第3章 Scrapy爬虫框架

3.1 基本原理

Scrapy是一个Python爬虫框架，用来提取结构性数据，非常适合做一些大型爬虫项目，并且开发者利用这个框架，可以不用关注细节，它比BeautifulSoup更加完善，如果说BeautifulSoup是车轮，而Scrapy则是一辆车。

安装Scrapy的命令：pip install scrapy

Scrapy数据流是由执行的核心引擎(ENGINE)控制，工作流程如图3-1所示。下面简要说明图3-1中各组件的功能。

1. ENGINE

爬虫引擎负责控制各个组件之间的数据流，当某些操作触发事件后都是通过ENGINE来处理的。

2. SCHEDULER

接收来自ENGINE的请求并将请求放入队列中，且通过事件返回给ENGINE。

3. DOWNLOADER

通过ENGINE请求下载网络数据并将结果响应给ENGINE。

4. SPIDERS

SPIDERS发出请求，并处理ENGINE返回给它的下载器响应数据，将ITEMS和数据请求返回给ENGINE。

5. ITEM PIPELINES

负责处理ENGINE返回SPIDERS解析后的数据，并且将数据持久化，例如将数据存入数据库或者文件。

6. MIDDLEWARE

ENGINE和SPIDERS之间的交互组件，以插件的形式存在。

图3-1中Scrapy工作流程的步骤说明如下。

(1) 爬虫引擎ENGINE获得初始请求开始抓取。

(2) 爬虫引擎ENGINE开始请求调度程序SCHEDULER，并准备对下一次的请求进行抓取。

(3) 爬虫调度器返回下一个请求给爬虫引擎。

(4) 引擎请求发送到下载器DOWNLOADER，通过下载中间件下载网络数据。

（5）一旦下载器完成页面下载，将下载结果返回给爬虫引擎 ENGINE。

（6）爬虫引擎 ENGINE 将下载器 DOWNLOADER 的响应通过中间件 MIDDLEWARE 返回给爬虫 SPIDERS 进行处理。

（7）爬虫 SPIDERS 处理响应，并通过中间件 MIDDLEWARE 返回处理后的 ITEMS，以及新的请求给引擎。

（8）引擎发送处理后的 ITEMS 到项目管道，然后把处理结果返回给调度器 SCHEDULER，调度器计划处理下一个请求抓取。

重复以上过程，直到抓取完所有的 URL 请求。

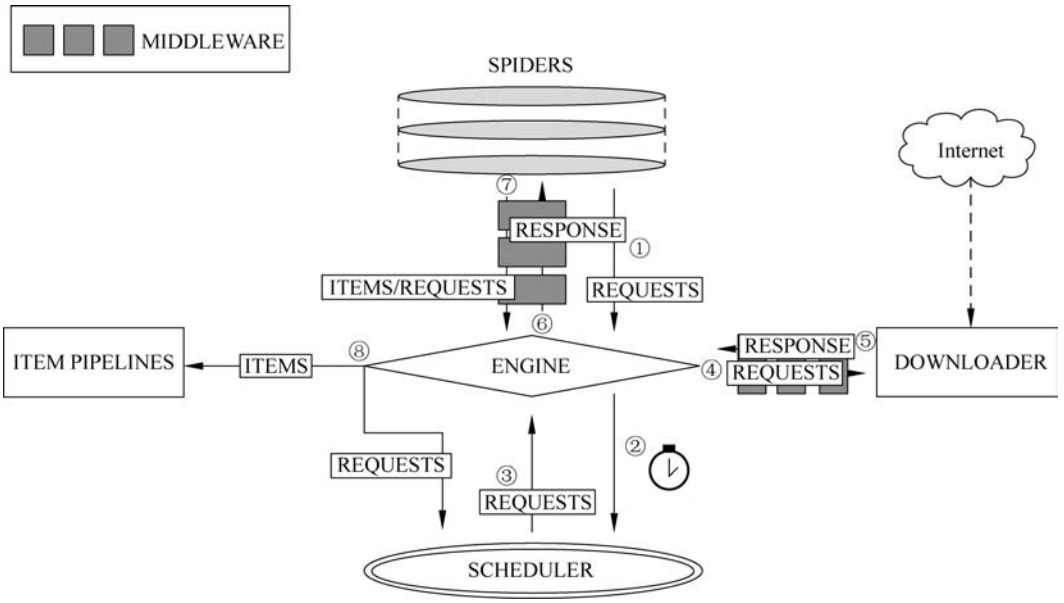

图 3-1　Scrapy 工作流程

3.2　应用举例

【例 3-1】 利用 Scrapy 抓取豆瓣图书的标签信息。

（1）建立项目和 spider 模板。

在命令窗口中输入以下命令：

```
scrapy startproject dushu
cd dushu
scrapy genspider book book.douban.com
```

本例中项目名称是 dushu，爬虫文件是 book.py，网站域名是 book.douban.com。爬虫目录结构如图 3-2 所示。

（2）编写 book.py。

按 F12 键，查看源码结构，如图 3-3 所示。可以发现书籍的信息在标签< tr >属性为

图 3-2 爬虫目录结构

item 的代码块中，而书籍的地址在标签< a >中，利用 yield 命令将这个请求的结果返回。

图 3-3 源码结构

然后打开书籍信息界面的源代码，搜索 tag 找到了书籍标签的所在位置，如图 3-4 所示。可以用正则表达式 tag/. * ?"来得到书籍的标签，然后用 yield 命令返回得到的书籍信息。

```
<div class="indent">    <span class="">
    <a class="  tag" href="/tag/红楼梦">红楼梦</a>      </span>
  <span class="">
    <a class="  tag" href="/tag/古典文学">古典文学</a>      </span>
  <span class="">
    <a class="  tag" href="/tag/曹雪芹">曹雪芹</a>      </span>
  <span class="">
    <a class="  tag" href="/tag/经典">经典</a>      </span>
  <span class="">
    <a class="  tag" href="/tag/古典名著">古典名著</a>      </span>
  <span class="">
    <a class="  tag" href="/tag/名著">名著</a>      </span>
  <span class="">
    <a class="  tag" href="/tag/四大名著">四大名著</a>      </span>
  <span class="">
    <a class="  tag" href="/tag/小说">小说</a>      </span>
</div>
</div>
```

图 3-4 书籍标签

book.py 的完整代码如下：

```
import scrapy
from bs4 import BeautifulSoup
import re

class BookSpider(scrapy.Spider):
```

```
    name = 'book'
    start_urls = ['https://book.douban.com/top250?icn=index-book250-all']

    def parse(self, response):
        soup = BeautifulSoup(response.text, 'html.parser')
        for item in soup.find_all('tr', attrs={'class': 'item'}):
            for href in item.find_all('a'):
                if href.string != None:
                    url = href.attrs['href']
                    yield scrapy.Request(url, callback=self.parse_book)

    def parse_book(self, response):
        infoDict = {}
        booksoup = BeautifulSoup(
            response.text, 'html.parser')
        infoDict.update(
            {'bookname': booksoup.title.string[:-4]})
        tagInfo = re.findall('tag/.*?"', response.text)
        tag = []
        for i in tagInfo:
            tag.append(i[4:])
        infoDict['tag'] = tag
        yield infoDict
```

（3）编写 pipelines.py。

在 pipelines.py 文件中设定一个 filename 存放文件名，然后打开这个文件将得到的内容写进去。

```
class DushuPipeline(object):
    filename = 'book.txt'
    def open_spider(self, spider):
        self.f = open(self.filename, 'w')

    def close_spider(self, spider):
        self.f.close()

    def process_item(self, item, spider):
        try:
            line = str(dict(item)) + '\n'
            self.f.write(line)
        except:
            pass
        return item
```

（4）编写 settings.py。

打开 settings.py 文件，修改 USER-AGENT，并将 pipelines 设定为所编写的类。

```
USER_AGENT = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/55.0.2883.87 Safari/537.36'
ITEM_PIPELINES = {
    'dushu.pipelines.DushuPipeline': 300,}
```

(5) 启动抓取。

```
scrapy crawl book
```

运行结束后文件夹中就会得到一个 book.txt 文件,如图 3-5 所示。

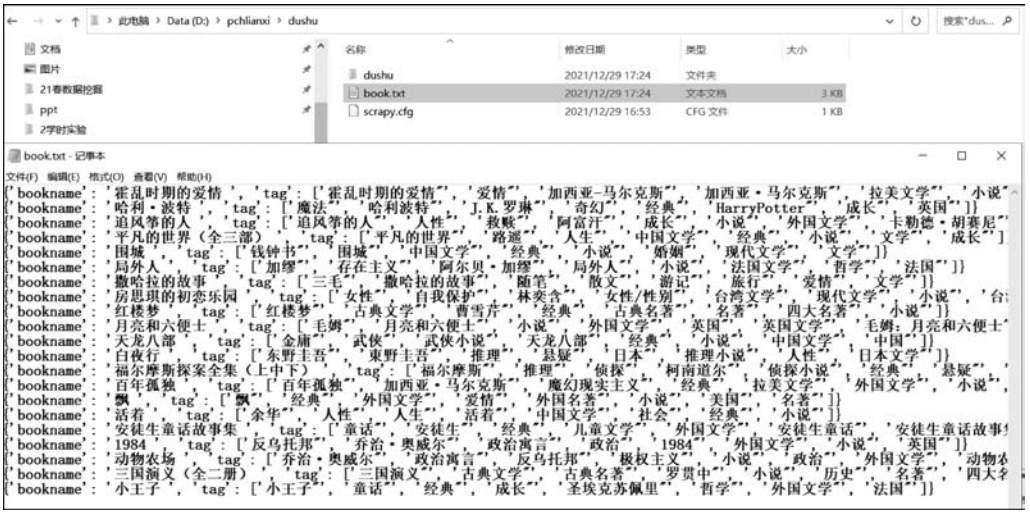

图 3-5 book.txt 文件

习 题

利用 Scrapy 框架抓取 CSDN 博客数据,并保存到文件中。