# 第五章 pythonscrapy高级进阶

本节所讲内容：

5.1 Scrapy response响应介绍

5.2 Scrapy item与pipline

5.3 Scrapy 项目实战之爬取哔哩哔哩视频评论

## 5.1 Scrapy response响应介绍

### 5.1.1 Response分类

TextResponse，HtmlResponse，Xmlresponse

我们默认将scrapy请求到的内容放到parse当中的response参数当中，但是在之前的学习当中我们没有对爬取下来的数据进行详细的处理，我们接下来学习对爬虫数据的详细处理。

Response可用的方法

# 'body', http响应的正文，字节

# 'body\_as\_unicode', 字符串类型的响应

# 'copy', 复制

# 'css', 以css进行匹配

# 'encoding',加码

# 'headers', 响应头部

# 'meta', 响应处理的参数

# 'replace', 替换

# 'request', 产生http请求的request对象

# 'selector', scrapy 的字符匹配器

# 'status', 状态码 200 500 300

# 'text', 文本形式的响应内容

# 'url', http响应的地址

# 'urljoin', 构造绝对url

# 'xpath' 以xpath进行匹配

Response当中最主要是返回的内容，那么我们如何妥善的处理这些内容就摆到面前了。

那么我们来认识scrapy的匹配核心 selector

Beautifulsoup 比较方便，但是解析速度比较慢

Lxml 解析速度比较快

Scrapy集合二者的优点，进行总和。

import scrapy

from scrapy.selector import Selector

class BaiduSpider(scrapy.spiders.Spider):

name = "baidu"

def start\_requests(self):

url = "https://www.baidu.com/"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW65) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"

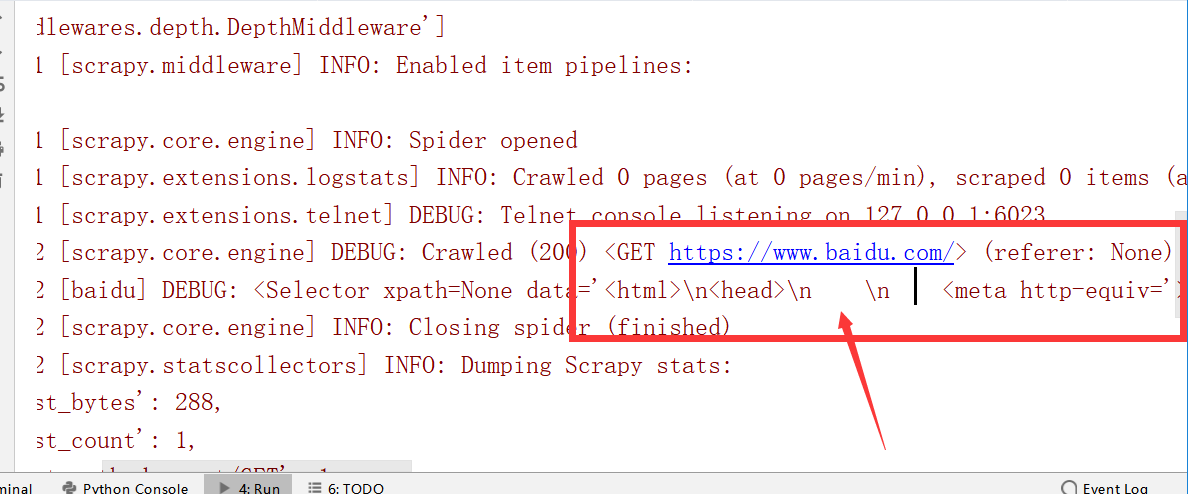
}

yield scrapy.Request(url=url,headers=headers,method='GET')#默认是get方法

def parse(self,response):

select = Selector(response)

self.log(select)



### 5.1.2 Selector对象支持

1、Xpath

在scrapy当中写xpath不会有 text() attrib() tag()这样的方法，我们需要把这些方法写到匹配当中.

|  |  |  |
| --- | --- | --- |
| 表达式 | 描述 | 例子 |
| / | 当前文档的根或者层 | /html/body/div |
| text() | 文本 | /html/body/div/a/text() <p>test</p> |
| @attrib | 属性 | /html/body/div/a/@href |
| \* | 代表所有 | /html/body/\*[@class=’hello’]  /html/body/a/@\* |
| [] | 修饰语 | /html/body/div[5]  /html/body/div[@class=”while”] |

import scrapy

from scrapy.selector import Selector

class BaiduSpider(scrapy.spiders.Spider):

name = "baidu"

def start\_requests(self):

url = "https://www.baidu.com/"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW65) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"

}

yield scrapy.Request(url=url,headers=headers,method='GET')#默认是get方法

def parse(self,response):

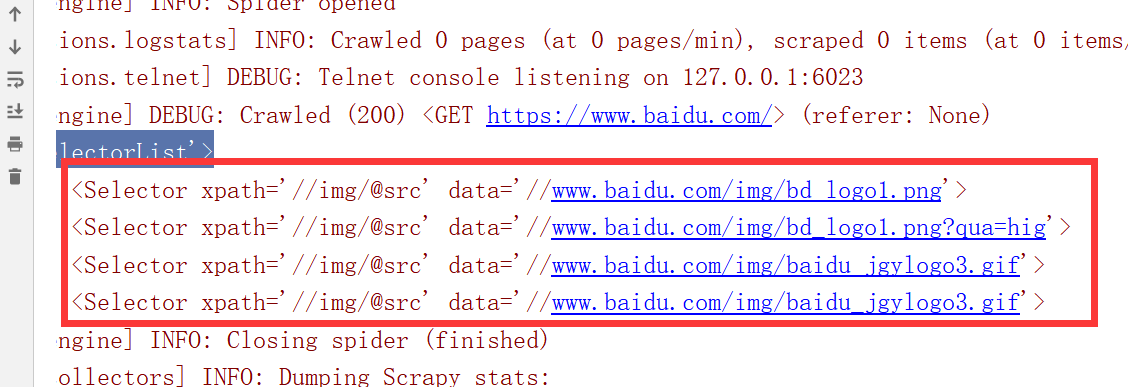
select = Selector(response)

img\_list = select.xpath("//img/@src")

print(type(img\_list))##<class'scrapy.selector.unified.SelectorList'>

for img in img\_list:

self.log(img)



2、css 选择器

|  |  |  |
| --- | --- | --- |
| 表达式 | 描述 | 例子 |
| \* | 所有元素 | \* 所有的标签 |
| Tag | 指定标签 | img 所有的img标签 |
| Tag1,tag2 | 指定多个标签 | img,a img和a标签 |
| Tag1 tage2 (中间是空格) | 下一层标签 | Img a img下的a标签 |
| Attrib = value | 指定属性 | Id = 1 id等于1的标签 |

import scrapy

from scrapy.selector import Selector

class BaiduSpider(scrapy.spiders.Spider):

name = "baidu"

def start\_requests(self):

url = "https://www.baidu.com/"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW65) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"

}

yield scrapy.Request(url=url,headers=headers,method='GET')#默认是get方法

def parse(self,response):

select = Selector(response)

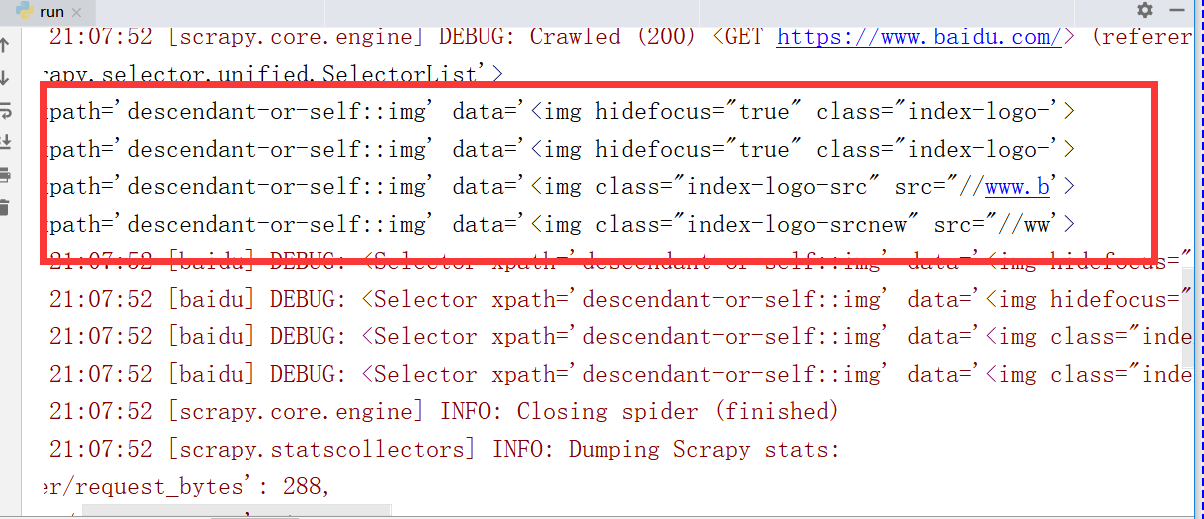
img\_list = select.css("img")

print(type(img\_list))

for img in img\_list:

self.log(img)

print(img)



3、re正则匹配

正则在scrapy当中比较尴尬，他不可以独立用，只可以加在匹配项后面

import scrapy

from scrapy.selector import Selector

class BaiduSpider(scrapy.spiders.Spider):

name = "tupian"

def start\_requests(self):

url = "http://www.gif5.net/"

headers = {

"Referer": "https://www.baidu.com/link?url=dW2F8FcLKkCF62nEciiyg5MStOXGkdlO9gwE0Y5kRzi&wd=&eqid=90fd75130006063b000000035bf5092c",

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW65) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"

}

yield scrapy.Request(url=url,headers=headers,method='GET')#默认是get方法

def parse(self,response):

select = Selector(response)

img\_list = select.xpath("//img/@src")

for img in img\_list:

Img =img.re("(.\*?)gif$")#匹配前面所有

self.log(Img)



5、Selector对象返回字符结果的方法：Extract

import scrapy

from scrapy.selector import Selector

class BaiduSpider(scrapy.spiders.Spider):

name = "baidu"

def start\_requests(self):

url = "https://www.baidu.com/"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW65) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"

}

yield scrapy.Request(url=url,headers=headers,method='GET')#默认是get方法

def parse(self,response):

select = Selector(response)

img\_list = select.xpath("//img/@src")

print(type(img\_list))

for img in img\_list:

Img = img.extract()

self.log(Img)



## 5.2 Scrapy item与pipline

1. items.py ———（是定义scrapy内部数据的文件 ）Field()####type（dict）

2. pipelines.py———（ pipelines.py ：当我们的items被返回的时候，会自动调用我们的pipelines类中process\_item(),需要加到settings.py里面：）

Scrapy有一个巨大的优势，scrapy可以定义数据模型，我们用item可以定义我们的数据模型类，定义的方法类似django的模型，但也有不同。

Scrapy默认会创建一个模型，我们可以在里面定义我们想要的数据模型。

1、scrapy 的item当中所有的字段都可以为Field

2、scrapy当中定义模型是一个类，是类就要遵守Python的语法规则

import scrapy

from scrapy.selector import Selector

from mySpider.items import MyspiderItem

class BaiduSpider(scrapy.spiders.Spider):

name = "baidu"

def start\_requests(self):

url = "https://www.baidu.com/"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW65) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"

}

yield scrapy.Request(url=url,headers=headers,method='GET')#默认是get方法

def parse(self,response):

select = Selector(response)

img\_list = select.xpath("//img/@src")

print(type(img\_list))

for img in img\_list:

item =MyspiderItem()

item["src"] = img.extract()

self.log(item)

Item.py

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

# Define here the models for your scraped items

#

# See documentation in:

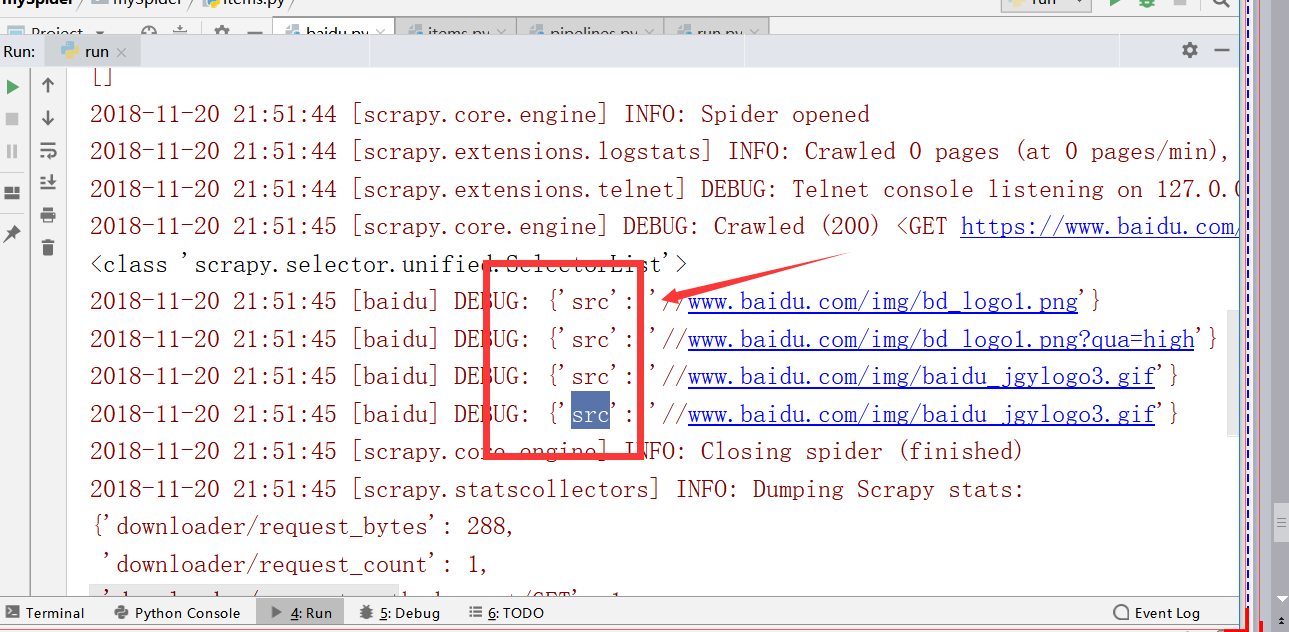
# https://doc.scrapy.org/en/latest/topics/items.html

import scrapy

class MyspiderItem(scrapy.Item):

# define the fields for your item here like:

src = scrapy.Field()



### 5.2.2 piplines的使用

这个时候，我们可以把数据格式化了，但是数据输出我们还需要使用piplines

1、使用piplines的第一步是在settings当中解开piplines的配置

2、 优先级，优先级分为1-1000，越高越先下载

Settings.py #释放pipelines.py

ITEM\_PIPELINES = {

'mySpider.pipelines.MyspiderPipeline': 300, #第一个参数是位置，第二个参数是优先级1-1000，优先级越高越先下载

}

Baidu.py ##生成item对象

import scrapy

from scrapy.selector import Selector

from mySpider.items import MyspiderItem

class BaiduSpider(scrapy.spiders.Spider):

name = "baidu"

def start\_requests(self):

url = "https://www.baidu.com/"

headers = {

"User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW65) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"

}

yield scrapy.Request(url=url,headers=headers,method='GET')#默认是get方法

def parse(self,response):

select = Selector(response)

img\_list = select.xpath("//img/@src")

# print(type(img\_list))

for img in img\_list:

item =MyspiderItem()

item["src"] = img.extract()

self.log(item)

yield item

Pipelines.py ##捕捉生成的对象

class MyspiderPipeline(object):

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

print(item)

return item

## 5.3 Scrapy 项目实例

爬取哔哩哔哩小视频网站，中的评论信息。网址：

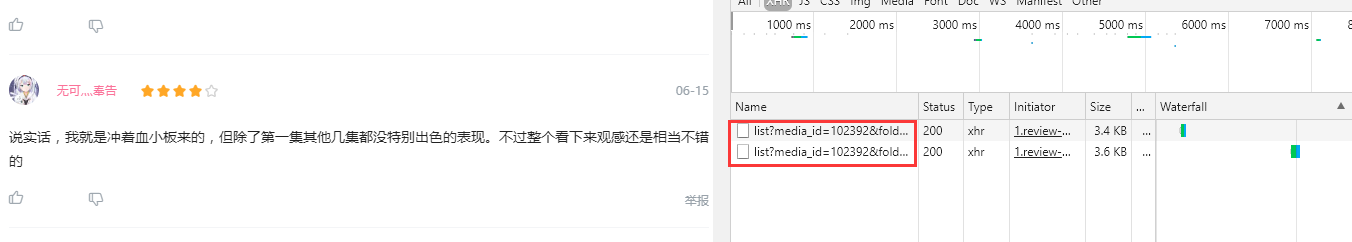
[https://www.bilibili.com/bangumi/media/md102392/?from=search&seid=1637766565190035592](https://www.bilibili.com/bangumi/media/md102392/?from=search&seid=1637766565190034492)



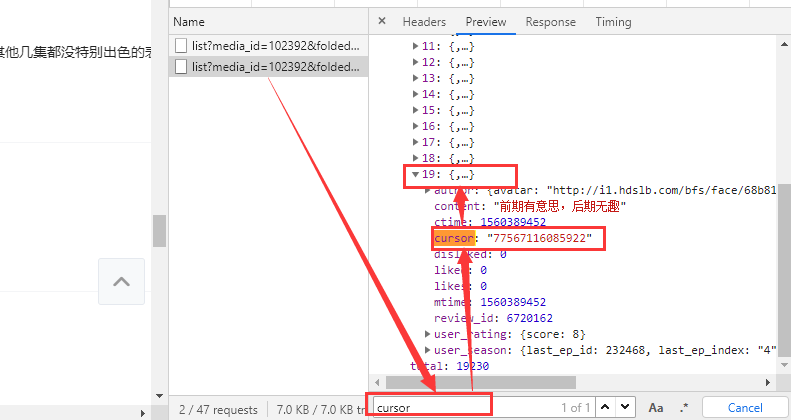
这个评论为19230，数据量还可以，我们就爬取它了。

1. 网站分析

从开发者工具中你能轻易的得到如下链接，有链接之后就好办了，如何创建项目就不在啰嗦了，我们直接进入主题，评论数据案例—获取链接。



经过分析，我们知道这个网址中的有所更改的就是cursor这个参数，也就是说我们能够知道cursor的变换规律你就掌握了爬虫基本的规律



可以看到我们搜索的数据在list中的最后一条数据。

2、项目爬取

在这里我们要尽量用到所有的模块，加深自己的印象。

Spiders/workcell.py中的代码如下：

import scrapy  
import json  
from WorkCell.items import WorkcellItem  
class WorkcellSpider(scrapy.Spider):  
 BASE\_URL = 'https://bangumi.bilibili.com/review/web\_api/short/list?media\_id=102392&folded=0&page\_size=20&sort=0&cursor={}'  
 name = 'workcell'  
 allowed\_domains = ['bangumi.bilibili.com']  
 start\_urls = [BASE\_URL.format('77592885700897')]  
 def parse(self, response):  
 print(response.url)  
 *# print('请求头',response.request.headers)* resdata = json.loads(response.body\_as\_unicode())  
 *# print(resdata)* if resdata['code'] == 0:  
 *# print(resdata)* if len(resdata['result']['list']) > 0:  
 data = resdata['result']['list']  
 cursor = data[-1]['cursor']  
 *# print(cursor)* for one in data:  
 item = WorkcellItem()  
 item['author'] = one['author']['uname']  
 item['content'] = one['content']  
 item['ctime'] = one['ctime']  
 item["disliked"] = one["disliked"]  
 item["liked"] = one["liked"]  
 item["likes"] = one["likes"]  
 item["user\_season"] = one["user\_season"]["last\_ep\_index"] if "user\_season" in one else ""  
 item["score"] = one["user\_rating"]["score"]  
 yield item  
 yield scrapy.Request(self.BASE\_URL.format(cursor),callback=self.parse)

3、实现随机UA

第一步， 在settings文件中添加一些UserAgent.

*#请求头部信息*USER\_AGENT\_LIST=[  
 "Mozilla/5.0 (Windows NT 6.1; WOW65) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/22.0.1207.1 Safari/537.1",  
 "Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11",  
 "Mozilla/5.0 (Windows NT 6.1; WOW65) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1092.0 Safari/536.6",  
 "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6",  
 "Mozilla/5.0 (Windows NT 6.2; WOW65) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/19.77.35.5 Safari/537.1",  
 "Mozilla/5.0 (X11; Linux x86\_65) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1085.9 Safari/536.5",  
 "Mozilla/5.0 (Windows NT 6.0) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1085.36 Safari/536.5",  
 "Mozilla/5.0 (Windows NT 6.1; WOW65) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",  
 "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",  
 "Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/5.0; SE 2.X MetaSr 1.0; SE 2.X MetaSr 1.0; .NET CLR 2.0.50727; SE 2.X MetaSr 1.0)",  
 "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",  
 "Mozilla/5.0 (Windows NT 6.1; WOW65) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",  
 "Mozilla/5.0 (compatible; MSIE 7.0; Windows NT 5.1; 360SE)",  
 "Mozilla/5.0 (Windows NT 6.1; WOW65) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",  
 "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",  
 "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.0 Safari/536.3",  
 "Mozilla/5.0 (X11; Linux x86\_65) AppleWebKit/535.25 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.25",  
 "Mozilla/5.0 (Windows NT 6.2; WOW65) AppleWebKit/535.25 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.25"  
]

第二步，在settings文件中设置 “DOWNLOADER\_MIDDLEWARES”,毕竟我们进行响应的时候还是通过downloader模块，将数据下载到本地

# Enable or disable downloader middlewares

# See https://doc.scrapy.org/en/latest/topics/downloader-middleware.html

DOWNLOADER\_MIDDLEWARES = {  
 *# 'WorkCell.middlewares.WorkcellDownloaderMiddleware': 553,* 'WorkCell.middlewares.RandomUserAgentMiddleware': 553,  
  
}

第三步，在 middlewares.py 文件中导入 settings模块中的 USER\_AGENT\_LIST 方法

from WorkCell.settings import USER\_AGENT\_LIST  
import random  
class RandomUserAgentMiddleware(object):  
 def process\_request(self,request,spider):  
 rand\_use= random.choice(USER\_AGENT\_LIST)  
 if rand\_use:  
 request.headers.setdefault('User-Agent',rand\_use)

通过上面几个基本步骤，随机的UA已经实现，你可以在parse函数中编写如下代码进行测试

print(response.request.headers)

5、完善item

这个操作相对简单，这些数据就是我们要保存的数据。

Items.py 文件中：

class WorkcellItem(scrapy.Item):  
 *# define the fields for your item here like:  
 # name = scrapy.Field()* author = scrapy.Field()  
 content = scrapy.Field()  
 ctime = scrapy.Field()  
 disliked = scrapy.Field()  
 liked = scrapy.Field()  
 likes = scrapy.Field()  
 score = scrapy.Field()  
 user\_season = scrapy.Field()

5、提高爬取速度

在settings.py中设置如下参数：

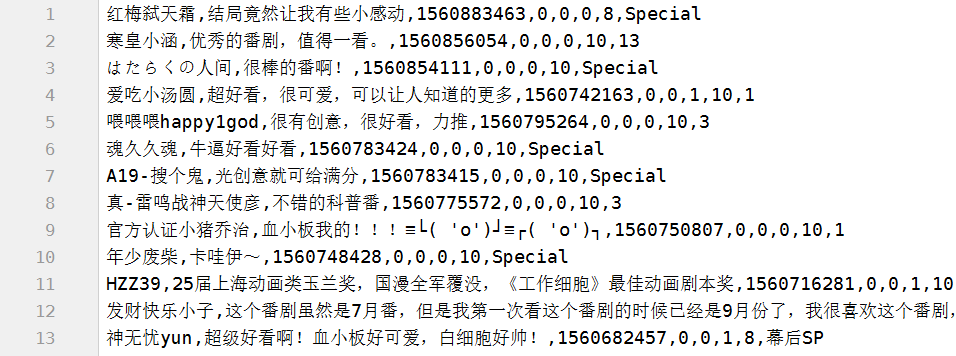
*# Obey robots.txt rules*ROBOTSTXT\_OBEY = False  
  
*# Configure maximum concurrent requests performed by Scrapy (default: 16)*CONCURRENT\_REQUESTS = 32  
  
*# Configure a delay for requests for the same website (default: 0)  
# See https://doc.scrapy.org/en/latest/topics/settings.html#download-delay  
# See also autothrottle settings and docs*DOWNLOAD\_DELAY = 0  
*# The download delay setting will honor only one of:*CONCURRENT\_REQUESTS\_PER\_DOMAIN = 16  
CONCURRENT\_REQUESTS\_PER\_IP = 16  
  
*# Disable cookies (enabled by default)*COOKIES\_ENABLED = False

6、保存数据

最后在pipelines.py 文件中，编写保存代码即可

import os  
import csv  
class WorkcellPipeline(object):  
 def \_\_init\_\_(self):  
 store\_file = os.path.dirname(\_\_file\_\_)+'/spiders/cellcsv'  
 self.file = open(store\_file,'a+',newline='',encoding='utf-8')  
 self.writer = csv.writer(self.file)  
 def process\_item(self, item, spider):  
 try:  
 self.writer.writerow((  
 item["author"],  
 item["content"],  
 item["ctime"],  
 item["disliked"],  
 item["liked"],  
 item["likes"],  
 item["score"],  
 item["user\_season"]  
 ))  
 except Exception as e:  
 print(e.args)  
 def close\_spider(self, spider):  
 self.file.close()

运行结果如下：

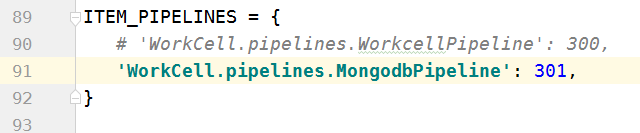


7、存储到MongoDB中

第一步：编写pipeline中的存储类

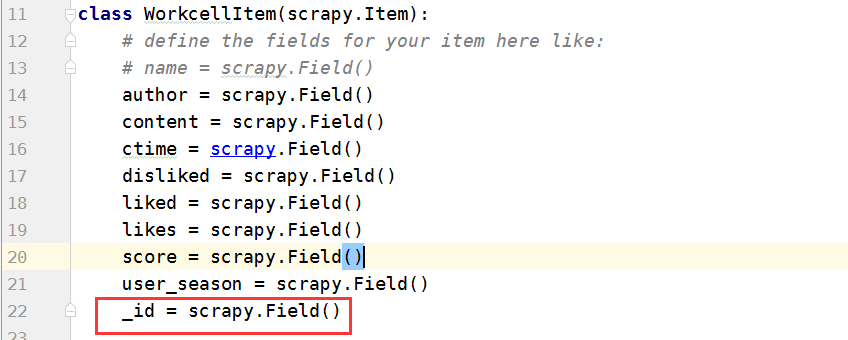
import pymongo  
DATABASE\_IP = '127.0.0.1'*#指定ip*DATABASE\_PORT = 27017*#指定端口*DATABASE\_NAME = 'sun'*#指定数据库*client = pymongo.MongoClient(DATABASE\_IP,DATABASE\_PORT)*#连接*db = client.sun*#连接数据库*collection = db.workcell*#生成表*class MongodbPipeline(object):  
 def process\_item(self, item, spider):  
 print(item)  
 try:  
 collection.insert(item)  
 except Exception as e:  
 print(e)

第二步：settings中配置item\_pipeline开启管道

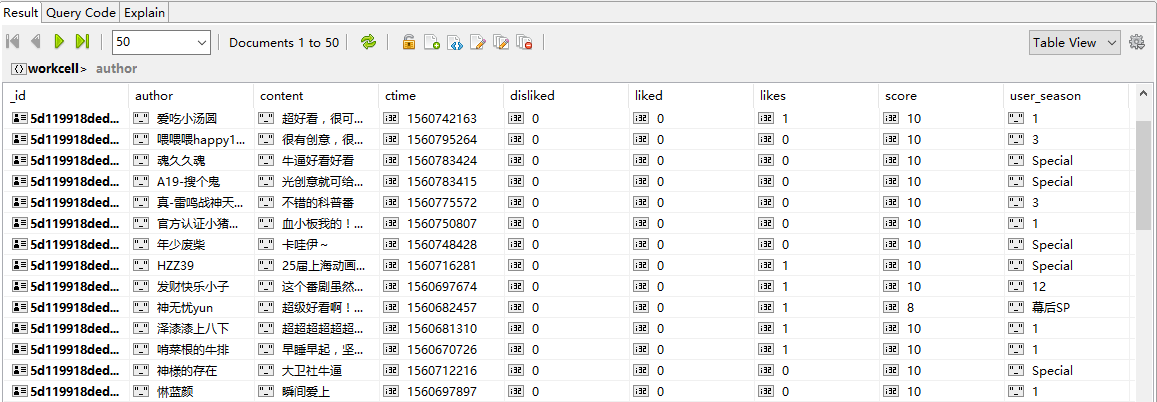


第三步：配置items.py中的字段加个‘\_id’

因为mondodb中出入数据需要这个随机id做具体标识



运行结果如下：



8、存储到mysql中

注意：先行创建表，构造自己的表结构

class MysqlPipeline(object):  
 def \_\_init\_\_(self):  
 *# 连接数据库* self.conn = pymysql.connect(  
 host='localhost',  
 port=3306,  
 user='root',  
 password='123556',  
 db='workcell',  
 charset='utf8' *# 编码格式* )  
 self.cursor = self.conn.cursor() *# 游标* def process\_item(self, item, spider):sql = """insert into workcell(….) value (%s, %s, %s, %s, %s, %s, %s, %s);"""  
 try:  
 self.cursor.execute(sql, (item[''], item[''], item[''], item[''], item[''],  
 json.dumps(item['']),  
 json.dumps(item['']), item['']))  
 self.connect.commit()  
 except BaseException as e:  
 print("存储出错", e)  
 self.conn.commit()  
 def close\_spider(self, spider):  
 self.cursor.close() *# 关闭游标* self.conn.close() *# 关闭数据库*

总结：

5.1 Scrapy response响应介绍

5.2 Scrapy item与pipline

5.3 Scrapy 项目实例