Day01回顾

请求模块(requests)

```
html =
requests.get(url=url,headers=headers).tex
t
html =
requests.get(url=url,headers=headers).con
tent.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(爬虫专用) ：  .*?
```

- **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

```
# 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 - MySQL

- **pymysql回顾**

```
import pymysql

db =
pymysql.connect('localhost','root','123
456','maoyandb',charset='utf8')
cursor = db.cursor()

ins = 'insert into filmtab
values(%s,%s,%s)'
cursor.execute(ins,['霸王别姬','张国
荣','1993'])

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

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

- **代码实现**

```
"""
猫眼电影top100抓取（电影名称、主演、上映时间）存入csv文件,使用writerow()方法
"""
import requests
import re
import time
import random
import csv

class MaoyanSpider:
    def __init__(self):
        self.url = 'https://maoyan.com/board/4?offset={}'
        self.headers = {'User-Agent':'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0; .NET CLR 2.0.50727; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; Tablet PC 2.0; .NET4.0E)'}
```

```
        # 打开文件,初始化写入对象
        self.f = open('maoyan.csv',
'w', newline='', encoding='utf-8')
        self.writer =
csv.writer(self.f)

    def get_html(self, url):
        html = requests.get(url=url,
headers=self.headers).text
        # 直接调用解析函数
        self.parse_html(html)

    def parse_html(self, html):
        """解析提取数据"""
        regex = '<div class="movie-
item-info">.*?title="(.*?)".*?<p
class="star">(.*?)</p>.*?<p
class="releasetime">(.*?)</p>'
        pattern = re.compile(regex,
re.S)
        r_list = pattern.findall(html)
        # r_list: [('活着','牛犇','2000-
01-01'),(),(),...,()]
        self.save_html(r_list)

    def save_html(self, r_list):
        """数据处理函数"""
        for r in r_list:
```

```
                li = [ r[0].strip(),
r[1].strip(), r[2].strip() ]
                self.writer.writerow(li)
                print(li)

    def run(self):
        """程序入口函数"""
        for offset in range(0, 91, 10):
            url =
self.url.format(offset)
            self.get_html(url=url)
            # 控制数据抓取频率:uniform()生
成指定范围内的浮点数

 time.sleep(random.uniform(1,2))

        # 所有数据抓取并写入完成后关闭文件
        self.f.close()

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

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，端口27017
【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模块使用**

```
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'})
```

- **练习 - 将电影信息存入MongoDB数据库**

```
"""
猫眼电影top100抓取（电影名称、主演、上映时间）
存入mongodb数据库中
"""
import requests
import re
import time
import random
import pymongo

class MaoyanSpider:
    def __init__(self):
        self.url =
'https://maoyan.com/board/4?offset={}'
```

```
        self.headers = {'User-
Agent':'Mozilla/5.0 (Windows NT 10.0;
WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/81.0.4044.113
Safari/537.36'}
        # 三个对象：连接对象、库对象、集合对
象
        self.conn =
pymongo.MongoClient('127.0.0.1', 27017)
        self.db = self.conn['maoyandb']
        self.myset =
self.db['maoyanset2']

    def get_html(self, url):
        html = requests.get(url=url,
headers=self.headers).text
        # 直接调用解析函数
        self.parse_html(html)

    def parse_html(self, html):
        """解析提取数据"""
        regex = '<div class="movie-
item-info">.*?title="(.*?)".*?<p
class="star">(.*?)</p>.*?<p
class="releasetime">(.*?)</p>'
        pattern = re.compile(regex,
re.S)
        r_list = pattern.findall(html)
```

```
        # r_list: [('活着','牛犇','2000-
01-01'),(),(),...,()]
        self.save_html(r_list)

    def save_html(self, r_list):
        """数据处理函数"""
        for r in r_list:
            item = {}
            item['name'] = r[0].strip()
            item['star'] = r[1].strip()
            item['time'] = r[2].strip()
            print(item)
            # 存入到mongodb数据库
            self.myset.insert_one(item)

    def run(self):
        """程序入口函数"""
        for offset in range(0, 91, 10):
            url =
self.url.format(offset)
            self.get_html(url=url)
            # 控制数据抓取频率:uniform()生
成指定范围内的浮点数

 time.sleep(random.uniform(0,1))

if __name__ == '__main__':
    spider = MaoyanSpider()
    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="(.*?)".*?

二级页面正则表达式: .*?<h3 class="car-brand-name">(.*?)</h3>.*?<ul class="brand-unit-item fn-clear">.*?<li>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)</h4>.*?￥(.*?)<b>

【4】代码实现

- **代码实现**

```
"""
汽车之家二手车信息抓取
思路
    1、一级页面：汽车的链接
    2、二级页面：具体汽车信息

建立User-Agent池：防止被网站检测到是爬虫
    使用fake_useragent模块
    安装：sudo pip3 install fake_useragent
    使用：
        from fake_useragent import UserAgent
        UserAgent().random
"""
import requests
import re
import time
import random
from fake_useragent import UserAgent

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

```
    def get_html(self, url):
        """功能函数1 - 获取html"""
        headers = { 'User-
Agent':UserAgent().random }
        html = requests.get(url=url,
headers=headers).text

        return html

    def re_func(self, regex, html):
        """功能函数2 - 正则解析函数"""
        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 =
self.re_func(regex=one_regex,
html=one_html)
        for href in href_list:
```

```
                two_url =
'https://www.che168.com' + href
                # 获取1辆汽车的具体信息
                self.get_car_info(two_url)
                # 控制爬取频率

 time.sleep(random.randint(1,2))

    def get_car_info(self, two_url):
        """获取1辆汽车的具体信息"""
        two_html =
self.get_html(url=two_url)
        two_regex = '<div class="car-
box">.*?<h3 class="car-brand-name">
(.*?)</h3>.*?<h4>(.*?)</h4>.*?<h4>(.*?)
</h4>.*?<h4>(.*?)</h4>.*?<h4>(.*?)
</h4>.*?<span class="price"
id="overlayPrice">￥(.*?)<b>'
        # car_list: [('福睿斯','3万公
里','2016年3月','手动 / 1.5L', '廊坊',
'5.60'),]
        car_list =
self.re_func(regex=two_regex,
html=two_html)
        item = {}
        item['name'] = car_list[0]
[0].strip()
        item['km'] = car_list[0]
[1].strip()
```

```
            item['time'] = car_list[0]
[2].strip()
            item['type'] = car_list[0]
[3].split('/')[0].strip()
            item['displace'] = car_list[0]
[3].split('/')[1].strip()
            item['address'] = car_list[0]
[4].strip()
            item['price'] = car_list[0]
[5].strip()
            print(item)

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

```
"""
  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_0msd
gscncgpi1lto1csp{}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','123
456','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.heade
rs).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()
```

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()
```

==lxml解析库==

- **安装**

```
【1】Ubuntu:  sudo pip3 install lxml
【2】Windows: python -m pip install lxml
```

- **使用流程**

```
1、导模块
    from lxml import etree
2、创建解析对象
    parse_html = etree.HTML(html)
3、解析对象调用xpath
    r_list = parse_html.xpath('xpath表达式')
```

- **xpath最常用**

```
【1】基准xpath: 匹配所有电影信息的节点对象列表
    //dl[@class="board-wrapper"]/dd
    [<element dd at xxx>,<element dd at xxx>,...]

【2】遍历对象列表，依次获取每个电影信息
    item = {}
    for dd in dd_list:
        item['name'] = dd.xpath('.//p[@class="name"]/a/text()').strip()
        item['star'] = dd.xpath('.//p[@class="star"]/text()').strip()
        item['time'] = dd.xpath('.//p[@class="releasetime"]/text()').strip()
```

- **猫眼电影案例-xpath实现**

```
"""
猫眼电影top100抓取（电影名称、主演、上映时间）
"""
import requests
import time
import random
from lxml import etree

class MaoyanSpider:
    def __init__(self):
        self.url =
'https://maoyan.com/board/4?offset={}'
        self.headers = {'User-
Agent':'Mozilla/5.0 (compatible; MSIE
9.0; Windows NT 6.1; Win64; x64;
Trident/5.0; .NET CLR 2.0.50727; SLCC2;
.NET CLR 3.5.30729; .NET CLR 3.0.30729;
Media Center PC 6.0; InfoPath.3;
.NET4.0C; Tablet PC 2.0; .NET4.0E)'}

    def get_html(self, url):
        html = requests.get(url=url,
headers=self.headers).text
        # 直接调用解析函数
        self.parse_html(html)

    def parse_html(self, html):
        """解析提取数据 - xpath"""
        p = etree.HTML(html)
```

```
        # 基准xpath：每个电影信息的节点对象dd列表 [<element dd at xxx>, <element dd at xxx>,...]
        dd_list = p.xpath('//dl[@class="board-wrapper"]/dd')
        print(dd_list)
        item = {}
        for dd in dd_list:
            item['name'] = dd.xpath('.//p[@class="name"]/a/@title')[0].strip()
            item['star'] = dd.xpath('.//p[@class="star"]/text()')[0].strip()
            item['time'] = dd.xpath('.//p[@class="releasetime"]/text()')[0].strip()
            print(item)

    def run(self):
        """程序入口函数"""
        for offset in range(0, 91, 10):
            url = self.url.format(offset)
            self.get_html(url=url)
            # 控制数据抓取频率:uniform()生成指定范围内的浮点数
```

```
 time.sleep(random.uniform(0,1))

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

- **小作业**

汽车之家案例使用lxml+xpath实现