14–scrapy的用法

一.scrapy是什么

scrapy是一个强大的爬虫框架

之前我们进行爬虫操作，需要使用requests模块进行请求，使用csv模块存储数据等，而在Scrapy里，很多爬虫所涉及到的功能都已经实现了，我们只需要使用就可以了

二.scrapy的组成

- Scrapy Engine(引擎)
- Scheduler(调度器):
  - 负责处理引擎发送过来的requests对象，会把请求的URL以有序的方式排队，等待引擎提取
- Donwloader(下载器) :
  - 负责处理引擎发送过来的requests，进行网页爬取，并将返回response交给引擎
- Spiders(爬虫) :
  - 创建requests对象和接受引擎发送过来的response,从中提取出有用的数据
- Item Pipeline(数据管道):
  - 负责存储和处理Spiders提取到的有用数据
- Downloader Middlewares(下载中间件):
  - 提前对引擎发送的requests作出处理
- Spider Middlewares(爬虫中间件):
  - 提前接受并处理引擎发送来的response

三.Scrapy的工作原理

引擎统领其他成员进行协作

四.如何使用Scrapy

步骤:

1. 创建Scrapy项目
2. 定义Item
3. 编写spiders
4. 修改settings.py
5. 运行Scrapy

Scrapy的安装:

win系统：pip install Scrapy

mac系统：pip3 install Scrapy

首先在本地电脑中跳转到想要保存的目录下，在命令行使用cd命令进行跳转。

1.创建Scrapy项目

scrapy startproject douban，douban就是我们项目的名字

Scrapy项目里每个文件都有它对应的具体功能。例如settings.py 是scrapy里的各种设置、items.py是用来定义数据的、pipelines.py是用来处理数据的，它们对应的就是Scrapy的结构中的Item Pipeline（数据管道）。

spiders是放置爬虫的目录。我们可以在spiders这个文件夹里创建爬虫文件。我们来把这个文件，命名为Book_douban_Top250。后面的大部分代码都需要在这个Book_douban_Top250.py文件里编写。

先在Book_douban_Top250.py文件里导入我们需要的模块

```
import scrapy
import bs4
```

导入BeautifulSoup用于解析和提取数据;导入scrapy是待会我们要用创建类的方式写这个爬虫，我们所创建的类将直接继承scrapy中的scrapy.Spider类。这样，有许多好用属性和方法，就能够直接使用。

2.编辑爬虫

爬虫代码

```
import scrapy
import bs4
from ..items import BookDoubanItem
#定义一个类DoubanSpider，它继承自scrapy.Spider
class DoubanSpider(scrapy.Spider):
    #定义爬虫的名字为book_douban。
    name = 'book_douban'
    #定义爬虫爬取网址的域名。
    allowed_domains = ['book.douban.com']
    #定义爬虫爬取网址的域名。
    start_urls = ['https://book.douban.com/top250?start=0']
    #定义起始网址。
    for x in range(3):
        url = 'https://book.douban.com/top250?start={num}'.format(x * 25)
        #添加新的url到sart_urls中
        start_urls.append(url)
  # parse是默认处理response的方法
    def parse(self, response):
        # 用BeautifulSoup解析response
        bs = bs4.BeautifulSoup(response.text, 'html.parser')
        #用find_all提取<tr class="item">元素，这个元素里含有书籍信息
        datas = bs.find_all('tr', class_="item")
        # 遍历data
```

```
        for data in datas:
            # 实例化DoubanItem这个类
            item = BookDoubanItem()
            # 提取出书名，并把这个数据放回DoubanItem类的title属性里
item['title'] = data.find_all('a')[1]['title']
            # 提取出出版信息，并把这个数据放回DoubanItem类的publish里
          item['publish'] = data.find('p', class_='pl').text
            # 提取出评分，并把这个数据放回DoubanItem类的score属性里。
          item['score'] = data.find('span', class_='rating_nums').text
            # 打印书名
            print(item['title'])
            # yield item是把获得的item传递给引擎
            yield item
```

3.设置

Scrapy里的默认设置没被修改。比如我们需要修改请求头（User-Agent需要设置）。点击settings.py文件，你能在里面找到如下的默认设置代码：

```
# Crawl responsibly by identifying yourself (and your website) on the
user-agent
#USER_AGENT = 'class13 (+http://www.yourdomain.com)'
# Obey robots.txt rules
ROBOTSTXT_OBEY = True
```

把USER _AGENT的注释取消（删除#），然后替换掉user-agent的内容，就是修改了请求头。

因为Scrapy是遵守robots协议的，如果是robots协议禁止爬取的内容，Scrapy也会默认不去爬取，所以我们还得修改Scrapy中的默认设置。

把ROBOTSTXT_OBEY=True改成ROBOTSTXT_OBEY=False，就是把遵守robots协议换成无需遵从robots协议，这样Scrapy就能不受限制地运行。

```
# Crawl responsibly by identifying yourself (and your website) on the
user-agent
USER_AGENT = 'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/78.0.3904.87 Safari/537.36'
# Obey robots.txt rules
ROBOTSTXT_OBEY = False
```

4.运行

想要运行Scrapy有两种方法：

一种是在本地电脑的终端跳转到scrapy项目的文件夹（跳转方法：cd+文件夹的路径名），然后输入命令行：scrapy crawl book_douban（book_douban就是我们爬虫的名字）

另一种运行方式需要我们在最外层的大文件夹里新建一个main.py文件（与scrapy.cfg同级）。在这个main.py文件里，输入以下代码，点击运行，Scrapy的程序就会启动。

```
from scrapy import cmdline
#导入cmdline模块,可以实现控制终端命令行
```

```
cmdline.execute(['scrapy','crawl','book_douban'])
#用execute（）方法，输入运行scrapy的命令
```