Spider_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
```

数据持久化 - csv

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

数据持久化 - MySQL

```
import pymysql

# __init__(self):
    self.db = pymysql.connect('IP',... ...)
    self.cursor = self.db.cursor()
# write_data(self):
    self.cursor.execute('sql',[data1])
    self.cursor.executemany('sql',[(data1),(data2),(data3)])
    self.db.commit()
# main(self):
    self.cursor.close()
    self.db.close()
```

数据持久化 - MongoDB

```
import pymongo

```

```
# __init__(self):
    self.conn = pymongo.MongoClient('IP',27017)
    self.db = self.conn['db_name']
    self.myset = self.db['set_name']
# write_data(self):
    self.myset.insert_one(dict)

# MongoDB - Commmand
>show dbs
>use db_name
>show collections
>db.collection_name.find().pretty()
>db.collection_name.count()
>db.collection_name.drop()
>db.dropDatabase()
```

多级页面数据抓取

```
# 整体思路
1、爬取一级页面,提取 所需数据+链接,继续跟进
2、爬取二级页面,提取 所需数据+链接,继续跟进
3、... ...
# 代码实现思路
1、所有数据最终都会在一级页面解析函数中拿到
2、避免重复代码 - 请求、解析需定义函数
```

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)
```

Chrome浏览器安装插件

- 安装方法

```
# 在线安装
1、下载插件 - google访问助手
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)
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

# 离线安装
1、下载插件 - xxx.crx 重命名为 xxx.zip
2、输入地址: chrome://extensions/   打开- 开发者模式
3、拖拽 插件(或者解压后文件夹) 到浏览器中
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
   开启/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
```

Spider_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="CarList">
    <li class="bjd" id="car_001" href="http://www.bjd.com/">
        <p class="name">布加迪</p>
        <p class="model">威航</p>
        <p class="price">2500万</p>
        <p class="color">红色</p>
    </li>

    <li class="byd" id="car_002" href="http://www.byd.com/">
        <p class="name">比亚迪</p>
        <p class="model">秦</p>
        <p class="price">15万</p>
        <p class="color">白色</p>
    </li>
```

```
15 </ul>
```

- 匹配演示

```
1、查找所有的li节点
   //li
2、获取所有汽车的名称：所有li节点下的子节点p的值（class属性值为name）
   //li/p[@class="name"]
3、找比亚迪车的信息：获取ul节点下第2个li节点的汽车信息
   //ul[@class="CarList"]/li[2]/p
4、获取所有汽车的链接：ul节点下所有li子节点的href属性的值
   //ul[@class="CarList"]/li/@href

# 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-item-info"]
   # 使用场景2（直接获取属性值）
     //div[@class="movie-item-info"]/a/img/@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id属性值中包含字符串 "car_" 的 li 节点
     //li[contains(@id,"car_")]
2、text() ：获取节点的文本内容
   # 查找所有汽车的价格
     //ul[@class="CarList"]/li/p[@class="price"]/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')
```

- **html样本**

```
<div class="wrapper">
    <a href="/" id="channel">新浪社会</a>
    <ul id="nav">
        <li><a href="http://domestic.sina.com/" title="国内">国内</a></li>
        <li><a href="http://world.sina.com/" title="国际">国际</a></li>
        <li><a href="http://mil.sina.com/" title="军事">军事</a></li>
        <li><a href="http://photo.sina.com/" title="图片">图片</a></li>
        <li><a href="http://society.sina.com/" title="社会">社会</a></li>
        <li><a href="http://ent.sina.com/" title="娱乐">娱乐</a></li>
        <li><a href="http://tech.sina.com/" title="科技">科技</a></li>
        <li><a href="http://sports.sina.com/" title="体育">体育</a></li>
        <li><a href="http://finance.sina.com/" title="财经">财经</a></li>
        <li><a href="http://auto.sina.com/" title="汽车">汽车</a></li>
    </ul>
</div>
```

- **示例+练习**

```
from lxml import etree

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

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

```
28
29  # 提取所有href的值,不包括 /
30  r3 = parse_html.xpath('//ul[@id="nav"]/li/a/@href')
31  #print(r3)
32
33  # 获取 图片、军事、...,不包括新浪社会
34  r4 = parse_html.xpath('//ul[@id="nav"]/li/a/text()')
35  for r in r4:
36      print(r)
```

xpath最常使用方法

```
1、先匹配节点对象列表
  # r_list: ['节点对象1','节点对象2']
  r_list = parse_html.xpath('基准xpath表达式')
2、遍历每个节点对象,利用节点对象继续调用 xpath
  for r in r_list:
        name = r.xpath('./xxxxxx')
        star = r.xpath('.//xxxxx')
        time = r.xpath('.//xxxxx')
```

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、基准xpath表达式(匹配每个房源信息节点列表)
  //ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]

2、依次遍历后每个房源信息xpath表达式
   * 名称: './/a[@data-el="region"]/text()'

   # 户型+面积+方位+是否精装
   info_list = './/div[@class="houseInfo"]/text()'  [0].strip().split('|')
   * 户型: info_list[1]
   * 面积: info_list[2]
   * 方位: info_list[3]
   * 精装: info_list[4]


   * 楼层: './/div[@class="positionInfo"]/text()'
   * 区域: './/div[@class="positionInfo"]/a/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
import random

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):
    try:
        # 设定超时时间,超时后抛出异常,被except捕捉,继续执行此函数再次请求
        res = requests.get(url,headers=self.headers,timeout=5)
        res.encoding = 'utf-8'
        html = res.text
        self.parse_page(html)
    except Exception as e:
        self.get_page(url)

  def parse_page(self,html):
    parse_html = etree.HTML(html)
    # 基准xpath,匹配每个房源信息的节点对象
    li_list = parse_html.xpath('//ul[@class="sellListContent"]/li[@class="clear LOGVIEWDATA LOGCLICKDATA"]')
    # 定义空字典,用来存储抓取的最终数据
    house_dict = {}
    # 遍历依次匹配每个房源信息,获取所有所需数据
    for li in li_list:
      # 房源名称
      name_list = li.xpath('.//a[@data-el="region"]/text()')
      house_dict['house_name'] = [ name_list[0] if name_list else None ][0]

      # 列表: 户型+面积+方位+是否精装
      info_list = li.xpath('.//div[@class="houseInfo"]/text()')
      house_info = [ info_list[0].strip().split('|') if info_list else None ][0]
      if house_info:
          # 户型
          house_dict['house_model'] = house_info[1]
          # 面积
          house_dict['area'] = house_info[2]
          # 方位
          house_dict['direction'] = house_info[3]
          # 是否精装
          house_dict['hardcover'] = house_info[4]
      ##########################################
      # 楼层
      floor_list = li.xpath('.//div[@class="positionInfo"]/text()')
      house_dict['floor'] = [ floor_list[0].strip()[:-2] if floor_list else None ][0]
      # 区域
      address_list = li.xpath('.//div[@class="positionInfo"]/a/text()')
      house_dict['address'] = [ address_list[0].strip() if address_list else None ][0]
      # 总价
      total_list = li.xpath('.//div[@class="totalPrice"]/span/text()')
      house_dict['total_price'] = [ total_list[0].strip() if total_list else None ][0]
      # 单价
```

```
        unit_list = li.xpath('.//div[@class="unitPrice"]/span/text()')
        house_dict['unit_price'] = [ unit_list[0].strip() if unit_list else None ][0]

        print(house_dict)

    def main(self):
        for pg in range(1,11):
            url = self.url.format(str(pg))
            self.get_page(url)
            print('第%d页爬取成功' % pg)
            time.sleep(random.randint(1,3))

if __name__ == '__main__':
    start = time.time()
    spider = LianjiaSpider()
    spider.main()
    end = time.time()
    print('执行时间:%.2f' % (end-start))
```

练习: 将猫眼电影用xpath去实现

- **1、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()
```

- **2、代码实现**

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

3、完整代码实现

```
import requests
from lxml import etree
```

```python
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
    try:
        res = requests.get(url,headers=self.headers)
        res.encoding = 'utf-8'
        html = res.text
        self.parse_page(html)
    except Exception as e:
        print('Error')
        self.get_page(url)

  # 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')
    item = {}
    # 依次遍历每个节点对象,提取数据
    if dd_list:
        for dd in dd_list:
          # 电影名称
          name_list = dd.xpath('.//p/a/@title')
          item['name'] = [ name_list[0].strip() if name_list else None ][0]
          # 电影主演
          star_list = dd.xpath('.//p[@class="star"]/text()')
          item['star'] = [ star_list[0].strip() if star_list else None ][0]
          # 上映时间
          time_list  = dd.xpath('.//p[@class="releasetime"]/text()')
          item['time'] = [ time_list[0].strip() if time_list else None ]

          print(item)
    else:
        print('No Data')

  # 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()
```

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
   //div[@class="t_con cleafix"]/div/div/div/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
import time
import random

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html函数
```

```python
def get_html(self,url):
    try:
        res = requests.get(url=url,headers=self.headers)
        res.encoding = 'utf-8'
        html = res.text

        return html
    except Exception as e:
        self.get_html(url)

# 解析html函数
def xpath_func(self,xpath_bds,html):
    parse_html = etree.HTML(html)
    r_list = parse_html.xpath(xpath_bds)

    return r_list


# 一级页面:获取帖子链接,最终搞定所有图片下载
# 还记得吗? 多级页面抓取所有数据都在一级页面中搞定!!!
def get_tlink(self,url):
  html = self.get_html(url)
  xpath_bds = '//div[@class="t_con cleafix"]/div/div/div/a/@href'
  # tlink_list: ['/p/23234','/p/9032323']
  tlink_list = self.xpath_func(xpath_bds,html)
  # 依次遍历每个帖子链接,搞定所有的图片下载
  if tlink_list:
      for tlink in tlink_list:
        t_url = 'http://tieba.baidu.com' + tlink
        # 提取图片链接并保存
        self.get_image(t_url)
        time.sleep(random.randint(1,3))
  else:
      print('No Data')

# 获取图片链接
def get_image(self,t_url):
  html = self.get_html(t_url)
  # 提取图片链接
  xpath_bds = '//*[@class="d_post_content j_d_post_content  clearfix"]/img/@src'
  imglink_list = self.xpath_func(xpath_bds,html)

  for imglink in imglink_list:
    html_content = requests.get(imglink,headers=self.headers).content
    filename = imglink[-10:]
    with open(filename,'wb') as f:
        f.write(html_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
from lxml import etree
import os

class NoteSpider(object):
    def __init__(self):
        self.url = 'http://code.tarena.com.cn/AIDCode/aid1904/14-redis/'
        self.headers = {'User-Agent':'Mozilla/5.0'}
        self.auth = ('tarenacode','code_2013')

    # 获取
    def get_html(self):
        html = requests.get(url=self.url,auth=self.auth,headers=self.headers).text
        return html

    # 解析提取数据 + 把笔记压缩包下载完成
    def parse_page(self):
        html = self.get_html()
        xpath_bds = '//a/@href'
        parse_html = etree.HTML(html)
        # r_list : ['../','day01','day02','redis_day01.zip']
        r_list = parse_html.xpath(xpath_bds)
        for r in r_list:
            if r.endswith('zip') or r.endswith('rar'):
                print(r)

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

思考：爬取具体的笔记文件?

```
import requests
from lxml import etree
import os

class NoteSpider(object):
    def __init__(self):
        self.url = 'http://code.tarena.com.cn/AIDCode/aid1904/14-redis/'
        self.headers = {'User-Agent':'Mozilla/5.0'}
        self.auth = ('tarenacode','code_2013')

    # 获取
    def get_html(self):
        html = requests.get(url=self.url,auth=self.auth,headers=self.headers).text
        return html

    # 解析提取数据 + 把笔记压缩包下载完成
```

```
    def parse_page(self):
        html = self.get_html()
        xpath_bds = '//a/@href'
        parse_html = etree.HTML(html)
        # r_list : ['../','day01','day02','redis_day01.zip']
        r_list = parse_html.xpath(xpath_bds)
        for r in r_list:
            if r.endswith('zip') or r.endswith('rar'):
                file_url = self.url + r
                self.save_files(file_url,r)

    def save_files(self,file_url,r):
        html_content = requests.get(file_url,headers=self.headers,auth=self.auth).content
        # 判断保存路径是否存在
        directory = '/home/tarena/AID1904/redis/'
        filename = directory + r
        if not os.path.exists(directory):
            os.makedirs(directory)

        with open(filename,'wb') as f:
            f.write(html_content)
            print(r,'下载成功')

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

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://112.85.164.220:9999',
    'https':'https://112.85.164.220:9999'
}
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
from lxml import etree
from fake_useragent import UserAgent


class GetProxyIP(object):
    def __init__(self):
        self.url = 'https://www.xicidaili.com/nn/'
        self.test_url = 'http://www.baidu.com/'

```

```python

    # 生成随机的User-Agent
    def get_random_header(self):
        # 创建User-Agent对象
        try:
            ua = UserAgent()
            headers = {'User-Agent':ua.random}
            return headers
        except Exception as e:
            self.get_random_header()


    # 从西刺代理网站上获取随机的代理IP
    def get_ip_list(self):
        headers = self.get_random_header()
        # 访问西刺代理网站国内高匿代理，找到所有的tr节点对象
        html = requests.get(self.url, headers=headers).content.decode('utf-8')
        parse_html = etree.HTML(html)
        # 基准xpath，匹配每个代理IP的节点对象列表
        ipobj_list = parse_html.xpath('//tr')

        # 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、... ...）
        with open('ip_port.txt','a') as f:
            for ip in ipobj_list[1:]:
                ip_info = ip.xpath('./td[2]/text()')[0]
                port_info = ip.xpath('./td[3]/text()')[0]
                ip_port = ip_info + ':' + port_info + '\n'

                proxies = {
                    'http': 'http://' + ip_info + ':' + port_info,
                    'https': 'https://' + ip_info + ':' + port_info
                }

                if self.test_proxy_ip(proxies):
                    f.write(ip_port)
                    print(ip_port,'success')
                else:
                    print(ip_port,'failed')

    # 测试抓取的代理IP是否可用
    def test_proxy_ip(self,proxies):
        try:
            res =
requests.get(self.test_url,headers=self.get_random_header(),proxies=proxies,timeout=3)
            if res.status_code == 200:
                return True
        except Exception as e:
            return False


if __name__ == '__main__':
    spider = GetProxyIP()
    spider.get_ip_list()
```

今日作业

电影天堂（xpath）

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
```