Day01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**

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

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))
```

spider-day02笔记

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】单行写入 - writerow([])方法
    import csv
    with open('test.csv','w') as f:
        writer = csv.writer(f)
        writer.writerow(['步惊云','36'])
        writer.writerow(['超哥哥','25'])

【3】多行写入 - writerows([(),(),()]) 方法
    import csv
    with open('test.csv','w') as f:
        writer = csv.writer(f)
        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】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
```

- **练习2 - 使用 writerows() 方法将猫眼电影数据存入本地 maoyan.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 __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】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】在run() 中等数据抓取完成一次性使用writerows()写入后关闭文件
```

数据持久化 - MySQL

- **pymysql回顾**

```
# 1. 单条插入表记录 - excute()方法
```

```
# 2. 多条插入表记录 - excutemany()方法

# 示例代码如下:
import pymysql

db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')
cursor = db.cursor()

ins = 'insert into filmtab values(%s,%s,%s)'
# 1. 单条插入表记录之 excute() 方法
cursor.execute(ins,['霸王别姬','张国荣','1993'])
# 2. 多条插入表记录之 excutemany() 方法 - 高效且节省资源
cursor.executemany(ins,[('大话1','周','1993'),('大话2','周','1994')])

db.commit()
cursor.close()
db.close()
```

- **练习 - 将电影信息存入MySQL数据库**

```
【1】提前建库建表
mysql -h127.0.0.1 -uroot -p123456
create database maoyandb charset utf8;
use maoyandb;
create table maoyantab(
name varchar(100),
star varchar(300),
time varchar(100)
)charset=utf8;

【2】 使用excute()方法将数据存入数据库 - 在初始代码基础上做如下改动
    2.1) 在 __init__() 中连接数据库并创建游标对象
    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
    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

import pymysql

def __init__(self):
    # 添加代码
    self.db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')
    self.cursor = self.db.cursor()

def save_html(self,dd_list):
    # 覆盖原来代码
    ins = 'insert into maoyantab values(%s,%s,%s)'
    for dd in dd_list:
        # 将每个电影信息处理成列表
        dd_li = [dd[0].strip(),dd[1].strip(),dd[2].strip()]
        self.cursor.execute(ins,dd_li)
        self.db.commit()
        print(dd_li)
        self.i += 1

def run(self):
    # 添加代码
    self.cursor.close()
```

```
    self.db.close()

【3】使用excutemany()方法将数据存入数据库 - 在初始代码基础上做如下改动
    3.1) 在 __init__() 中连接数据库及创建游标对象
    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
    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
    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

import pymysql

def __init__(self):
    # 添加代码
    self.db = pymysql.connect('localhost','root','123456','maoyandb',charset='utf8')
    self.cursor = self.db.cursor()
    # 存放所有电影信息的大列表
    self.all_film_list = []

def save_html(self,dd_list):
    # 覆盖原来代码
    for dd in dd_list:
        dd_tuple = (dd[0].strip(),dd[1].strip(),dd[2].strip())
        self.all_film_list.append(dd_tuple)
        self.i += 1

def run(self):
    # 添加代码
    ins = 'insert into maoyantab values(%s,%s,%s)'
    self.cursor.executemany(ins,self.all_film_list)
    self.db.commit()
    self.cursor.close()
    self.db.close()
```

数据持久化 - MongoDB

- **MongoDB特点**

```
【1】非关系型数据库,数据以键值对方式存储
【2】MongoDB基于磁盘存储
【3】MongoDB数据类型单一,值为JSON文档,而Redis基于内存,
  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
  3.2> Redis数据类型：字符串、列表、哈希、集合、有序集合
  3.3> MongoDB数据类型：值为JSON文档
【4】MongoDB：库 -> 集合 -> 文档
   MySQL  ：库 -> 表  ->  表记录
```

- **MongoDB常用命令**

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

- **pymongo回顾**

```
import pymongo

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

- **练习 - 将电影信息存入MongoDB数据库**

```
"""在初始代码基础上做如下更改"""
import pymongo

def __init__(self):
    # 添加
    self.conn = pymongo.MongoClient('localhost',27017)
    self.db = self.conn['maoyandb']
    self.myset = self.db['maoyanset']

def save_html(self,r_list):
    # 覆盖
    # 将数据处理为字典,执行insert_one()
    for r in r_list:
        item = {}
        item['name'] = r[0].strip()
        item['star'] = r[1].strip()
        item['time'] = r[2].strip()
        self.myset.insert_one(item)
```

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
    所有汽车儿得 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
```

- **代码实现**

```python
import requests
import re
import time
import random


class CarSpider(object):
    def __init__(self):
        self.url = 'https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp{}exx0/'
        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'}

    # 功能函数1 - 获取响应内容
    def get_html(self,url):
        html = requests.get(url=url,headers=self.headers).text

        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**

```
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**

```
"""
  提示: 使用redis中的集合,sadd()方法,添加成功返回1,否则返回0
  请各位大佬忽略掉下面代码,自己独立实现
"""

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


class CarSpider(object):
    def __init__(self):
        self.url = 'https://www.che168.com/beijing/a0_0msdgscncgpi1lto1csp{}exx0/'
        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'}
        self.db = pymysql.connect('localhost','root','123456','cardb',charset='utf8')
        self.cursor = self.db.cursor()
        # 连接redis去重
        self.r = redis.Redis(host='localhost',port=6379,db=0)

    # 功能函数1 - 获取响应内容
    def get_html(self,url):
        html = requests.get(url=url,headers=self.headers).text

        return html

    # 功能函数2 - 正则解析
    def re_func(self,regex,html):
        pattern = re.compile(regex,re.S)
        r_list = pattern.findall(html)

        return r_list

    # 爬虫函数开始
    def parse_html(self,one_url):
        one_html = self.get_html(one_url)
```

```
        one_regex = '<li class="cards-li list-photo-li".*?<a href="(.*?)".*?</li>'
        href_list = self.re_func(one_regex,one_html)
        for href in href_list:
            # 加密指纹
            s = md5()
            s.update(href.encode())
            finger = s.hexdigest()
            # 如果指纹表中不存在
            if self.r.sadd('car:urls',finger):
                # 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
                url = 'https://www.che168.com' + href

                # 获取一辆汽车的信息
                self.get_data(url)
                time.sleep(random.randint(1,2))
            else:
                sys.exit('抓取结束')

    # 判断是否存在：存在返回False，不存在返回True
    def go_spider(self,finger):
        sel = 'select * from request_finger where finger=%s'
        result = self.cursor.execute(sel,[finger])
        if result:
            return False
        return True

    # 获取一辆汽车信息
    def get_data(self,url):
        two_html = self.get_html(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'
        item = {}
        car_info_list = self.re_func(two_regex,two_html)
        item['name'] = car_info_list[0][0]
        item['km'] = car_info_list[0][1]
        item['year'] = car_info_list[0][2]
        item['type'] = car_info_list[0][3].split('/')[0]
        item['displacement'] = car_info_list[0][3].split('/')[1]
        item['city'] = car_info_list[0][4]
        item['price'] = car_info_list[0][5]
        print(item)

        one_car_list = [
            item['name'],
            item['km'],
            item['year'],
            item['type'],
            item['displacement'],
            item['city'],
            item['price']
        ]
        ins = 'insert into cartab values(%s,%s,%s,%s,%s,%s,%s)'
        self.cursor.execute(ins,one_car_list)
        self.db.commit()

    def run(self):
```

```
        for p in range(1,2):
            url = self.url.format(p)
            self.parse_html(url)

        # 断开数据库链接
        self.cursor.close()
        self.db.close()

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

requests模块高级

- **requests.get()**

```
【1】作用
    向目标网站发起请求,并获取响应对象
    res = requests.get(url=url,headers=headers,timeout=3)

【2】参数
    2.1) url : 需要抓取的URL地址
    2.2) headers : 请求头
    2.3) timeout : 超时时间，超过时间会抛出异常

【3】响应对象(res)属性
    3.1) text : 字符串
    3.2) content : 字节流
    3.3) status_code : HTTP响应码
    3.4) url : 实际数据的URL地址
```

- **非结构化数据保存**

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

- **示例代码 - 图片抓取**

```
# 保存赵丽颖图片到本地

import requests

url = 'https://timgsa.baidu.com/timg?image&quality=80&size=b9999_10000&sec=1567090051520&di=77e8b97b3280f999cf51340af4315b4b&imgtype=jpg&src=http%3A%2F%2F5b0988e595225.cdn.sohucs.com%2Fimages%2F20171121%2F4e6759d153d04c6badbb0a5262ec103d.jpeg'
headers = {'User-Agent':'Mozilla/5.0'}

html = requests.get(url=url,headers=headers).content
with open('赵丽颖.jpg','wb') as f:
    f.write(html)
```

今日作业

- **百度图片抓取**

```
【1】百度图片官网指定图片抓取：
  1.1> 百度图片官网：http://image.baidu.com/
  1.2> 运行效果
       请输入关键字：赵丽颖
       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

【2】注意
  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）

【3】颠覆前两天课程认知的一个现实
  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
  3.2> 原因1：可能会有部分数据为动态加载的
       原因2：响应内容中存在JavaScript，对页面结构做了一定调整

【4】那我们写正则表达式时要以谁为准?
  4.1> 必须以响应内容为准!!!!!! -> 右键,查看网页源代码为准
  4.1> 必须以响应内容为准!!!!!! -> 右键,查看网页源代码为准
  4.1> 必须以响应内容为准!!!!!! -> 右键,查看网页源代码为准

  @@ 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】右键,查看网页源码,搜索图片链接关键字 -> 存在
【2】分析URL地址规律
   https://image.baidu.com/search/index?tn=baiduimage&word={}
【3】正则表达式 - 以响应内容为准
   "thumbURL":"(.*?)"
【4】代码实现
   4.1) Windows中路径如何表示
        方式1：E:\\spider\\spider_day03\\
        方式2：E:/spider/spider_day03/
```

- **百度图片代码实现**

```python
import requests
import re
import time
import random
from fake_useragent import UserAgent
import os
from urllib import parse

class BaiduImageSpider(object):
    def __init__(self):
        self.url = 'https://image.baidu.com/search/index?tn=baiduimage&word={}'
        self.word = input('请输入关键字:')
        self.directory = '/home/tarena/images/{}/'.format(self.word)
        if not os.path.exists(self.directory):
            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()
```