4.5 实践项目—爬取当当网站图书数据

4.5.1 网站图书数据分析

当当图书网站是国内比较大型的图书网站，这个项目的目的就是对该网站的某个主题的一类图书的数据的爬取，把爬取的数据存储到 MySQL 数据库中。

例如我们关心 Python 类的图书，想知道网站上有什么 Python 的图书，用 Chrome 浏览器进入当当网站，在搜索关键字中输入"Python"搜索得到 Python 的图书，地址转为：

http://search.dangdang.com/?key=Python&act=input

这类的图书很多，点击“下一页”后地址转为：

http://search.dangdang.com/?key=Python&act=input&page_index=2

从地址上我们知道知道搜索的关键字是 key 参数，页码参数是 page_index，而 act=input 参数只是表明是通过输入进行的查询。

把鼠标放在一本书上点击右键弹出菜单，执行“检查”命令，就可以看到这本书对应的 HTML 代码，如图 4-5-1 所示。

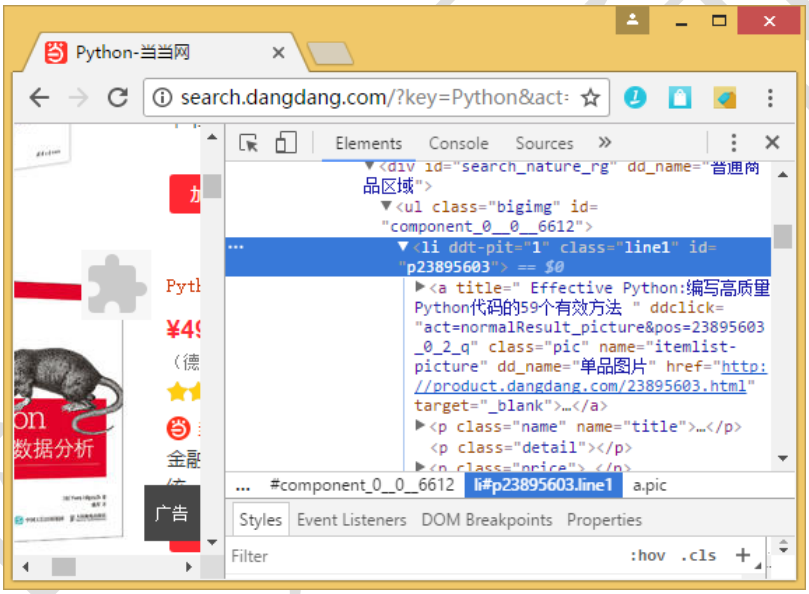

图 4-5-1 HTML 代码

仔细分析 HTML 代码的结构，我们看到每本书都是一个<li>的项目，而且它们的结构是完全一样的，这些<li>包含在一个<ul>中。

在代码中选择第一个<li>，点击鼠标右键弹出菜单，执行"Edit as HTML"进入文本编辑，复制出一本书<li>项目的代码，这段代码放到记事本中，保存为 book.txt 文本文件时提示包含 Unicode 编码字符，于是按要求以 Unicode 编码保存为 book.txt 文件。然后编写一小段程序用 BeautifulSoup 装载：

```
from bs4 import BeautifulSoup
fobj=open("c:\\example\\book.txt","rb")
data=fobj.read()
fobj.close()
data=data.decode("utf-16")
soup=BeautifulSoup(data,"lxml")
```

print(soup.prettify())

我们通过 prettify 整理后就可以清晰看到<li>层次结构，结果如下：

```
<li class="line1" ddt-pit="1" id="p23473514">
 <a class="pic" dd_name="单品图片"
ddclick="act=normalResult_picture&amp;pos=23473514_0_2_q"
href="http://product.dangdang.com/23473514.html" name="itemlist-picture" target="_blank"
title=" Python 基础教程(第 2 版・修订版) ">
  <img alt=" Python 基础教程(第 2 版・修订版) "
src="http://img3x4.ddimg.cn/20/11/23473514-1_b_5.jpg"/>
 </a>
 <p class="name" name="title">
  <a dd_name="单品标题"
ddclick="act=normalResult_title&amp;pos=23473514_0_2_q"
href="http://product.dangdang.com/23473514.html" name="itemlist-title" target="_blank" title="
Python 基础教程(第 2 版 •修订版)  Python 入门佳作 经典教程的全新修订 10 个项目引人入
胜 ">
   <font class="skcolor_ljg">
    Python
   </font>
   基础教程(第 2 版 •修订版)  Python 入门佳作 经典教程的全新修订 10 个项目引
人入胜
  </a>
 </p>
 <p class="detail">
  本书是经典的 Python 入门教程，层次鲜明，结构严谨，内容翔实，特别是后几章，
作者将前面讲述的内容应用到 10 个引人入胜的项目中，并以模板的形式介绍了项目的开发
过程，手把手教授 Python 开发，让读者从项目中领略 Python 的真正魅力。  本书既适合
初学者夯实基础，又能帮助 Python 程序员提升技能，即使是 Python 方面的技术专家，也能
从书里找到耳目一新的内容。
 </p>
 <p class="price">
  <span class="search_now_price">
   ¥53.00
  </span>
  <a class="search_discount" style="text-decoration:none;">
   定价：
  </a>
  <span class="search_pre_price">
   ¥79.00
  </span>
  <span class="search_discount">
   (6.71 折)
```

```
          </span>
         </p>
         <p class="dang" style="display: block">
          当当自营
         </p>
         <p class="search_star_line">
          <span class="search_star_black">
           <span style="width: 100%;">
           </span>
          </span>
          <a class="search_comment_num" dd_name="单品评论"
ddclick="act=click_review_count&amp;pos=23473514_0_2_q"
href="http://product.dangdang.com/23473514.html?point=comment_point"
name="itemlist-review" target="_blank">
           11355 条评论
          </a>
         </p>
         <span class="tag_box"
style="background:url(http://img62.ddimg.cn/upload_img/00660/search_new_icon/jjg.png)
no-repeat 0
0;_background-image:none;_filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='
http://img62.ddimg.cn/upload_img/00660/search_new_icon/jjg.png',sizingMethod='noscale');">
         </span>
         <p class="search_book_author">
          <span>
           (挪)
           <a dd_name="单品作者" href="http://search.dangdang.com/?key2=海特兰德
&amp;medium=01&amp;category_path=01.00.00.00.00.00" name="itemlist-author" title="(挪)海
特兰德">
            海特兰德
           </a>
          </span>
          <span>
           /2014-06-01
          </span>
          <span>
           /
           <a dd_name="单品出版社"
href="http://search.dangdang.com/?key=&amp;key3=%C8%CB%C3%F1%D3%CA%B5%E7%B3
%F6%B0%E6%C9%E7&amp;medium=01&amp;category_path=01.00.00.00.00.00" name="P_cbs"
title="人民邮电出版社">
            人民邮电出版社
           </a>
          </span>
```

```
        </p>
        <div class="shop_button">
          <p class="bottom_p">
            <a class="search_btn_cart " dd_name="加入购物车"
ddclick="act=normalResult_addToCart&amp;pos=23473514_0_2_q"
href="javascript:AddToShoppingCart(23473514)" name="Buy">
              加入购物车
            </a>
            <a class="search_btn_collect" dd_name="加入收藏"
ddclick="act=normalResult_favor&amp;pos=23473514_0_2_q"
href="javascript:showMsgBox('lcase23473514','23473514','http://wish.dangdang.com/wishlist/re
mote_addtofavorlist.aspx');" id="lcase23473514" name="collect">
              收藏
            </a>
          </p>
        </div>
      </li>
```

从这段代码我们分析怎么样取出所需要的数据。

4.5.2 网站图书数据提取

我们假定只关心图书的名称 title、作者 author、出版时间 date、出版社 publisher、价格 price 以及书的内容简介 detail，那么用 book.txt 存储的代码来测试获取的方法。从 book.txt 中的代码的分析，我们可以编写 test.py 程序获取这些数据，程序如下：

```
from bs4 import BeautifulSoup
from bs4 import UnicodeDammit
import scrapy

class TestItem:
    def __init__(self):
        self.title=""
        self.author=""
        self.date=""
        self.publisher=""
        self.price=""
        self.detail=""

    def show(self):
        print(self.title)
        print(self.author)
        print(self.date)
        print(self.price)
        print(self.publisher)
        print(self.detail)
```

```
    try:
        fobj = open("c:\\example\\book.txt", "rb")
        data = fobj.read()
        fobj.close()
        dammit = UnicodeDammit(data, ["utf-8", "utf-16","gbk"])
        data = dammit.unicode_markup
        selector = scrapy.Selector(text=data)
    try:
        fobj = open("c:\\example\\book.txt", "rb")
        data = fobj.read()
        fobj.close()
        dammit = UnicodeDammit(data, ["utf-8", "utf-16","gbk"])
        data = dammit.unicode_markup
        selector = scrapy.Selector(text=data)
        li = selector.xpath("//li")
        title = li.xpath("./a[position()=1]/@title").extract_first()
        price =
li.xpath("./p[@class='price']/span[@class='search_now_price']/text()").extract_first()
        author =
li.xpath("./p[@class='search_book_author']/span[position()=1]/a/@title").extract_first()
        date =
li.xpath("./p[@class='search_book_author']/span[position()=last()-1]/text()").extract_first()
        publisher =
li.xpath("./p[@class='search_book_author']/span[position()=last()]/a/@title").extract_first()
        detail = li.xpath("./p[@class='detail']/text()").extract_first()
        item = TestItem()
        item.title = title.strip() if title else ""
        item.author = author.strip() if author else ""
        item.date = date.strip()[1:] if date else ""
        item.publisher = publisher.strip() if publisher else ""
        item.price = price.strip() if price else ""
        item.detail = detail.strip() if detail else ""
        item.show()
    except Exception as err:
        print(err)
```

程序执行结果：

```
Python 基础教程(第 2 版・修订版)
(挪)海特兰德
2014-06-01
¥53.00
人民邮电出版社
```

本书是经典的 Python 入门教程，层次鲜明，结构严谨，内容翔实，特别是后几章，作者将前面讲述的内容应用到 10 个引人入胜的项目中，并以模板的形式介绍了项目的开发过程，手把手教授 Python 开发，让读者从项目中领略 Python 的真正魅力。　本书既适合初学者夯实基础，又能帮助 Python 程序员提升技能，即使是 Python 方面的技术专家，也能从书里找到耳目一新的内容。

由此可见这个程序的确能正确获取所要的数据，下面来分析它的工作原理：

(1)

```
fobj = open("c:\\example\\book.txt", "rb")
data = fobj.read()
fobj.close()
dammit = UnicodeDammit(data, ["utf-8", "utf-16","gbk"])
data = dammit.unicode_markup
selector = scrapy.Selector(text=data)
li = selector.xpath("//li")
```

这段程序从 book.txt 中装载数据，并识别它的编码，生成 Selector 对象，并由此找到<li>元素节点。

(2)

```
title = li.xpath("./a[position()=1]/@title").extract_first()
```

中有多个<a>，从 HTML 代码可以看到书名包含在第一个<a>的 title 属性中，因此通过 position()=1 找出第一个<a>，然后取出 title 属性值就是书名 title。

(3)

```
price =
li.xpath("./p[@class='price']/span[@class='search_now_price']/text()").extract_first()
```

价钱包含在<li>中的 class='price'的<p>元素下面的 class='search_now_price'的元素的文本中。

(4)

```
author =
li.xpath("./p[@class='search_book_author']/span[position()=1]/a/@title").extract_first()
```

作者包含在<li>下面的 class='search_book_author'的<p>元素下面的第一个元素的 title 属性中，其中 span[position()=1]就是限定第一个 。

(5)

```
date =
li.xpath("./p[@class='search_book_author']/span[position()=last()-1]/text()").extract_first()
```

出版日期包含在<li>下面的 class='search_book_author'的<p>元素下面的倒数第二个元素的文本中，其中 span[position()=last()-1]就是限定倒数第二个 ，last()是最后一个的序号。

(6)

```
publisher =
li.xpath("./p[@class='search_book_author']/span[position()=last()]/a/@title").extract_first()
```

出版社包含在<li>下面的 class='search_book_author'的<p>元素下面的最后一个元素的 title 属性中，其中 span[position()=last()]就是最后一个 ，last()是最后一个的序号。

(7)

```
        detail = li.xpath("./p[@class='detail']/text()").extract_first()
```

在<li>下面的 class='detail'的<p>的文本就是书的简介。

(8)

```
        item = TestItem()
        item.title = title.strip() if title else ""
        item.author = author.strip() if author else ""
        item.date = date.strip()[1:] if date else ""
        item.publisher = publisher.strip() if publisher else ""
        item.price = price.strip() if price else ""
        item.detail = detail.strip() if detail else ""
        item.show()
```

无论是哪个数据，如果存在那么 extract_first()就返回这个数据的值，如果不存在就返回 None，为了避免出现 None 的值，我们把 None 转为空字符串。其中：

```
        item.date = date.strip()[1:] if date else ""
```

从 HTML 中看到日期前面有一个符号"/"，因此如果日期存在时就把这个前导的符号"/"去掉。

4.5.3 网站图书数据爬取

1. 创建 MySQL 数据库

在 MySQL 中创建数据库 MyDB:

```
create database MyDB
```

然后在 MySQL 中创建一个图书表 books 如下：

```
create table books
(
   bTitle varchar(512) primary key,
   bAuthor varchar(256),
   bPublisher varchar(256),
   bDate varchar(32),
   bPrice varchar(16),
   bDetail text
)
```

2. 创建 scrapy 项目

我们仍然使用 c:\eample 下面的 demo 项目。

3. 编写 items.py 中的数据项目类

items.py 的数据项目类 BookItem 编写如下：

```
import scrapy
class BookItem(scrapy.Item):
    # define the fields for your item here like:
    title = scrapy.Field()
    author=scrapy.Field()
    date=scrapy.Field()
    publisher=scrapy.Field()
```

```
        detail=scrapy.Field()
        price=scrapy.Field()
```

4. 编写 pipelines.pay 中的数据处理类

pipelines.py 的数据处理类 BookPipeline 编写如下：

```
import pymysql

class BookPipeline(object):
    def open_spider(self,spider):
        print("opened")
        try:

self.con=pymysql.connect(host="127.0.0.1",port=3306,user="root",passwd="123456",charset="utf8")
            self.cursor=self.con.cursor(pymysql.cursors.DictCursor)
            try:
                self.cursor.execute("create database mydb")
            except:
                pass
            self.con.select_db("mydb")
            try:
                self.cursor.execute("drop table books")
            except:
                pass
            try:
                sql="""
                create table books
(
    bTitle varchar(512) primary key,
    bAuthor varchar(256),
    bPublisher varchar(256),
    bDate varchar(32),
    bPrice varchar(16),
    bDetail text
)
                """
                self.cursor.execute(sql)
            except:
                self.cursor.execute("delete from books")
            self.opened=True
            self.count=0
        except Exception as err:
            print(err)
            self.opened=False
```

```
    def close_spider(self, spider):
        if self.opened:
            self.con.commit()
            self.con.close()
            self.opened=False
        print("closed")
        print("总共爬取",self.count,"本书籍")

    def process_item(self, item, spider):
        try:
            print(item["title"])
            print(item["author"])
            print(item["publisher"])
            print(item["date"])
            print(item["price"])
            print(item["detail"])
            print()
            if self.opened:
                self.cursor.execute("insert into books
(bTitle,bAuthor,bPublisher,bDate,bPrice,bDetail) values
(%s,%s,%s,%s,%s,%s)",(item["title"],item["author"],item["publisher"],item["date"],item["price"],
item["detail"]))
                self.count+=1
        except Exception as err:
            print(err)
        return item
```

在 scrapy 的过程中一旦打开一个 spider 爬虫就会执行这个类的 open_spider(self,spider)函数，一旦这个 spider 爬虫关闭就执行这个类的 close_spider(self,spider)函数。因此程序在 open_spider 函数中连接 MySQL 数据库，并创建操作游标 self.cursor，在 close_spider 中提交数据库并关闭数据库，程序中使用 count 变量统计爬取的书籍数量。

在数据处理函数中每次有数据到达，就显示数据内容，并使用 insert 的 SQL 语句把数据插入到数据库中。

5. 编写 scrapy 的配置文件

scrapy 的配置文件 settings.py 中要编写一句：

```
ITEM_PIPELINES = {
    'demo.pipelines.BookPipeline': 300,
}
```

这样就可以把爬取的数据推送到 pipelines 的 BookPipeline 类中去了。

6. 编写 scrapy 爬虫程序

根据前面的 HTML 代码分析，我们可以编写出爬虫程序 mySpider.py 如下：

```
import scrapy
from demo.items import BookItem
from bs4 import BeautifulSoup
from bs4 import UnicodeDammit

class MySpider(scrapy.Spider):
    name = "mySpider"
    key = 'python'
    source_url='http://search.dangdang.com/'

    def start_requests(self):
        url = MySpider.source_url+"?key="+MySpider.key
        yield scrapy.Request(url=url,callback=self.parse)

    def parse(self, response):
        try:
            dammit = UnicodeDammit(response.body, ["utf-8", "gbk"])
            data = dammit.unicode_markup
            selector=scrapy.Selector(text=data)
            lis=selector.xpath("//li['@ddt-pit'][starts-with(@class,'line')]")
            for li in lis:
                title=li.xpath("./a[position()=1]/@title").extract_first()
                price =
li.xpath("./p[@class='price']/span[@class='search_now_price']/text()").extract_first()
                author =
li.xpath("./p[@class='search_book_author']/span[position()=1]/a/@title").extract_first()
                date =
li.xpath("./p[@class='search_book_author']/span[position()=last()-1]/text()").extract_first()
                publisher =
li.xpath("./p[@class='search_book_author']/span[position()=last()]/a/@title").extract_first()
                detail = li.xpath("./p[@class='detail']/text()").extract_first()
                #detail 有时没有，结果 None

                item=BookItem()
                item["title"]=title.strip() if title else ""
                item["author"]=author.strip() if author else ""
                item["date"] = date.strip()[1:] if date else ""
                item["publisher"] = publisher.strip() if publisher else ""
                item["price"] = price.strip() if price else ""
                item["detail"] = detail.strip() if detail else ""
                yield item

            #最后一页时 link 为 None
```

```
link=selector.xpath("//div[@class='paging']/ul[@name='Fy']/li[@class='next']/a/@href").extract_
first()
                    if link:
                        url=response.urljoin(link)
                        yield scrapy.Request(url=url, callback=self.parse)

                except Exception as err:
                    print(err)
```

这个程序的核心部分就是如何提取图书的 title、author、publisher、date、price 及 detail 数据，这些我们前面已经分析过了。

程序还有一个核心部分是如何连续爬取不同的网页：

```
                    #最后一页时 link 为 None

link=selector.xpath("//div[@class='paging']/ul[@name='Fy']/li[@class='next']/a/@href").extract_
first()
                    if link:
                        url=response.urljoin(link)
                        yield scrapy.Request(url=url, callback=self.parse)
```

仔细分析网站的 HTML 代码发现在一个 的元素中包含了翻页的信息， 下面的<ul name='Fy'>下面的<li class='next'>下面的<a>链接就是下一页的链接，取出这个链接地址，通过 response.urljoin 函数整理成绝对地址，再次产生一个 scrapy.Request 对象请求，回调函数仍然为这个 parse 函数，这样就可以递归调用 parse 函数，实现下一个网页的数据爬取。爬取到最后一页时，下一页的链接为空，link=None 就不再递归调用了。

7. 执行 scrapy 爬虫程序

在 c:\example\demo\demo 中编写 run.py 程序：

```
from scrapy import cmdline
cmdline.execute("scrapy crawl mySpider -s LOG_ENABLED=False".split())
```

执行这个程序就可以爬取到所有关于 Python 的书籍，这些书籍的数据存储到 MySQL 的 MyDB 数据库中，执行完毕后在 MySQL 中可以看到爬取的结果，爬取的 Python 书籍有一千多本。