3.3> MongoDB数据类型: 值为JSON文档
- 7 【4】MongoDB: 库 -> 集合 -> 文档
- 8 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     for r in r_list:
14         item = {}
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_0msdgcncgpi1lto1csp1exx0/

```

```

4
5
6 【2】爬取目标
7     所有汽车儿得 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_0msdgsncncgp11to1csp1exx0/
5     第2页: https://www.che168.com/beijing/a0_0msdgsncncgp11to1csp2exx0/
6     第n页: https://www.che168.com/beijing/a0_0msdgsncncgp11to1csp{ }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
11    class="brand-unit-item fn-clear">.*?<li>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)
12    </h4>.*?<h4>(.*?)</h4>.*?<span class="price" id="overlayPrice">¥(.*?)<b>

```

## ■ 代码实现

```

1 import requests
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_0msdgsncncgp11to1csp{ }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            html = requests.get(url=url, headers=self.headers).text
16
17            return html

```

```

18
19 # 功能函数2 - 正则解析
20 def re_func(self, regex, html):
21     pattern = re.compile(regex, re.S)
22     r_list = pattern.findall(html)
23
24     return r_list
25
26 # 爬虫函数开始
27 def parse_html(self, one_url):
28     one_html = self.get_html(one_url)
29     one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
30     href_list = self.re_func(one_regex, one_html)
31     for href in href_list:
32         # 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
33         url = 'https://www.che168.com' + href
34
35         # 获取一辆汽车的信息
36         self.get_data(url)
37         time.sleep(random.randint(1, 2))
38
39 # 获取一辆汽车信息
40 def get_data(self, url):
41     two_html = self.get_html(url)
42     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'
43     item = {}
44     car_info_list = self.re_func(two_regex, two_html)
45     item['name'] = car_info_list[0][0]
46     item['km'] = car_info_list[0][1]
47     item['year'] = car_info_list[0][2]
48     item['type'] = car_info_list[0][3].split('/')[0]
49     item['displacement'] = car_info_list[0][3].split('/')[1]
50     item['city'] = car_info_list[0][4]
51     item['price'] = car_info_list[0][5]
52     print(item)
53
54 def run(self):
55     for p in range(1, 11):
56         url = self.url.format(p)
57         self.parse_html(url)
58
59 if __name__ == '__main__':
60     spider = CarSpider()
61     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 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 import requests
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_0msdgsncncgpillto1csp{}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', '123456', '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         html = requests.get(url=url, headers=self.headers).text
28
29         return html
30
31     # 功能函数2 - 正则解析
32     def re_func(self, regex, html):
33         pattern = re.compile(regex, re.S)
34         r_list = pattern.findall(html)
35
36         return r_list
37
38     # 爬虫函数开始
39     def parse_html(self, one_url):
40         one_html = self.get_html(one_url)

```

```

41 one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
42 href_list = self.re_func(one_regex,one_html)
43 for href in href_list:
44     # 加密指纹
45     s = md5()
46     s.update(href.encode())
47     finger = s.hexdigest()
48     # 如果指纹表中不存在
49     if self.r.sadd('car:urls',finger):
50         # 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
51         url = 'https://www.che168.com' + href
52
53         # 获取一辆汽车的信息
54         self.get_data(url)
55         time.sleep(random.randint(1,2))
56     else:
57         sys.exit('抓取结束')
58
59 # 判断是否存在：存在返回False，不存在返回True
60 def go_spider(self,finger):
61     sel = 'select * from request_finger where finger=%s'
62     result = self.cursor.execute(sel,[finger])
63     if result:
64         return False
65     return True
66
67 # 获取一辆汽车信息
68 def get_data(self,url):
69     two_html = self.get_html(url)
70     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'
71     item = {}
72     car_info_list = self.re_func(two_regex,two_html)
73     item['name'] = car_info_list[0][0]
74     item['km'] = car_info_list[0][1]
75     item['year'] = car_info_list[0][2]
76     item['type'] = car_info_list[0][3].split('/')[0]
77     item['displacement'] = car_info_list[0][3].split('/')[1]
78     item['city'] = car_info_list[0][4]
79     item['price'] = car_info_list[0][5]
80     print(item)
81
82     one_car_list = [
83         item['name'],
84         item['km'],
85         item['year'],
86         item['type'],
87         item['displacement'],
88         item['city'],
89         item['price']
90     ]
91     ins = 'insert into cartab values(%s,%s,%s,%s,%s,%s,%s,%s)'
92     self.cursor.execute(ins,one_car_list)
93     self.db.commit()
94
95 def run(self):

```

```

96         for p in range(1,2):
97             url = self.url.format(p)
98             self.parse_html(url)
99
100         # 断开数据库链接
101         self.cursor.close()
102         self.db.close()
103
104     if __name__ == '__main__':
105         spider = CarSpider()
106         spider.run()

```

## requests模块高级

### ■ 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,timeout=3)
4
5  【2】参数
6      2.1) url : 需要抓取的URL地址
7      2.2) headers : 请求头
8      2.3) 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     3.1) text : 字符串
12     3.2) content : 字节流
13     3.3) status_code : HTTP响应码
14     3.4) 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?
        image&quality=80&size=b9999_10000&sec=1567090051520&di=77e8b97b3280f999cf51340af4315b4b&img
        type=jpg&src=http%3A%2F%2F5b0988e595225.cdn.sohucs.com%2Fimages%2F20171121%2F4e6759d153d04c
        6badbb0a5262ec103d.jpeg'
6  headers = {'User-Agent':'Mozilla/5.0'}
7
8  html = requests.get(url=url,headers=headers).content
9  with open('赵丽颖.jpg','wb') as f:
10     f.write(html)

```

# 今日作业

## ■ 百度图片抓取

```
1  【1】百度图片官网指定图片抓取：
2    1.1> 百度图片官网：http://image.baidu.com/
3    1.2> 运行效果
4          请输入关键字：赵丽颖
5          则自动创建文件夹： /home/tarena/images/赵丽颖/ 并把首页30张图片保存到此文件夹下
6
7  【2】注意
8    2.1> 一定要以响应内容为主来写正则表达式（右键 - 查看网页源代码）
9
10 【3】颠覆前两天课程认知的一个现实
11   3.1> 页面结构 - Elements，为页面最终渲染完成后的结构，和响应内容不一定完全一样
12   3.2> 原因1：可能会有部分数据为动态加载的
13         原因2：响应内容中存在JavaScript，对页面结构做了一定调整
14
15 【4】那我们写正则表达式时要以谁为准？
16   4.1> 必须以响应内容为准!!!!!! -> 右键，查看网页源代码为准
17   4.1> 必须以响应内容为准!!!!!! -> 右键，查看网页源代码为准
18   4.1> 必须以响应内容为准!!!!!! -> 右键，查看网页源代码为准
19
20   @@ 重要的事情说三遍，必须以响应内容为准 @@
```

## ■ 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) Windows中路径如何表示
8          方式1: E:\\spider\\spider_day03\\
9          方式2: E:/spider/spider_day03/
```

## ■ 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()
```