*Day02回顾**

爬取网站思路

```
1、先确定是否为动态加载网站
2、找URL规律
3、正则表达式
4、定义程序框架，补全并测试代码
```

存入csv文件

```
import csv
with open('xxx.csv','w') as f:
    writer = csv.writer(f)
    writer.writerow([])
    writer.writerows([(),(),()])
```

持久化存储之MySQL

```
db = pymysql.connect('IP',...)
cursor = db.cursor()
# cursor.execute('SQL',[])
# cursor.executemany('SQL',[(),(),()])
db.commit()
cursor.close()
db.close()
```

requests模块

- get()

```
1、发请求并获取响应对象
2、res = requests.get(url,headers=headers)
```

- 响应对象res属性

```
res.text : 字符串
res.content : bytes
res.encoding: 字符编码 res.encoding='utf-8'
res.status_code : HTTP响应码
res.url : 实际数据URL地址
```

- 非结构化数据保存

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

多级页面数据抓取

```
1、先爬去一级页面,提取链接,继续跟进
2、爬取二级页面,提取数据
3、... ...
```

Chrome浏览器安装插件

- 安装方法

```
1、把下载的相关插件（对应操作系统浏览器）后缀改为 .zip
2、打开Chrome浏览器 -> 右上角设置 -> 更多工具 -> 扩展程序 -> 点开开发者模式
3、把相关插件 拖拽 到浏览器中，释放鼠标即可安装
4、重启浏览器，使插件生效
```

- 需要安装插件

```
1、Xpath Helper: 轻松获取HTML元素的xPath路径
2、Proxy SwitchyOmega: Chrome浏览器中的代理管理扩展程序
3、JsonView: 格式化输出json格式数据
```

Day03笔记

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
```

- 示例HTML代码

```
<ul class="book_list">
    <li>
        <title class="book_001">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>69.99</price>
    </li>

    <li>
        <title class="book_002">Spider</title>
        <author>Forever</author>
        <year>2019</year>
        <price>49.99</price>
    </li>
</ul>
```

- 匹配演示

```
1、查找所有的li节点
   //li
2、查找li节点下的title子节点中,class属性值为'book_001'的节点
   //li/title[@class="book_001"]
3、查找li节点下所有title节点的,class属性的值
   //li//title/@class

# 只要涉及到条件,加 []
# 只要获取属性值,加 @
```

- 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
   # 使用场景1（属性值作为条件）
     //div[@class="movie"]
   # 使用场景2（直接获取属性值）
     //div/a/@src
```

- 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、contains() : 匹配属性值中包含某些字符串节点
   # 查找class属性值中包含"book_"的title节点
     //title[contains(@class,"book_")]
2、text() : 获取节点的文本内容
   # 查找所有书籍的名称
     //ul[@class="book_list"]/li/title/text()
```

lxml解析库

- 安装

```
sudo pip3 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')
```

- 练习

```
from lxml import etree

html = '''<div class="wrapper">
    <i class="iconfont icon-back" id="back"></i>
    <a href="/" id="channel">新浪社会</a>
    <ul id="nav">
        <li><a href="http://domestic.firefox.sina.com/" title="国内">国内</a></li>
        <li><a href="http://world.firefox.sina.com/" title="国际">国际</a></li>
        <li><a href="http://mil.firefox.sina.com/" title="军事">军事</a></li>
        <li><a href="http://photo.firefox.sina.com/" title="图片">图片</a></li>
        <li><a href="http://society.firefox.sina.com/" title="社会">社会</a></li>
        <li><a href="http://ent.firefox.sina.com/" title="娱乐">娱乐</a></li>
        <li><a href="http://tech.firefox.sina.com/" title="科技">科技</a></li>
        <li><a href="http://sports.firefox.sina.com/" title="体育">体育</a></li>
        <li><a href="http://finance.firefox.sina.com/" title="财经">财经</a></li>
        <li><a href="http://auto.firefox.sina.com/" title="汽车">汽车</a></li>
    </ul>
    <i class="iconfont icon-liebiao" id="menu"></i>
</div>'''
# 创建解析对象
parseHtml = etree.HTML(html)
# 调用xpath返回结束,text()为文本内容
rList = parseHtml.xpath('//a/text()')
#print(rList)

# 提取所有的href的属性值
r2 = parseHtml.xpath('//a/@href')
```

```
#print(r2)

# 提取所有href的值,不包括 /
r3 = parseHtml.xpath('//ul[@id="nav"]/li/a/@href')
#print(r3)

# 获取 图片、军事、...,不包括新浪社会
r4 = parseHtml.xpath('//ul[@id="nav"]/li/a/text()')
for r in r4:
    print(r)
```

猫眼电影（xpath）

- 目标

```
1、地址：猫眼电影 - 榜单 - top100榜
2、目标：电影名称、主演、上映时间
```

- 步骤

```
1、确定是否为静态页面（右键-查看网页源代码，搜索关键字确认）
2、写xpath表达式
3、写程序框架
```

- xpath表达式

```
1、基准xpath：匹配所有电影信息的节点对象列表
    //dl[@class="board-wrapper"]/dd

2、遍历对象列表，依次获取每个电影信息
   for dd in dd_list:
       电影名称 ：dd.xpath('./a/@title')[0].strip()
       电影主演 ：dd.xpath('.//p[@class="star"]/text()')[0].strip()
       上映时间 ：dd.xpath('.//p[@class="releasetime"]/text()')[0].strip()
```

- 代码实现（修改之前urllib库代码）

```
1、将urllib库改为requests模块实现
2、改写parse_page()方法
```

```
import requests
from lxml import etree
import time
import random

class MaoyanSpider(object):
  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/74.0.3729.169 Safari/537.36'}
```

```
    # 添加计数(页数)
    self.page = 1

  # 获取页面
  def get_page(self,url):
    # random.choice一定要写在这里,每次请求都会随机选择
    res = requests.get(url,headers=self.headers)
    res.encoding = 'utf-8'
    html = res.text
    self.parse_page(html)

  # 解析页面
  def parse_page(self,html):
    #  创建解析对象
    parse_html = etree.HTML(html)
    # 基准xpath节点对象列表
    dd_list = parse_html.xpath('//dl[@class="board-wrapper"]/dd')
    movie_dict = {}
    # 依次遍历每个节点对象,提取数据
    for dd in dd_list:
      movie_dict['name'] = dd.xpath('.//p/a/@title')[0].strip()
      movie_dict['star'] = dd.xpath('.//p[@class="star"]/text()')[0].strip()
      movie_dict['time'] = dd.xpath('.//p[@class="releasetime"]/text()')[0].strip()

      print(movie_dict)

  # 主函数
  def main(self):
    for offset in range(0,31,10):
      url = self.url.format(str(offset))
      self.get_page(url)
      print('第%d页完成' % self.page)
      time.sleep(random.randint(1,3))
      self.page += 1

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

链家二手房案例（xpath）

- 实现步骤

1. 确定是否为静态

```
打开二手房页面 -> 查看网页源码 -> 搜索关键字
```

2. **xpath表达式**

```
1、修改方法：右键 -> copy xpath -> 测试修改
2、基准xpath表达式(匹配每个房源信息节点列表)
   //ul[@class="sellListContent"]/li[@class="clear LOGCLICKDATA"] |
//ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]
3、依次遍历后每个房源信息xpath表达式
   * 名称：.//a[@data-el="region"]/text()
   * 总价：.//div[@class="totalPrice"]/span/text()
   * 单价：.//div[@class="unitPrice"]/span/text()
```

3. **代码实现**

```python
import requests
from lxml import etree
import time

class LianjiaSpider(object):
  def __init__(self):
    self.url = 'https://bj.lianjia.com/ershoufang/pg{}/'
    self.headers = {'User-Agent' : 'Mozilla/5.0'}

  def get_page(self,url):
    res = requests.get(url,headers=self.headers,timeout=10)
    res.encoding = 'utf-8'
    html = res.text
    self.parse_page(html)

  def parse_page(self,html):
    parse_html = etree.HTML(html)
    # 基准xpath
    li_list = parse_html.xpath('//ul[@class="sellListContent"]/li[@class="clear LOGCLICKDATA"]
| //ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]')
    print(len(li_list))
    house_dict = {}
    # 遍历依次匹配每个房源信息
    for li in li_list:
      house_dict['house_name'] = li.xpath('.//a[@data-el="region"]/text()')[0].strip()
      house_dict['total_price'] = li.xpath('.//div[@class="totalPrice"]/span/text()')
[0].strip()
      house_dict['unit_price'] = li.xpath('.//div[@class="unitPrice"]/span/text()')[0].strip()

      print(house_dict)

  def main(self):
    for pg in range(1,4):
      url = self.url.format(str(pg))
      self.get_page(url)
      print('第%d页爬取成功' % pg)
      time.sleep(0.5)

if __name__ == '__main__':
  spider = LianjiaSpider()
  spider.main()
```

百度贴吧图片抓取

- 目标

```
抓取指定贴吧所有图片
```

- 思路

```
1、获取贴吧主页URL,下一页,找到不同页的URL规律
2、获取1页中所有帖子URL地址: [帖子链接1,帖子链接2,...]
3、对每个帖子链接发请求,获取图片URL
4、向图片的URL发请求,以wb方式写入本地文件
```

- 实现步骤

1. **贴吧URL规律**

```
http://tieba.baidu.com/f?kw=??&pn=50
```

2. **xpath表达式**

```
1、帖子链接xpath
   //*[@id="thread_list"]/li[@class=" j_thread_list clearfix"]/div/div[2]/div[1]/div[1]/a/@href

2、图片链接xpath
   //div[@class="d_post_content j_d_post_content  clearfix"]/img[@class="BDE_Image"]/@src

3、视频链接xpath
   //div[@class="video_src_wrapper"]/embed/@data-video
   # 注意: 此处视频链接前端对响应内容做了处理,需要查看网页源代码来查看, 复制HTML代码在线格式化
```

3. **代码实现**

```
import requests
from urllib import parse
from lxml import etree

class BaiduImgSpider(object):
  def __init__(self):
    self.url = 'http://tieba.baidu.com/f?{}'
    self.headers = {'User-Agent':'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)'}

  # 获取帖子链接
  def get_tlink(self,url):
    html = requests.get(url,headers=self.headers).text
    # 提取帖子链接
    parse_html = etree.HTML(html)
    tlink_list = parse_html.xpath('//*[@id="thread_list"]/li[@class=" j_thread_list clearfix"]/div/div[2]/div[1]/div[1]/a/@href')
    # tlink_list: ['/p/23234','/p/9032323']
```

```
    for tlink in tlink_list:
      t_url = 'http://tieba.baidu.com' + tlink
      # 提取图片链接并保存
      self.get_imglink(t_url)

  # 获取图片链接
  def get_imglink(self,t_url):
    res = requests.get(t_url,headers=self.headers)
    res.encoding = 'utf-8'
    html = res.text
    # 提取图片链接
    parse_html = etree.HTML(html)
    # 匹配图片和视频的xpath表达式,中间加或 |
    imglink_list = parse_html.xpath('//*[@class="d_post_content j_d_post_content
clearfix"]/img/@src | //div[@class="video_src_wrapper"]/embed/@data-video')

    for imglink in imglink_list:
      self.write_img(imglink)

  # 保存图片
  def write_img(self,imglink):
    res = requests.get(imglink,headers=self.headers)
    # 切取链接的后10位作为文件名
    filename = imglink[-10:]
    with open(filename,'wb') as f:
      f.write(res.content)
      print('%s下载成功' % filename)

  # 指定贴吧名称,起始页和终止页,爬取图片
  def main(self):
    name = input('请输入贴吧名:')
    begin = int(input('请输入起始页:'))
    end = int(input('请输入终止页:'))
    for page in range(begin,end+1):
      # 查询参数编码
      params = {
        'kw' : name,
        'pn' : str( (page-1)*50 )
      }
      params = parse.urlencode(params)
      url = self.url.format(params)
      # 开始获取图片
      self.get_tlink(url)

if __name__ == '__main__':
  spider = BaiduImgSpider()
  spider.main()
```

requests.get()参数

查询参数-params

- 参数类型

```
字典,字典中键值对作为查询参数
```

- 使用方法

```
1、res = requests.get(url,params=params,headers=headers)
2、特点:
   * url为基准的url地址，不包含查询参数
   * 该方法会自动对params字典编码,然后和url拼接
```

- 示例

```
import requests

baseurl = 'http://tieba.baidu.com/f?'
params = {
  'kw' : '赵丽颖吧',
  'pn' : '50'
}
headers = {'User-Agent' : 'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; InfoPath.3)'}
# 自动对params进行编码,然后自动和url进行拼接,去发请求
res = requests.get(baseurl,params=params,headers=headers)
res.encoding = 'utf-8'
print(res.text)
```

Web客户端验证 参数-auth

- 作用及类型

```
1、针对于需要web客户端用户名密码认证的网站
2、auth = ('username','password')
```

- 达内code课程方向案例

```
import requests
import re

class NoteSpider(object):
    def __init__(self):
        self.url = 'http://code.tarena.com.cn/'
        self.headers = {'User-Agent':'Mozilla/5.0'}
        self.auth = ('tarenacode','code_2013')

    # 获取+解析
    def get_parse_page(self):
        res = requests.get(
            url=self.url,
```

```
            auth=self.auth,
            headers=self.headers
        )
        res.encoding = 'utf-8'
        html = res.text
        # 解析
        p = re.compile('<a href=.*?>(.*?)/</a>',re.S)
        r_list = p.findall(html)
        # r_list : ['..','AIDCode','ACCCode']
        for r in r_list:
            if r != '..':
                print({ '课程方向' : r })

if __name__ == '__main__':
    spider = NoteSpider()
    spider.get_parse_page()
```

思考：爬取具体的笔记文件?

SSL证书认证参数-verify

- 适用网站及场景

```
1、适用网站：https类型网站但是没有经过 证书认证机构 认证的网站
2、适用场景：抛出 SSLError 异常则考虑使用此参数
```

- 参数类型

```
1、verify=True(默认)   ：检查证书认证
2、verify=False (常用) ：忽略证书认证
# 示例
response = requests.get(
    url=url,
    params=params,
    headers=headers,
    verify=False
)
```

代理参数-proxies

- 定义

```
1、定义：代替你原来的IP地址去对接网络的IP地址。
2、作用：隐藏自身真实IP,避免被封。
```

- 普通代理

获取代理IP网站

```
西刺代理、快代理、全网代理、代理精灵、... ...
```

参数类型

```
1、语法结构
    proxies = {
        '协议':'协议://IP:端口号'
    }
2、示例
    proxies = {
        'http':'http://IP:端口号',
        'https':'https://IP:端口号'
    }
```

示例

1. 使用免费普通代理IP访问测试网站: http://httpbin.org/get

```
import requests

url = 'http://httpbin.org/get'
headers = {
    'User-Agent':'Mozilla/5.0'
}
# 定义代理,在代理IP网站中查找免费代理IP
proxies = {
    'http':'http://115.171.85.221:9000',
    'https':'https://115.171.85.221:9000'
}
html = requests.get(url,proxies=proxies,headers=headers,timeout=5).text
print(html)
```

2. 思考: 建立一个自己的代理IP池，随时更新用来抓取网站数据

```
import requests
import random
from lxml import etree
from fake_useragent import UserAgent
import time

# 生成随机的User-Agent
def get_random_ua():
    # 创建User-Agent对象
    ua = UserAgent()
    # 随机生成1个User-Agent
    return ua.random

# 请求头
headers = {
    'User-Agent': get_random_ua()
```

```python
}
url = 'http://httpbin.org/get'

# 从西刺代理网站上获取随机的代理IP
def get_ip_list():
    # 访问西刺代理网站国内高匿代理，找到所有的tr节点对象
    res = requests.get('https://www.xicidaili.com/nn/', headers=headers)
    parse_html = etree.HTML(res.text)
    # 基准xpath，匹配每个代理IP的节点对象列表
    ipobj_list = parse_html.xpath('//tr')
    # 定义空列表，获取网页中所有代理IP地址及端口号
    ip_list = []
    # 从列表中第2个元素开始遍历，因为第1个为：字段名（国家、IP、... ...）
    for ip in ipobj_list[1:]:
        ip_info = ip.xpath('./td[2]/text()')[0]
        port_info = ip.xpath('./td[3]/text()')[0]
        ip_list.append(
            {
                'http' :'http://' + ip_info + ':' + port_info,
                'https':'https://' + ip_info + ':' + port_info
            }
        )
    # 随机选择一个代理
    proxies = random.choice(ip_list)
    # 返回代理IP及代理池（列表ip_list）
    return ip_list

# 主程序
def main_print():
    # 我的IP代理池
    ip_list = get_ip_list()
    while True:
        try:
            # 设置超时时间，如果代理不能使用则切换下一个
            proxies = random.choice(ip_list)
            res = requests.get(url=url, headers=headers, proxies=proxies,timeout=5)
            res.encoding = 'utf-8'
            print(res.text)

        except Exception as e:
            # 此代理IP不能使用，从代理池中移除
            ip_list.remove(proxies)
            print('%s不能用，已经移除' % proxies)
            # 继续循环获取最后1个代理IP
            continue


if __name__ == '__main__':
    main_print()
```

今日作业

糗事百科（xpath）

```
1、URL地址: https://www.qiushibaike.com/text/
2、目标 ： 用户昵称、段子内容、好笑数量、评论数量
```

电影天堂（xpath）