# 爬腾讯招聘信息

## crawl爬虫说明

**Spider**类与CrawlSpider类，在写代码中的**差异**：

1. 仅仅是spiders目录下的爬虫模块有差异
2. settings/items/pipelines几个模块并无差异

将“**Spider**类”爬虫，改造成“CrawlSpider类”爬虫

只须修改“spiders目录下的爬虫模块”

## 创建爬虫项目

**scrapy startproject tencent\_spider**

## 定义item

# -\*- coding: utf-8 -\*-

# Define here the models for your scraped items

#

# See documentation in:

# https://doc.scrapy.org/en/latest/topics/items.html

import scrapy

import scrapy

*class* TencentItem(*scrapy*.*Item*):

# define the fields for your item here like:

# 职位名

position\_name = scrapy.Field()

# 详情连接

position\_link = scrapy.Field()

# 职位类别

position\_type = scrapy.Field()

# 招聘人数

people\_num = scrapy.Field()

# 工作地点

work\_address = scrapy.Field()

# 发布时间

publish\_time = scrapy.Field()

## 创建深度爬虫模块

用crawl模板创建爬虫模块

**cd tencent\_spider/**

**scrapy genspider -t crawl tencent hr.tencent.com**

# -\*- coding: utf-8 -\*-

import scrapy

# 导入CrawlSpider类和Rule

from scrapy.spiders import CrawlSpider, Rule

# 导入链接规则匹配类，用来提取符合规则的连接

from scrapy.linkextractors import LinkExtractor

from tencent\_spider.items import TencentItem

*class* TencentSpider(*CrawlSpider*):

name = 'tencent'

allowed\_domains = ['hr.tencent.com']

start\_urls = ['http://hr.tencent.com/position.php?&start=0#a']

# # 广度上：限制地理位置

# start\_urls = [

# # 深圳招聘

# "http://hr.tencent.com/position.php?keywords=&tid=0&lid=2218",

# # 北京招聘

# "http://hr.tencent.com/position.php?keywords=&tid=0&lid=2156",

# # 上海招聘

# "http://hr.tencent.com/position.php?keywords=&tid=0&lid=2175"

# ]

# Response里链接的提取规则，返回的符合匹配规则的链接匹配对象的列表

# allow:一个正则表达式（或者正则表达式列表）

page\_link = LinkExtractor(*allow*=*r*"start=\d+")

rules = [

# 获取这个列表里的链接，依次发送请求，并且继续跟进，调用指定回调函数处理

Rule(page\_link, *callback*='parse\_item', *follow*=True),

]

# 指定的回调函数

*def* parse\_item(*self*, *response*):

for each in response.xpath("//tr[@class='even'] | //tr[@class='odd']"):

item = TencentItem()

# 职位名称

item['position\_name'] = each.xpath("./td[1]/a/text()").extract()[0]

# 详情连接

item['position\_link'] = each.xpath("./td[1]/a/@href").extract()[0]

# 职位类别

item['position\_type'] = each.xpath("./td[2]/text()").extract()[0]

# 招聘人数

item['people\_num'] = each.xpath("./td[3]/text()").extract()[0]

# 工作地点

item['work\_address'] = each.xpath("./td[4]/text()").extract()[0]

# 发布时间

item['publish\_time'] = each.xpath("./td[5]/text()").extract()[0]

yield item

## 构建pipeline

# -\*- coding: utf-8 -\*-

# Define your item pipelines here

#

# Don't forget to add your pipeline to the ITEM\_PIPELINES setting

# See: https://doc.scrapy.org/en/latest/topics/item-pipeline.html

import json

# # 保存数据为json格式

# class TencentPipeline(object):

# def \_\_init\_\_(self):

# self.filename = open("tencentHR.json", "w")

#

# def process\_item(self, item, spider):

# text = json.dumps(dict(item), ensure\_ascii=False) + "\n"

# self.filename.write(text.encode("utf-8"))

# return item

#

# def close\_spider(self, spider):

# self.filename.close()

# 对字段值做验证，看是否符合要求，不符合的丢弃

# from scrapy.exceptions import DropItem

#

# class PeoplePipeline(object):

#

# def process\_item(self, item, spider):

# if item['people\_num']>0:

# return item

# else:

# raise DropItem("此岗位已经不再招人了，因此这条数据丢掉")

# # 去重

# # 一个用于去重的过滤器，丢弃那些已经被处理过的item。

# # 让我们假设我们的item有一个唯一的id，但是我们spider返回的多个item中包含有相同的id:

# from scrapy.exceptions import DropItem

# # 这里假设字段position\_link值是唯一的，不能重复，如果重复就删除

# class DuplicatesPipeline(object):

#

# def \_\_init\_\_(self):

# self.ids\_seen = set()

#

# def process\_item(self, item, spider):

# if item['position\_name'] in self.ids\_seen:

# raise DropItem("Duplicate item found: %s" % item)

# else:

# self.ids\_seen.add(item['position\_name'])

# return item

# 保存数据到mongo数据库

# 数据库名：items

# 集合名：scrapy\_items

import pymongo

*class* MongoPipeline(*object*):

collection\_name = 'scrapy\_items'

*def* \_\_init\_\_(*self*, *mongo\_uri*, *mongo\_db*):

self.mongo\_uri = mongo\_uri

self.mongo\_db = mongo\_db

@*classmethod*

*def* from\_crawler(*cls*, *crawler*):

return cls(

*mongo\_uri*=crawler.settings.get('MONGO\_URI'),

*mongo\_db*=crawler.settings.get('MONGO\_DATABASE', 'items')

)

*def* open\_spider(*self*, *spider*):

self.client = pymongo.MongoClient(self.mongo\_uri)

self.db = self.client[self.mongo\_db]

*def* close\_spider(*self*, *spider*):

self.client.close()

*def* process\_item(*self*, *item*, *spider*):

self.db[self.collection\_name].insert(*dict*(item))

return item

## 设置settings文件

# -\*- coding: utf-8 -\*-

BOT\_NAME = 'tencent\_spider'

SPIDER\_MODULES = ['tencent\_spider.spiders']

NEWSPIDER\_MODULE = 'tencent\_spider.spiders'

# 保存日志信息的文件名

# LOG\_FILE = "tencentlog.log"

# 保存日志等级，低于|等于此等级的信息都被保存

LOG\_LEVEL = "DEBUG"

# mongo数据库

MONGO\_URI = "mongodb://ip:27017"

# MONGO\_DATABASE=

# Crawl responsibly by identifying yourself (and your website) on the user-agent

USER\_AGENT = "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36"

# Obey robots.txt rules

ROBOTSTXT\_OBEY = False

# Disable cookies (enabled by default)

COOKIES\_ENABLED = False

# Configure item pipelines

# See https://doc.scrapy.org/en/latest/topics/item-pipeline.html

ITEM\_PIPELINES = {

# 'tencent\_spider.pipelines.TencentPipeline': 300,

# 'tencent\_spider.pipelines.PeoplePipeline': 400,

# 'tencent\_spider.pipelines.DuplicatesPipeline': 500,

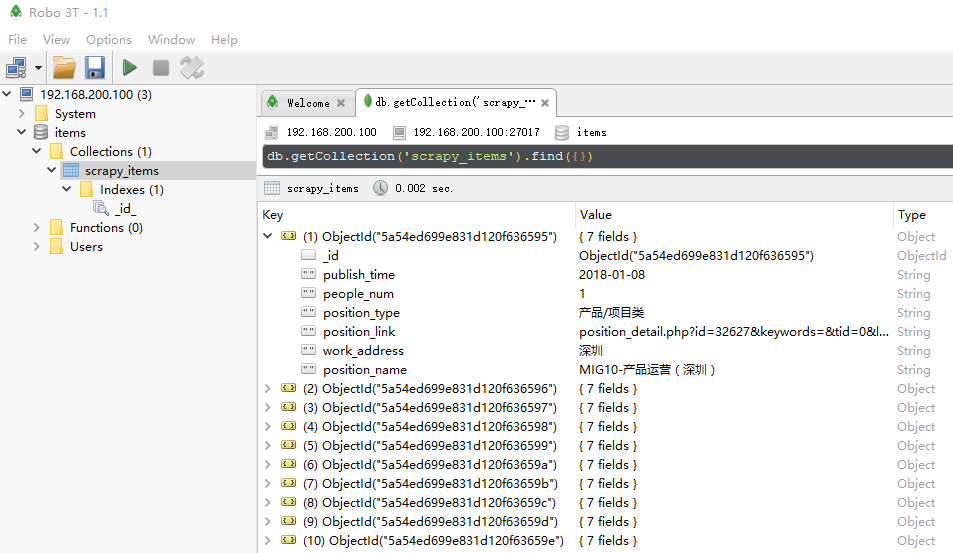
'tencent\_spider.pipelines.MongoPipeline': 600,

}

# 启动爬虫

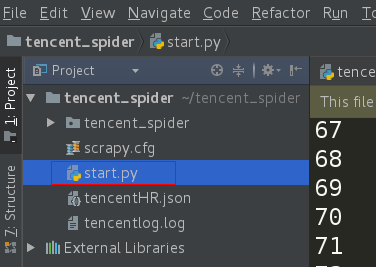
scrapy crawl tencent

# 查看存储到MongoDB中的数据



# PyCharm中启动爬虫

## +编辑start.py



from scrapy import cmdline

cmdline.execute("scrapy crawl tencent".split())

## Edit Configuration

