4.4 scrapy 爬取网站数据

我们为了说明 scrapy 爬虫爬取网站多个网页的数据的过程，还是用 Flask 搭建一个小型的 Web 网站。

4.4.1 建立 Web 网站

(1) books.htm

```
<h3>计算机</h3>
<ul>
<li><a href="database.htm">数据库</a></li>
<li><a href="program.htm">程序设计</a></li>
<li><a href="network.htm">计算机网络</a></li>
</ul>
```

(2) database.htm

```
<h3>数据库</h3>
<ul>
<li><a href="mysql.htm">MySQL 数据库</a></li>
</ul>
<a href="books.htm">Home</a>
```

(3) program.htm

```
<h3>程序设计</h3>
<ul>
<li><a href="python.htm">Python 程序设计</a></li>
<li><a href="java.htm">Java 程序设计</a></li>
</ul>
<a href="books.htm">Home</a>
```

(4) network.htm

```
<h3>计算机网络</h3>
<a href="books.htm">Home</a>
```

(5) mysql.htm

```
<h3>MySQL 数据库</h3>
<a href="books.htm">Home</a>
```

(6) python.htm

```
<h3>Python 程序设计</h3>
<a href="books.htm">Home</a>
```

(7) java.htm

```
<h3>Java 程序设计</h3>
<a href="books.htm">Home</a>
```

编写一个爬虫程序爬取这个网站的所有页面的<h3>标题文字。

服务器程序：

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

def getFile(fileName):
    data=b""
    if os.path.exists(fileName):
        fobj=open(fileName,"rb")
        data=fobj.read()
        fobj.close()
    return data

@app.route("/")
def index():
    return getFile("books.htm")

@app.route("/<section>")
def process(section):
    data=""
    if section!="":
        data=getFile(section)
    return data

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

4.4.2 编写 scrapy 爬虫程序

我们仍然使用 4.1 节中的爬虫程序项目，重新编写 mySpider.py 程序如下：

```
import scrapy

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

    def parse(self, response):
        try:
            print(response.url)
            data=response.body.decode()
            selector=scrapy.Selector(text=data)
            print(selector.xpath("//h3/text()").extract_first())
```

```
            links=selector.xpath("//a/@href").extract()
            for link in links:
                url=response.urljoin(link)
                yield scrapy.Request(url=url,callback=self.parse)
        except Exception as err:
            print(err)
```

执行 run.py 结果：

http://127.0.0.1:5000
计算机
http://127.0.0.1:5000/network.htm
计算机网络
http://127.0.0.1:5000/program.htm
程序设计
http://127.0.0.1:5000/database.htm
数据库
http://127.0.0.1:5000/java.htm
Java 程序设计
http://127.0.0.1:5000/python.htm
Python 程序设计
http://127.0.0.1:5000/books.htm
计算机
http://127.0.0.1:5000/mysql.htm
MySQL 数据库

scrapy 自动筛选已经访问过的网站，我们来分析程序的执行过程：

(1) start_urls=['http://127.0.0.1:5000']

这是入口地址，访问这个地址成功后会回调 parse 函数；

(2) def parse(self, response):

这是回调函数，该函数的 response 对象包含了网站返回的信息；

(3)

```
            data=response.body.decode()
            selector=scrapy.Selector(text=data)
```

网站返回的 response.body 的二进制数据，要 decode 转为文本，然后建立 Selector 对象；

(4) print(selector.xpath("//h3/text()").extract_first())

获取网页中的<h3>标题的文本，这就是要爬取的数据，为了简单起见这个数据只有一项；

(5) links=selector.xpath("//a/@href").extract()

获取所有的<a href=...>链接的 href 值，组成 links 列表；

(6)

```
            for link in links:
                url=response.urljoin(link)
                yield scrapy.Request(url=url,callback=self.parse)
```

访问 links 的每个 link，通过 urljoin 函数与 response.url 地址组合成完整的 url 地址，再

次建立 Request 对象，回调函数仍然为 parse，即这个 parse 函数会被递归调用。其中使用了 yield 语句返回每个 Request 对象，这是 scrapy 程序的要求。

4.4.3 存储 scrapy 爬取的数据

爬虫程序爬取的数据要通过管道文件存储起来，管道文件 pipelines.py 中包含一个 BookPipeline 的类，这个类有 open_spider 与 close_spider 两个重要函数。

(1) open_spider 函数

scrapy 爬虫程序开始时会创建一个 BookPipeline 的对象，并调用 open_spider 函数，我们可以在这个函数中进行数据存储的准备，例如打开文件或者数据库等。

(2) close_spider 函数

scrapy 爬虫程序在结束时会自动调用 close_spider 函数，我们可以在这个函数中做数据的保存工作，例如关闭文件或者数据库。

根据这个原则，编写 BookPipeline 类如下：

```
class BookPipeline(object):
    def open_spider(self,spider):
        print("opened")
        self.fobj=open("books.txt","wt")
        self.opened=True

    def close_spider(self, spider):
        print("closed")
        if self.opened:
            self.fobj.close()

    def process_item(self, item, spider):
        try:
            #print(item["title"])
            self.fobj.write(item["title"]+"\n")
        except Exception as err:
            print(err)
        return item
```

由此可见在 open_spider 中打开了 books.txt 文件，使用 self.fobj 记录文件的对象，在 process_item 中把每个 item 数据写入文件，在 close_spider 中关闭文件。

重新修改 MySpider 爬虫程序，让它爬取数据后 yield 返回数据给管道程序进行存储，MySpider 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:
```

```
            print(response.url)
            data=response.body.decode()
            selector=scrapy.Selector(text=data)
            title=selector.xpath("//h3/text()").extract_first()
            print(title)
            item=BookItem()
            item["title"]=title
            yield item
            links=selector.xpath("//a/@href").extract()
            for link in links:
                url=response.urljoin(link)
                yield scrapy.Request(url=url,callback=self.parse)
        except Exception as err:
            print(err)
```

运行这个爬虫程序，结果可以看到 books.txt 中最后存储了爬到的数据如下：

```
计算机
计算机网络
程序设计
数据库
计算机
MySQL 数据库
Java 程序设计
Python 程序设计
```