Day10回顾

分布式爬虫原理及实现

```
# 原理
多台主机共享1个爬取队列
# 实现
scrapy_redis
```

分布式爬虫配置

```
1、完成非分布式scrapy爬虫项目
2、settings.py中指向新的调度器、去重机制及redis服务器IP地址及端口号
3、设置管道 - MySQL、MongoDB或Redis
```

redis_key分布式爬虫配置

```
1、完成非分布式scrapy爬虫项目 - 不能重写 start_requests()
2、settings.py中指向新的调度器、去重机制及redis服务器的IP地址及端口号
3、设置管道 - MySQL、MongoDB或Redis
4、爬虫文件使用RedisSpider类
   1、去掉start_urls
   2、添加redis_key : redis_key = "name:spider"
5、部署到多台服务器
6、连接redis, 执行 LPUSH name:spider First_URL
```

机器视觉

```
1、OCR
2、tesseract-ocr
3、pytesseract
```

Fiddler+Browser配置

```
1、Fiddler端 - 安装根证书、只抓浏览器、设置端口
2、Browser端 - 安装代理插件(SwithyOmega)并创建切换新代理
```

Fiddler+Phone配置

```
1、Fiddler端 - 安装根证书、抓所有进程、设置端口、允许远程
2、Phone端 - 修改网络、下载并安装证书
```

爬虫总结

```
# 1、什么是爬虫
  爬虫是请求网站并提取数据的自动化程序

# 2、robots协议是什么
  爬虫协议或机器人协议,网站通过robots协议告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取

# 3、爬虫的基本流程
  1、请求得到响应
  2、解析
  3、保存数据

# 4、请求
  1、urllib
  2、requests
  3、scrapy

# 5、解析
  1、re正则表达式
  2、lxml+xpath解析
  3、json解析模块

# 6、selenium+browser

# 7、常见反爬策略
  1、Headers ： 最基本的反爬手段，一般被关注的变量是UserAgent和Referer，可以考虑使用浏览器中
  2、UA ： 建立User-Agent池,每次访问页面随机切换
  3、拉黑高频访问IP
     数据量大用代理IP池伪装成多个访问者,也可控制爬取速度
  4、Cookies
     建立有效的cookie池，每次访问随机切换
  5、验证码
    验证码数量较少可人工填写
    图形验证码可使用tesseract识别
    其他情况只能在线打码、人工打码和训练机器学习模型
  6、动态生成
    一般由js动态生成的数据都是向特定的地址发get请求得到的，返回的一般是json
  7、签名及js加密
    一般为本地JS加密,查找本地JS文件,分析,或者使用execjs模块执行JS
  8、js调整页面结构
  9、js在响应中指向新的地址

# 8、scrapy框架的运行机制

# 9、分布式爬虫的原理
  多台主机共享一个爬取队列
```

Day11笔记

移动端app数据抓取 - 浏览器F12

有道翻译手机版破解案例

```
import requests
from lxml import etree


word = input('请输入要翻译的单词:')
url = 'http://m.youdao.com/translate'
data = {
    'inputtext': word,
    'type': 'AUTO',
}

html = requests.post(url,data=data).text
parse_html = etree.HTML(html)
result = parse_html.xpath('//ul[@id="translateResult"]/li/text()')[0]

print(result)
```

途牛旅游

目标

```
完成途牛旅游爬取系统，输入出发地、目的地，输入时间，抓取热门景点信息及相关评论
```

地址

```
1、地址：http://www.tuniu.com/
2、热门 - 搜索
3、选择：相关目的地、出发城市、出游时间（出发时间和结束时间）点击确定
4、示例地址如下：
http://s.tuniu.com/search_complex/whole-sh-0-%E7%83%AD%E9%97%A8/list-a{触发时间}_{结束时间}-{出发城市}-{相关目的地}/
```

项目实现

- **1、创建项目**

```
scrapy startproject Tuniu
cd Tuniu
scrapy genspider tuniu tuniu.com
```

- **2、定义要抓取的数据结构 - items.py**

```
    # 一级页面
    # 标题 + 链接 + 价格 + 满意度 + 出游人数 + 点评人数 + 推荐景点 + 供应商
    title = scrapy.Field()
    link = scrapy.Field()
    price = scrapy.Field()
    satisfaction = scrapy.Field()
    travelNum = scrapy.Field()
    reviewNum = scrapy.Field()
    recommended = scrapy.Field()
    supplier = scrapy.Field()

    # 二级页面
    # 优惠券 + 产品评论
    coupons = scrapy.Field()
    cp_comments = scrapy.Field()
```

- **3、爬虫文件数据分析与提取**

页面地址分析

```
http://s.tuniu.com/search_complex/whole-sh-0-热门/list-a20190828_20190930-l200-m3922/
# 分析
list-a{出发时间_结束时间-出发城市-相关目的地}/
# 如何解决?
提取  出发城市及目的地城市的字典,key为城市名称,value为其对应的编码

# 提取字典，定义config.py存放
```

代码实现

```
# -*- coding: utf-8 -*-
import scrapy
from ..config import *
from ..items import TuniuItem
import json

class TuniuSpider(scrapy.Spider):
    name = 'tuniu'
    allowed_domains = ['tuniu.com']

    def start_requests(self):
        s_city = input('出发城市:')
        d_city = input('相关目的地:')
        start_time = input('出发时间(20190828):')
        end_time = input('结束时间(例如20190830):')
        s_city = src_citys[s_city]
        d_city = dst_citys[d_city]

```

```
        url = 'http://s.tuniu.com/search_complex/whole-sh-0-%E7%83%AD%E9%97%A8/list-a{}_{}-{}-{}'.format(start_time,end_time,s_city, d_city)
        yield scrapy.Request(url, callback=self.parse)

    def parse(self, response):
        # 提取所有景点的li节点信息列表
        items = response.xpath('//ul[@class="thebox clearfix"]/li')

        for item in items:
            # 此处是否应该在for循环内创建?
            tuniuItem = TuniuItem()
            # 景点标题 + 链接 + 价格
            tuniuItem['title'] = item.xpath('.//span[@class="main-tit"]/@name').get()
            tuniuItem['link'] = 'http:' + item.xpath('./div/a/@href').get()
            tuniuItem['price'] = int(item.xpath('.//div[@class="tnPrice"]/em/text()').get())
            # 判断是否为新产品
            isnews = item.xpath('.//div[@class="new-pro"]').extract()
            if not len(isnews):
                # 满意度 + 出游人数 + 点评人数
                tuniuItem['satisfaction'] = item.xpath('.//div[@class="comment-satNum"]//i/text()').get()
                tuniuItem['travelNum'] = item.xpath('.//p[@class="person-num"]/i/text()').get()
                tuniuItem['reviewNum'] = item.xpath('.//p[@class="person-comment"]/i/text()').get()
            else:
                tuniuItem['satisfaction'] = '新产品'
                tuniuItem['travelNum'] = '新产品'
                tuniuItem['reviewNum'] = '新产品'
            # 包含景点+供应商
            tuniuItem['recommended'] = item.xpath('.//span[@class="overview-scenery"]/text()').extract()
            tuniuItem['supplier'] = item.xpath('.//span[@class="brand"]/span/text()').extract()

            yield scrapy.Request(tuniuItem['link'], callback=self.item_info, meta={'item': tuniuItem})

    # 解析二级页面
    def item_info(self, response):
        tuniuItem = response.meta['item']
        # 优惠信息
        coupons = ','.join(response.xpath('//div[@class="detail-favor-coupon-desc"]/@title').extract())
        tuniuItem['coupons'] = coupons

        # 想办法获取评论的地址
        # 产品点评 + 酒店点评 + 景点点评
        productId = response.url.split('/')[-1]
        # 产品点评
        cpdp_url = 'http://www.tuniu.com/papi/tour/comment/product?productId={}'.format(productId)

        yield scrapy.Request(cpdp_url, callback=self.cpdp_func, meta={'item': tuniuItem})

    # 解析产品点评
```

```
    def cpdp_func(self, response):
        tuniuItem = response.meta['item']

        html = json.loads(response.text)
        comment = {}
        for s in html['data']['list']:
            comment[s['realName']] = s['content']

        tuniuItem['cp_comments'] = comment

        yield tuniuItem
```

- **4、管道文件处理 - pipelines.py**

```
print(dict(item))
```

- **5、设置settings.py**

出发城市和目的地城市的编号如何获取？ - tools.py

```
# 出发城市
# 基准xpath表达式
//*//*[@id="niuren_list"]/div[2]/div[1]/div[2]/div[1]/div/div[1]/dl/dd/ul/li[contains(@class,"filter_input")]/a
name : ./text()
code : ./@href  [0].split('/')[-1].split('-')[-1]

# 目的地城市
# 基准xpath表达式
//*[@id="niuren_list"]/div[2]/div[1]/div[2]/div[1]/div/div[3]/dl/dd/ul/li[contains(@class,"filter_input")]/a
name :  ./text()
code :  ./@href  [0].split('/')[-1].split('-')[-1]
```

代码实现

```
import requests
from lxml import etree

url = 'http://s.tuniu.com/search_complex/whole-sh-0-%E7%83%AD%E9%97%A8/'
headers = {'User-Agent':'Mozilla/5.0'}

html = requests.get(url,headers=headers).text
parse_html = etree.HTML(html)
# 获取出发地字典
# 基准xpath
li_list = parse_html.xpath('//*[@id="niuren_list"]/div[2]/div[1]/div[2]/div[1]/div/div[3]/dl/dd/ul/li[contains(@class,"filter_input")]/a')
src_citys = {}
```

```
dst_citys = {}
for li in li_list:
    city_name_list = li.xpath('./text()')
    city_code_list = li.xpath('./@href')
    if city_name_list and city_code_list:
        city_name = city_name_list[0].strip()
        city_code = city_code_list[0].split('/')[-1].split('-')[-1]
        src_citys[city_name] = city_code
print(src_citys)

# 获取目的地字典
li_list = parse_html.xpath('//*
[@id="niuren_list"]/div[2]/div[1]/div[2]/div[1]/div/div[1]/dl/dd/ul/li[contains(@class,"filter
_input")]/a')
for li in li_list:
    city_name_list = li.xpath('./text()')
    city_code_list = li.xpath('./@href')
    if city_name_list and city_code_list:
        city_name = city_name_list[0].strip()
        city_code = city_code_list[0].split('/')[-1].split('-')[-1]
        dst_citys[city_name] = city_code
print(dst_citys)
```