基于 scrapy 框架的爬虫

22920212204359 陈新

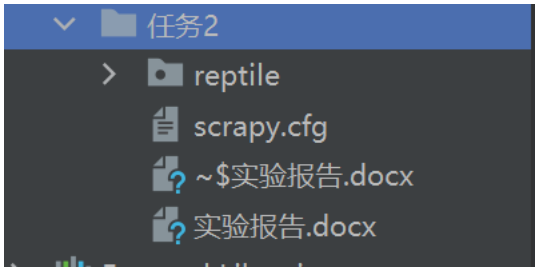
采集网页的地址：https://hongloumeng.5000yan.com/
网页描述格式：HTML 格式
采集数据：获取红楼梦每个章节的标题，保存到指定的文本文件 1 中；之后获取各章节的原文，保存到文本文件 2 中

主要步骤：

1. 建立工程文件

```
\Desktop\陈新> scrapy startproject 任务2
```

在终端输入命令 scrapy startproject 任务 2，此时生成一个项目文件夹任务 2



之后在 reptile\spiders 文件夹下面创建一个 spiders.py 文件，用于之后我们写入爬取网站的 url 和进行数据解析

2. 修改设置文件 settings.py

因为我们需要将文件存入 txt 文件，使用终端命令无法满足要求，故在该文件里开启 pipelines 用来后续输出

```
# See https://docs.scrapy.org/en/latest/topics/item-pipeline.html
ITEM_PIPELINES = {
    'reptile.pipelines.ReptilePipeline': 300,
}
```

同时我们修改对应的请求头，并且将 ROBOTSTXT_OBEY 修改为 False 否则无法进行爬虫。

```
BOT_NAME = 'reptile'

SPIDER_MODULES = ['reptile.spiders']
NEWSPIDER_MODULE = 'reptile.spiders'

# Crawl responsibly by identifying yourself (and your website) on the user-agent
USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)' \
             ' Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.42'

# Obey robots.txt rules
ROBOTSTXT_OBEY = False

LOG_LEVEL = 'ERROR'
```

为了让终端的输出简洁，我们仅允许打印错误日志

```
LOG_LEVEL = 'ERROR'
```

3. 修改设置文件 items.py

```
class ReptileItem(scrapy.Item):
    # define the fields for your item here like:
    title = scrapy.Field()
    url = scrapy.Field()
    content = scrapy.Field()
```

这里我们爬取的数据有标题、详情页 url 和原文内容，所以我们利用 scrapy.Field 方法实例化三个对象用于后续填入获取的变量

4. 修改设置文件 spiders.py

```
class SpidersSpider(scrapy.Spider):
    name = 'spiders'
    allowed_domains = ['hongloumeng.5000yan.com']
    start_urls = ['http://hongloumeng.5000yan.com/']

    def parse(self, response):
        li_list = response.xpath('//div[@class="sidamingzhu-list-mulu"]/ul/li')
        all_data = []
        for li in li_list:
            title = li.xpath('./a/text()')[0].extract()
            url = li.xpath('./a//@href')[0].get()
            item = ReptileItem()
            item['title'] = title.replace(u'\xa0', '') + '\n'
            item['url'] = url
            yield scrapy.Request(url=url, meta={'item': item}, callback=self.parse_detail)

    def parse_detail(self, response):
        item = response.meta['item']
        content=''
        div_list = response.xpath('//div[@class="grap"]/div')
        for div in div_list:
            content+=div.xpath('./text()')[0].extract().strip()+'\n'
        item['content']=content.replace(u'\xa0', '')+'\n\n'
        print(item['title'].strip()+' 已完成')
        yield item
```

在这个文件里我们填入需要爬取网站的 url，之后进行数据解析，利用 xpath 进行定位并获取相关的数据，之后实例化一个 item 对象，将需要爬取的数据写入 item 之中，这里为了能够爬取详细页面的内容，我们在 parse 方法中嵌套另一个 scrapy.request 方法，通过 meta 传递 item 的值到 parse_detail 方法之中，同理进行数据解析与获取存入 item 对象，最后返回 item 对象

5. 修改设置文件 pipelines.py

pipelines 会接收到刚刚返回 item 对象，在这个文件内我们进行数据的持久化存储，通过新建需要的文本文件，将爬取的数据写入其中，完成对数据的爬取与存储

```python
class ReptilePipeline:
    fp1 = None
    fp2 = None

    def open_spider(self, spider):
        print('************开始爬虫程序***********')
        self.fp1 = open('./1.txt', 'w', encoding='utf-8')
        self.fp2 = open('./2.txt', 'w', encoding='utf-8')

    def process_item(self, item, spider):
        title = item['title']
        content=item['content']
        self.fp1.write(title)
        self.fp2.write(title)
        self.fp2.write(content)

        return item

    def close_spider(self, spider):
        print('**********爬虫结束*****************')
        self.fp1.close()
        self.fp2.close()
```

6. 运行程序，获取爬取的数据

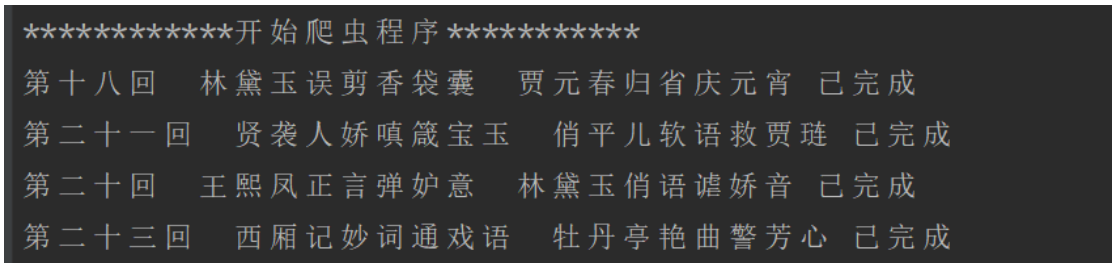
```
\Desktop\陈新> cd 任务2
\Desktop\陈新\任务2>
```

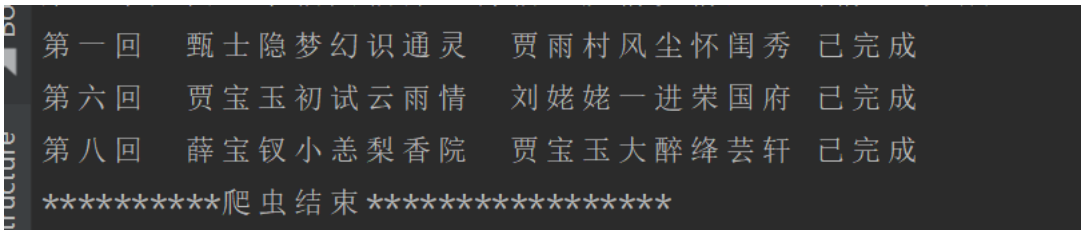
我们先输入 cd 指令进入对应的工程文件夹，之后调用 crawl 指令运行爬虫程序

获得输出

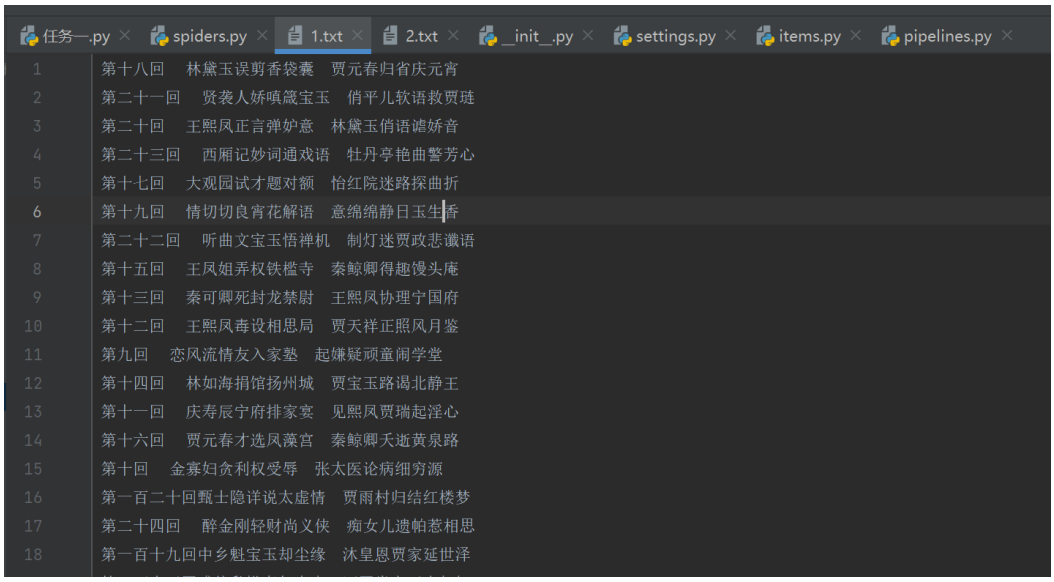
```
scrapy crawl spiders
```

（终端输出的日志信息）

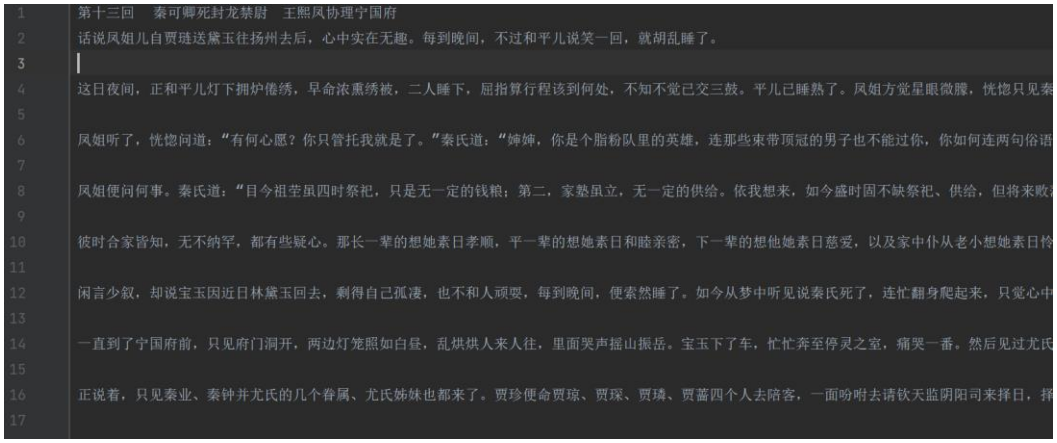




（输出的 1.txt）



（输出的 2..txt）



问题思考：

利用 scrapy 爬取的数据中明显存在顺序混乱，爬取章节顺序、所打印出来的标题顺序和原文顺序都发生错乱，且每次运行的时候结果都存在不同

原因：查阅资料我们知道 scrapy 框架在爬取数据时为提高运行速度采用了异步处理，也就是说 Scrapy 发送请求之后，不会等待这个请求的响应（也就是不会阻塞），而是可以同时发送其他请求或者做别的事情。而我们知道服务器对于请求的响应是由很多方面的因素影响的，如网络速度、解析速度、资源抢占等等，其响应的顺序是难以预测的。所以导致了我们对于每个章节完成的时间是不同的，最后输出的顺序也发生不同。