day10-课堂笔记

1. 综合案例

1.1 需求

抓取孔夫子旧书网中，所有图书分类下的所有图书信息。
`https://book.kongfz.com/Cxianzhuang/cat_8002/`

要抓取的字段包括：

- 图书分类的名字
- 图书分类的url地址
- 图书的名字
- 图书的几成新
- 价格
- 快递费
- 书店的名字
- 书店的地址
- 卖家的名字

1.2 步骤分析

要抓取所有的分类，获取到所有的分类的名字，还需要获取每个分类下的所有的图书的内容。应该再获取图书分类的url地址。

获取到分类的名字以及　分类的url地址之后，　要对图书分类的url地址发起请求，获取到每一个分类的图书的内容。

获取每一个分类下的所有的图书信息，那么就需要翻页，将所有页的数据获取到。

1.3 具体实现

先去使用scrapy创建项目：

```
scrapy startproject kfz_books
```

```
(spider_py3) E:\爬虫-左文彬\day10-Scrapy-案例\02-课堂代码>scrapy startproject kfz_books
New Scrapy project 'kfz_books', using template directory 'e:\py_env\spider_py3\lib\site-packages\scrapy\templates\project', created in:
    E:\爬虫-左文彬\day10-Scrapy-案例\02-课堂代码\kfz_books

You can start your first spider with:
    cd kfz_books
    scrapy genspider example example.com

(spider_py3) E:\爬虫-左文彬\day10-Scrapy-案例\02-课堂代码>
```

创建爬虫：

```
cd kfz_books
scrapy genspider books book.kongfz.com
```

```
(spider_py3) E:\爬虫-左文彬\day10-Scrapy-案例\02-课堂代码>cd kfz_books

(spider_py3) E:\爬虫-左文彬\day10-Scrapy-案例\02-课堂代码\kfz_books>scrapy genspider books book.kongfz.com
Created spider 'books' using template 'basic' in module:
  kfz_books.spiders.books

(spider_py3) E:\爬虫-左文彬\day10-Scrapy-案例\02-课堂代码\kfz_books>
```

在 `items.py` 将我们要抓取的字段定义好：

```python
import scrapy


class KfzBooksItem(scrapy.Item):

    # 图书的分类
    category_name = scrapy.Field()
    # 图书的分类的url地址
    category_href = scrapy.Field()
    # 图书的名字
    book_name = scrapy.Field()
    # 图书几成新
    new_num = scrapy.Field()
    # 价格
    book_price = scrapy.Field()
    # 运费
    book_freight = scrapy.Field()
    # 书店的名字
    bookstore_name = scrapy.Field()
    # 书店的地址
```

```
    bookstore_address = scrapy.Field()
    # 商家的名嘴
    business_name = scrapy.Field()
```

编写爬虫代码实现读数据的抓取：

先抓取图书的所有的分类，以及分类的url地址。

```
import scrapy
from kfz_books.items import KfzBooksItem


class BooksSpider(scrapy.Spider):
    name = 'books'
    allowed_domains = ['book.kongfz.com']
    # 修改起始页的url地址
    start_urls = ['https://book.kongfz.com/Cxianzhuang/']

    def parse(self, response):
        # 1. 获取到所有的图书的分类 以及分类的url地址
        # 获取到所有的分类的标签的分组
        a_list = response.xpath("//div[@class='filter-item p-b15']//a")

        for a in a_list:
            # 创建item对象
            item = KfzBooksItem()
            # 图书的分类
            item["category_name"] = a.xpath("./span[1]/text()").extract_first()
            if not item["category_name"]:
                continue
            # 获取一下分类的url地址
            item["category_href"] = a.xpath("./@href").extract_first()
            print(item)
```

对分类的url地址发起请求，获取到每个分类下的图书的数据：

```
import scrapy
from kfz_books.items import KfzBooksItem


class BooksSpider(scrapy.Spider):
    name = 'books'
    allowed_domains = ['book.kongfz.com']
    # 修改起始页的url地址
    start_urls = ['https://book.kongfz.com/Cxianzhuang/']

    def parse(self, response):
        # 1. 获取到所有的图书的分类 以及分类的url地址
        # 获取到所有的分类的标签的分组
        a_list = response.xpath("//div[@class='filter-item p-b15']//a")

        for a in a_list[:2]:
            # 创建item对象
            item = KfzBooksItem()
            # 图书的分类
            item["category_name"] = a.xpath("./span[1]/text()").extract_first()
            if not item["category_name"]:
                continue
            # 获取一下分类的url地址
            item["category_href"] = a.xpath("./@href").extract_first()
            # 2. 将分类的url地址构造成request对象,
```

```
            yield scrapy.Request(
                url=item["category_href"],
                callback=self.parse_books,
                meta={"item": item}
            )

    def parse_books(self, response):
        # 接收一下传递过来的数据
        item = response.meta['item']
        # 获取图书的分组，遍历的获取每一本图书的信息
        div_list = response.xpath("//div[@id='listBox']/div")
        # 遍历 获取每一本图书的信息
        for div in div_list:
            # 图书的名字
            item["book_name"] = div.xpath(".//a[@class='link']/text()").extract_first()
            # 图书几成新
            item["new_num"] = div.xpath(".//div[@class='quality bold gray3']/text()").extract_first()
            # 价格
            item["book_price"] = div.xpath(".//span[@class='bold']/text()").extract_first()
            # 运费
            item["book_author"] = div.xpath(".//div[@class='f_left']/div[1]/span[2]/text()").extract_first()
            # 书店的名字
            item["bookstore_name"] = div.xpath(".//a[@class='user-info-link']/span/text()").extract_first()
            # 书店的地址
            item["bookstore_address"] = div.xpath(".//div[@class='text user-place']/text()").extract_first()
            # 商家的名字
            item["business_name"] = div.xpath(".//div[@class='text on-line']/a/text()").extract_first()

            # 数据需要保存
            yield item
```

实现翻页

```
import scrapy
from kfz_books.items import KfzBooksItem


class BooksSpider(scrapy.Spider):
    name = 'books'
    allowed_domains = ['book.kongfz.com']
    # 修改起始页的url地址
    start_urls = ['https://book.kongfz.com/Cxianzhuang/']

    def parse(self, response):
        # 1. 获取到所有的图书的分类 以及分类的url地址
        # 获取到所有的分类的标签的分组
        a_list = response.xpath("//div[@class='filter-item p-b15']//a")

        for a in a_list[:2]:
            # 创建item对象
            item = KfzBooksItem()
            # 图书的分类
            item["category_name"] = a.xpath("./span[1]/text()").extract_first()
            if not item["category_name"]:
```

```
                continue
            # 获取一下分类的url地址
            item["category_href"] = a.xpath("./@href").extract_first()
            # 2. 将分类的url地址构造成request对象,
            yield scrapy.Request(
                url=item["category_href"],
                callback=self.parse_books,
                meta={"item": item}
            )

    def parse_books(self, response):
        # 接收一下传递过来的数据
        item = response.meta['item']
        # 获取图书的分组, 遍历的获取每一本图书的信息
        div_list = response.xpath("//div[@id='listBox']/div")
        # 遍历 获取每一本图书的信息
        for div in div_list:
            # 图书的名字
            item["book_name"] = div.xpath(".//a[@class='link']/text()").extract_first()
            # 图书几成新
            item["new_num"] = div.xpath(".//div[@class='quality bold gray3']/text()").extract_first()
            # 价格
            item["book_price"] = div.xpath(".//span[@class='bold']/text()").extract_first()
            # 运费
            item["book_author"] = div.xpath(".//div[@class='f_left']/div[1]/span[2]/text()").extract_first()
            # 书店的名字
            item["bookstore_name"] = div.xpath(".//a[@class='user-info-link']/span/text()").extract_first()
            # 书店的地址
            item["bookstore_address"] = div.xpath(".//div[@class='text user-place']/text()").extract_first()
            # 商家的名字
            item["business_name"] = div.xpath(".//div[@class='text on-line']/a/text()").extract_first()

            # 数据需要保存
            yield item

        # 获取到下一页的url地址 构造request对象
        next_url = response.xpath("//a[text()='下一页']/@href").extract_first()
        yield scrapy.Request(
            url=next_url,
            callback=self.parse_books,
            meta={"item": item}
        )
```

编写管道代码 `pipelines.py`

```
from pymongo import MongoClient


class KfzBooksPipeline:

    def open_spider(self, spider):
        # 创建数据库的连接以及要操作的集合对象
        client = MongoClient()
        self.collections = client.books.book
```

```
10
11      def process_item(self, item, spider):
12          # 将数据保存到MongoDB数据库中
13          self.collections.insert_one(dict(item))
14          return item
```

在`settings.py`文件中开启管道

```
1  ITEM_PIPELINES = {
2     'kfz_books.pipelines.KfzBooksPipeline': 300,
3  }
4
5  # Crawl responsibly by identifying yourself (and your website) on the user-agent
6  USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
   like Gecko) Chrome/93.0.4577.63 Safari/537.36'
7
8  # Obey robots.txt rules
9  ROBOTSTXT_OBEY = False
```