4.3 scrapy 爬取与存储数据

我们从一个网站爬取到数据后往往要存储数据到数据库中，scrapy 框架有十分方便的存储方法，为了说明这个存储过程我们先建立一个简单的网站，让后写一个 scrapy 爬虫程序爬取数据，最后存储数据。

4.3.1 建立 Web 网站

这个网站有一个网页，返回基本计算机教材，flask 程序如下：

```
import flask
app=flask.Flask(__name__)

@app.route("/")
def index():
    html="""
    <books>
    <book>
        <title>Python 程序设计</title>
        <author>James</author>
        <publisher>清华大学出版社</publisher>
    </book>
    <book>
        <title>Java 程序设计</title>
        <author>Robert</author>
        <publisher>人民邮电出版社</publisher>
    </book>
    <book>
        <title>MySQL 数据库</title>
        <author>Steven</author>
        <publisher>高等教育出版社</publisher>
    </book>
    </books>
    """
    return html

if __name__=="__main__":
    app.run()
```

我们访问这个网站时返回 xml 的数据，包含教材的名称、作者与出版社。

4.3.2 编写数据项目类

程序要爬取的数据是多本教材，每本教材有名称与作者，因此要建立一个教材的类，类中包含教材名称 title、作者 author 与出版社 publisher。在我们的 scrapy 框架中有的 c:\example\demo\demo 目录下有一个文件 items.py 就是用来设计数据项目类的，打开这个

文件，改造文件成如下形式：

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

其中 BookItem 是我们设计的教材项目类，这个类必须从 scrapy.Item 类继承，在类中定义教材的字段项目，每个字段项目都是一个 scrapy.Field 对象，这里定义了 3 个字段项目，用来存储教材名称 title、作者 author、出版社 publisher。

如果 item 是一个 BoolItem 的对象，那么可以通过 item["title"]、item["author"]、item["publisher"]来获取与设置各个字段的值，例如：

```
item=BookItem()
item["title"]="Python 程序设计"
item["author"]="James"
item["publisher"]="清华大学出版社"
print(item["title"])
print(item["author"])
print(item["publisher"])
```

4.3.3 编写爬虫程序 mySpider

数据的项目设计好后就可以编写爬虫程序如下：

```
import scrapy
from demo.items import BookItem

class MySpider(scrapy.Spider):
    name = "mySpider"
    start_urls=['http://127.0.0.1:5000']

    def parse(self, response):
        try:
            data=response.body.decode()
            selector=scrapy.Selector(text=data)
            books=selector.xpath("//book")
            for book in books:
                item=BookItem()
                item["title"]=book.xpath("./title/text()").extract_first()
                item["author"] = book.xpath("./author/text()").extract_first()
                item["publisher"] = book.xpath("./publisher/text()").extract_first()
                yield item
        except Exception as err:
            print(err)
```

这个程序访问 http://127.0.0.1:5000 的网站，得到的网页包含教材信息，程序过程如下：

(1) from demo.items import BookItem

从 demo 文件夹的 items.py 文件中引入 BookItem 类的定义。

(2)

```
data=response.body.decode()
selector=scrapy.Selector(text=data)
books=selector.xpath("//book")
```

得到网站数据并建立 Selector 对象，搜索到所有的<book>节点的元素。

(3)

```
for book in books:
    item=BookItem()
    item["title"]=book.xpath("./title/text()").extract_first()
    item["author"] = book.xpath("./author/text()").extract_first()
    item["publisher"] = book.xpath("./publisher/text()").extract_first()
    yield item
```

对于每个<book>节点，在它下面搜索到<title>节点，取出它的文本即教材名称，其中注意使用 book.xpath("./title/text()")搜索到<book>下面的<title>节点的文本，一定不能缺少"./"的部分，它表示从当前节点<book>往下搜索。同样道理搜索<author>、<publisher>节点的文本，它们组成一个 BookItem 对象，这个对象通过语句：

yield item

向上一级调用函数返回，接下来 scrapy 会把这个对象推送给与 items.py 同目录下的 pipelines.py 文件中的数据管道执行类取处理数据。

4.3.4 编写数据管道处理类

在我们的 scrapy 框架中有的 c:\example\demo\demo 目录下有一个文件 pipelines.py 就是用来数据管道处理类文件，打开这个文件可以看到一个默认的管道类，修改并设计数据管道类如下：

```
class BookPipeline(object):
    count=0
    def process_item(self, item, spider):
        BookPipeline.count+=1
        try:
            if BookPipeline.count==1:
                fobj=open("books.txt","wt")
            else:
                fobj=open("books.txt","at")
            print(item["title"], item["author"], item["publisher"])
            fobj.write(item["title"]+","+item["author"]+","+item["publisher"]+"\n")
            fobj.close()
        except Exception as err:
            print(err)
        return item
```

这个类我们命名为 BookPipeline，它继承自 object 类，类中最重要的函数是 process_item 函数，scrapy 爬取数据开始时会建立一个 BookPipeline 类对象，然后每爬取一个数据类

BookItem 项目 item，mySpider 程序会把这个对象推送给 BookPipeline 对象，同时调用 process_item 函数一次。process_item 函数的参数中的 item 就是推送来的数据，于是我们就可以在这个函数中保存爬取的数据了。注意 scrapy 要求 process_item 函数最后返回这个 item 对象。

在我们这个程序中采用文件存储爬取的数据，BookPipeline 类中先定义一个类成员 count=0，用它来记录 process_item 调用的次数。如果是第一次调用(count=1)那么就使用语句 fobj=open("books.txt","wt")新建立一个 books.txt 的文件，然后把 item 的数据写到文件中。如果不是第一次调用(count>1)，就使用语句 fobj=open("books.txt","at")打开已经存在的文件 books.txt，把 item 的数据追加到文件中。这样我们反复执行爬虫程序的过程保证每次清除掉上次的数据，记录本次爬取的数据。

4.3.5 设置 scrapy 的配置文件

我们说 mySpider 爬虫程序执行后每爬取一个 item 项目都会推送到 BookPipelines 类并调用的 process_item 函数，那么 scrapy 怎么样知道要这样做呢？前提是我们必须设置这样一个通道。

在 demo 文件夹中有一个 settings.py 的设置文件，打开这个文件可以看到很多设置项目，大部分是用#注释的语句，找到语句 ITEM_PIPLINES 的项目，把它设置成如下形式：

```
# Configure item pipelines
# See http://scrapy.readthedocs.org/en/latest/topics/item-pipeline.html

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

其中 ITEM_PIPLINES 是一个字典，把关键字改成 demo.pipelines.BookPipeline'，而 BookPipelines 就是我们在 pipelines.py 文件中设计的数据管道类的名称，后面的 300 是一个默认的整数，实际上它可以不是 300，它可以是任何整数。

这样设置后就连通了爬虫程序 mySpider 数据管道处理程序 pipelines.py 的通道，scrapy 工作时会把 mySpider 爬虫程序通过 yield 返回的每项数据推送给 pipelines.py 程序的 BoolPipeline 类，并执行 process_item 函数，这样就可以保存数据了。

从上面的分析可以看到 scrapy 把数据爬取与数据存储分开处理，它们都是异步执行的，mySpider 每爬取到一个数据项目 item，就 yield 推送给 pipelines.py 程序存储，等待存储完毕后又再次爬取另外一个数据项目 item，再次 yield 推送到 pipelines.py 程序，然后再次存储，.....，这个过程一直进行下去，直到爬取过程结束，文件 books.txt 中就存储了所有的爬取数据了。