# 项目实战：51job抓取--scrapy版

## 1、各个页面的地址

https://search.51job.com/jobsearch/search\_result.php?fromJs=1&jobarea=010000%2C00&district=000000&funtype=0000&industrytype=00&issuedate=9&providesalary=99&keyword=python&keywordtype=2&curr\_page=1&lang=c&stype=1&postchannel=0000&workyear=99&cotype=99&degreefrom=99&jobterm=99&companysize=99&lonlat=0%2C0&radius=-1&ord\_field=0&list\_type=0&fromType=14&dibiaoid=0&confirmdate=9

https://search.51job.com/jobsearch/search\_result.php?fromJs=1&jobarea=010000%2C00&district=000000&funtype=0000&industrytype=00&issuedate=9&providesalary=99&keyword=python&keywordtype=2&curr\_page=2&lang=c&stype=1&postchannel=0000&workyear=99&cotype=99&degreefrom=99&jobterm=99&companysize=99&lonlat=0%2C0&radius=-1&ord\_field=0&list\_type=0&fromType=14&dibiaoid=0&confirmdate=9

https://search.51job.com/jobsearch/search\_result.php?fromJs=1&jobarea=010000%2C00&district=000000&funtype=0000&industrytype=00&issuedate=9&providesalary=99&keyword=python&keywordtype=2&curr\_page=3&lang=c&stype=1&postchannel=0000&workyear=99&cotype=99&degreefrom=99&jobterm=99&companysize=99&lonlat=0%2C0&radius=-1&ord\_field=0&list\_type=0&fromType=14&dibiaoid=0&confirmdate=9

简化后的地址

第一页：

<http://search.51job.com/jobsearch/search_result.php?fromJs=1&jobarea=010000%2C00&keyword=python&curr_page=1>

第二页

<http://search.51job.com/jobsearch/search_result.php?fromJs=1&jobarea=010000%2C00&keyword=python&curr_page=2>

第三页

<http://search.51job.com/jobsearch/search_result.php?fromJs=1&jobarea=010000%2C00&keyword=python&curr_page=3>

## 2、要抓取的数据

需求分析：我们要抓取51jop上的python的招聘信息：

url（职位的连接）

title（招聘标题）

location（地点）

company\_name （公司名称）

salary（薪水）

company\_info(公司信息)

experience(工作经验)

job\_info（职位信息）

address(联系方式)





## 3、51job抓取--scrapy\_redis版

3.1新建项目，创建基础爬虫，分析xpath，

3.2写爬虫.py

# -\*- coding: utf-8 -\*-  
import scrapy  
from scrapy\_redis.spiders import RedisSpider  
from job51.items import Job51Item  
  
# class JobSpider(RedisSpider):后期要改为scrapy-redis要更改的地方-1  
class JobSpider(scrapy.Spider):  
 name = 'job'  
 allowed\_domains = ['51job.com']  
 # redis\_key = 'jobspider:start\_urls' 要改的-2  
 page = 1  
 url = "https://search.51job.com/jobsearch/search\_result.php?fromJs=1&jobarea=010000%2C00&keyword=python&curr\_page="  
 start\_urls = [url + str(page)]  
  
 def parse(self, response):  
 print("--------------------------------", response.url)  
 for url in response.xpath('//div[@class="el"]/p/span/a/@href').extract():  
 yield scrapy.Request(url=url, callback=self.real\_data)  
 # if self.page < 132:  
 # self.page += 1  
 # new\_url = 'https://search.51job.com/jobsearch/search\_result.php?fromJs=1&jobarea=010000%2C00&keyword=python&curr\_page=' + str(  
 # self.page)  
 # yield scrapy.Request(url=new\_url, callback=self.parse)  
 next\_url = response.xpath('//li[@class="bk"][last()]/a/@href').extract\_first()  
 print("next\_url==", next\_url)  
 if next\_url:  
 yield scrapy.Request(url=next\_url, callback=self.parse)  
 else:  
 print("请求结束")  
  
 def real\_data(self, response):  
 item = Job51Item()  
 item['url'] = response.url  
 item['title'] = response.xpath('//h1/@title').extract\_first()  
 # print(item['title'])  
 item['location'] = response.xpath('//div[@class="cn"]/p[2]/text()[1]').extract\_first().strip()  
 item['company\_name'] = response.xpath('//div[@class="cn"]/p/a[1]/text()').extract\_first().strip()  
 item['salary'] = response.xpath('//div[@class="cn"]/strong/text()').extract\_first()  
 item['company\_info'] = response.xpath('//div[@class="com\_tag"]/p/text()').extract()  
 item['experience'] = response.xpath('//div[@class="cn"]/p[2]/text()[2]').extract\_first().strip()  
 job\_info = response.xpath(  
 '//div[@class="bmsg job\_msg inbox"]/p/text()|//div[@class="bmsg job\_msg inbox"]/text()').extract()  
 item['job\_info'] = "".join(job\_info).strip()  
 address = response.xpath('//div[@class="bmsg inbox"]/p[@class="fp"]/text()').extract()  
 item['address'] = "".join(address).replace('\r','').replace('\n','').replace('\t','')  
 # print(item['address'])  
 yield item

3.3items.py

import scrapy  
  
  
class Job51Item(scrapy.Item):  
 # define the fields for your item here like:  
 title = scrapy.Field()  
 url = scrapy.Field()  
 location = scrapy.Field()  
 company\_name = scrapy.Field()  
 salary = scrapy.Field()  
 company\_info = scrapy.Field()  
 experience = scrapy.Field()  
 job\_info = scrapy.Field()  
 address = scrapy.Field()  
  
 crawled = scrapy.Field()  
 spider = scrapy.Field()

3.4setting.py主要配置

# -\*- coding: utf-8 -\*-  
  
# Scrapy settings for job51 project  
#  
# For simplicity, this file contains only settings considered important or  
# commonly used. You can find more settings consulting the documentation:  
#  
# https://doc.scrapy.org/en/latest/topics/settings.html  
# https://doc.scrapy.org/en/latest/topics/downloader-middleware.html  
# https://doc.scrapy.org/en/latest/topics/spider-middleware.html  
  
BOT\_NAME = 'job51'  
  
SPIDER\_MODULES = ['job51.spiders']  
NEWSPIDER\_MODULE = 'job51.spiders'  
  
# Obey robots.txt rules  
ROBOTSTXT\_OBEY = True  
  
DOWNLOADER\_MIDDLEWARES = {  
 # 'Douban.middlewares.DoubanDownloaderMiddleware': 543,  
 # 配置随机的浏览器  
 'job51.middlewares.RandomMiddleware': 543,  
 # 配置随机的代理  
 # 'mySpider.middlewares.RandomProxyIpMiddleware': 544,  
 # 把系统的默认关闭掉（否则不起作用）  
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,  
}

后期改redis后要改的地方-3  
# -------------------------------------------------------------------  
# 不用原来scrapy的去重了，使用自定义的去重过滤器  
DUPEFILTER\_CLASS = "scrapy\_redis.dupefilter.RFPDupeFilter"  
  
# 使用自己的调度器  
SCHEDULER = "scrapy\_redis.scheduler.Scheduler"  
  
# 是否可以暂停，是否可以继续爬取  
SCHEDULER\_PERSIST = True  
# 优先级队列  
SCHEDULER\_QUEUE\_CLASS = "scrapy\_redis.queue.SpiderPriorityQueue"  
  
# 普通队列  
# SCHEDULER\_QUEUE\_CLASS = "scrapy\_redis.queue.SpiderQueue"  
  
# 栈  
# SCHEDULER\_QUEUE\_CLASS = "scrapy\_redis.queue.SpiderStack"  
  
ITEM\_PIPELINES = {  
 # 加上爬取时间和爬虫的名称  
 'job51.pipelines.ExamplePipeline': 300,  
  
 # 调用系统默认的RedisPipeline，默认把数据存到redis  
 'scrapy\_redis.pipelines.RedisPipeline': 400,  
}  
  
LOG\_LEVEL = 'DEBUG'  
# Introduce an artifical delay to make use of parallelism. to speed up the  
# crawl.  
# DOWNLOAD\_DELAY = 1  
  
# 配置主机信息：存储请求request队列，指纹队列,数据队列  
REDIS\_HOST = "192.168.11.68"  
REDIS\_PORT = 6379

3.5pipeline.py

from datetime import datetime  
  
  
class ExamplePipeline(object):  
  
 #第三个参数是：爬虫MyCrawler类的实例  
 def process\_item(self, item, spider):  
 #添加爬取时间  
 item["crawled"] = datetime.utcnow()  
 #添加爬虫的名称：myspider\_redis+"windows\_afu"  
 item["spider"] = spider.name+"\_\_windows\_llllllwt"  
 return item  
  
  
class Job51Pipeline(object):  
 def process\_item(self, item, spider):  
 return item

3.6把redis的数据写入到MongoDB，注意：一旦写入redis中数据就没了，所以第一次写入后第二次需要重新执行main获取数据存入redis中，才能取到数据

import pymongo  
import pymysql  
import redis  
import json  
  
# mongodb的客户端  
client=pymongo.MongoClient(host='127.0.0.1',port=27017)  
# redis的客户端  
redis\_client=redis.StrictRedis(host='192.168.11.68',port=6379,db=0)  
  
sina=client['job']  
  
sina\_item=sina['job']  
while True:  
 source,data=redis\_client.blpop(['job:items'])  
 print('source===',source)  
 print('data===', data)  
 data=data.decode('utf-8')  
 data=json.loads(data)  
 sina\_item.insert\_one(data)

3.7把redis中数据存到mysql，同样，取出一次后不管是否成功存到mysql，redis中的数据都已经被取出了，没有数据了，需要再次存入redis然后再取数据

import pymysql  
import redis  
import time  
import json  
# 得到redis的数据 然后存入mysql  
  
# mysql的客户端  
client = pymysql.connect(host="127.0.0.1", user="root", password="atguigu",  
 db="scrapy", port=3306, charset='utf8')  
  
cursor = client.cursor()  
  
# redis的客户端  
reids\_client = redis.StrictRedis(host='192.168.11.68', port=6379, db=0)  
  
while True:  
 try:  
 source, data = reids\_client.blpop(["sina\_guide:items"])  
 print("--------------------")  
 # time.sleep(1)  
 print("source==", source)  
 data = data.decode("utf-8")  
 print("data==", data)  
 print("data==", type(data))  
 item = json.loads(data)  
  
 params = [item['parent\_title'], item['sub\_title'], item['sub\_url'], item['tiezi\_path'], item['tiezi\_url'],  
 item['tiezi\_title'], item['crawled'],item['spider']]  
 # 注意插入的字段要和数据库中表的字段相同  
 sql = "INSERT INTO sina2(parent\_title, sub\_title,sub\_url,tiezi\_path,tiezi\_url,tiezi\_title,crawled,spider) VALUES ( %s, %s, %s, %s, %s, %s, %s, %s )"  
 # 执行sql语句  
 cursor.execute(sql, params)  
  
 # 提交事务  
 client.commit()  
 # 字典  
 except Exception as e:  
 print("出错了==", e)

## 项目实战：51job抓取--设置下载中间件（动态User-Agent）和动态代理

4.1Setting中设置

DOWNLOADER\_MIDDLEWARES = {  
 # 'Douban.middlewares.DoubanDownloaderMiddleware': 543,  
 # 配置随机的浏览器  
 'job51.middlewares.RandomMiddleware': 543,  
 # 配置随机的代理  
 # 'mySpider.middlewares.RandomProxyIpMiddleware': 544,  
 # 把系统的默认关闭掉（否则不起作用）  
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,  
}

4.2middleware

from fake\_useragent import UserAgent  
from scrapy import signals  
  
  
class RandomMiddleware(object):  
 # Not all methods need to be defined. If a method is not defined,  
 # scrapy acts as if the spider middleware does not modify the  
 # passed objects.  
 def \_\_init\_\_(self,crawler):  
 super(RandomMiddleware,self).\_\_init\_\_()  
 self.us=UserAgent  
 @classmethod  
 def from\_crawler(cls, crawler):  
 # This method is used by Scrapy to create your spiders.  
 return cls(crawler)  
  
 def process\_request(self,request,spider):  
 # 随机user-agent  
 user\_agent=self.us.random  
 request.headers.setdefault=("User-Agent",user\_agent)  
 # 随机代理  
 request.meta['proxy']='http://118.190.95.43:9001'  
 return None

## 项目实战：51job抓取--数据直接存入mongodb

5.1创建项目

5.2setting

MONGO\_HOST = '127.0.0.1'  
# mongo 端口  
MONGO\_PORT = 27017  
# mongo 数据存放数据库库名称  
MONGO\_DBNAME = "jobmongo"  
# mongo 数据存放的表名称  
MONGO\_SHEETNAME = "job2"  
# Disable cookies (enabled by default)  
COOKIES\_ENABLED = False  
  
ITEM\_PIPELINES = {  
 # 'python\_job.pipelines.DoubanPipeline': 300,  
 'jobmongo.pipelines.JobmongoPipeline': 300,  
}

5.3pipeline

import json  
import pymongo  
from jobmongo.settings import MONGO\_HOST,MONGO\_PORT, MONGO\_DBNAME, MONGO\_SHEETNAME  
  
  
class JobmongoPipeline(object):  
 def open\_spider(self,spider):  
 self.file=open('job.json','w',encoding='utf-8')  
 host = MONGO\_HOST  
 port = MONGO\_PORT  
 db\_name = MONGO\_DBNAME  
 sheet\_name = MONGO\_SHEETNAME  
 client=pymongo.MongoClient(host=host,port=port)  
 job=client[db\_name]  
 self.sheetname=job[sheet\_name]  
  
 def close\_spider(self,spider):  
 self.file.close()  
  
 def process\_item(self, item, spider):  
 dict\_item = dict(item)  
 self.file.write(json.dumps(dict\_item, ensure\_ascii=False) + "\n")  
 # 插入数据  
 self.sheetname.insert\_one(dict\_item)  
 return item

5.4

import scrapy  
from jobmongo.items import JobmongoItem  
  
  
class Job2Spider(scrapy.Spider):  
 name = 'job2'  
 allowed\_domains = ['51job.com']  
 page = 1  
 url = "https://search.51job.com/jobsearch/search\_result.php?fromJs=1&jobarea=010000%2C00&keyword=python&curr\_page="  
 start\_urls = [url + str(page)]  
  
 def parse(self, response):  
 print("--------------------------------", response.url)  
 for url in response.xpath('//div[@class="el"]/p/span/a/@href').extract():  
 yield scrapy.Request(url=url, callback=self.real\_data)  
 next\_url = response.xpath('//li[@class="bk"][last()]/a/@href').extract\_first()  
 print("next\_url==", next\_url)  
 if next\_url:  
 yield scrapy.Request(url=next\_url, callback=self.parse)  
 else:  
 print("请求结束")  
  
 def real\_data(self, response):  
 item = JobmongoItem()  
 item['url'] = response.url  
 item['title'] = response.xpath('//h1/@title').extract\_first()  
 # print(item['title'])  
 item['location'] = response.xpath('//div[@class="cn"]/p[2]/text()[1]').extract\_first().strip()  
 item['company\_name'] = response.xpath('//div[@class="cn"]/p/a[1]/text()').extract\_first().strip()  
 item['salary'] = response.xpath('//div[@class="cn"]/strong/text()').extract\_first()  
 item['company\_info'] = response.xpath('//div[@class="com\_tag"]/p/text()').extract()  
 item['experience'] = response.xpath('//div[@class="cn"]/p[2]/text()[2]').extract\_first().strip()  
 job\_info = response.xpath(  
 '//div[@class="bmsg job\_msg inbox"]/p/text()|//div[@class="bmsg job\_msg inbox"]/text()').extract()  
 item['job\_info'] = "".join(job\_info).strip()  
 address = response.xpath('//div[@class="bmsg inbox"]/p[@class="fp"]/text()').extract()  
 item['address'] = "".join(address).replace('\r', '').replace('\n', '').replace('\t', '')  
 # print(item['address'])  
 yield item

5.5

import scrapy  
  
  
class JobmongoItem(scrapy.Item):  
 # define the fields for your item here like:  
 # name = scrapy.Field()  
 title = scrapy.Field()  
 url = scrapy.Field()  
 location = scrapy.Field()  
 company\_name = scrapy.Field()  
 salary = scrapy.Field()  
 company\_info = scrapy.Field()  
 experience = scrapy.Field()  
 job\_info = scrapy.Field()  
 address = scrapy.Field()

## 6、项目实战：51job抓取--数据直接存入mysql数据库

**该案例晚上，挨个检查**