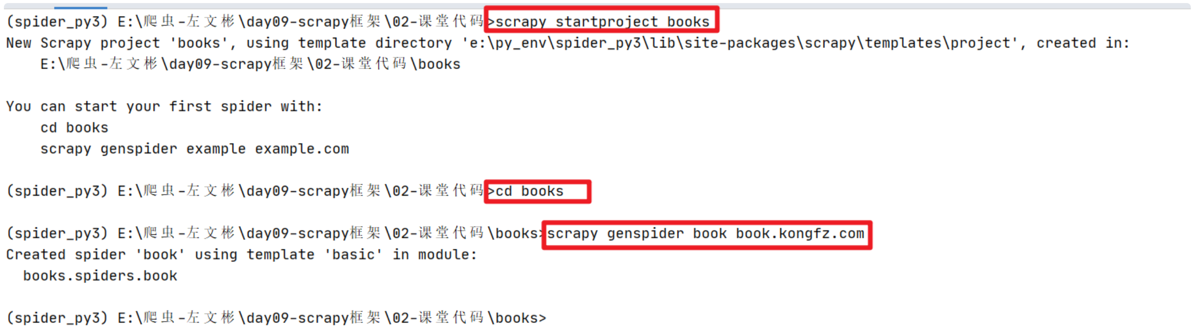
day09-课堂笔记

1. crawlSpider的使用

需求：

- 要抓取的地址： `https://book.kongfz.com/Cxianzhuang/cat_8001/`
- 要抓取的数据内容
  - 图书的名字
  - 几成新
  - 价格
  - 作者
  - 出版社
  - 出版日期
  - 店铺名
  - 还需要图书的图片： 详情页中的图片
- 还需要翻页

创建项目以及创建爬虫



```
(spider_py3) E:\爬虫-左文彬\day09-scrapy框架\02-课堂代码>scrapy startproject books
New Scrapy project 'books', using template directory 'e:\py_env\spider_py3\lib\site-packages\scrapy\templates\project', created in:
    E:\爬虫-左文彬\day09-scrapy框架\02-课堂代码\books

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

(spider_py3) E:\爬虫-左文彬\day09-scrapy框架\02-课堂代码>cd books

(spider_py3) E:\爬虫-左文彬\day09-scrapy框架\02-课堂代码\books>scrapy genspider book book.kongfz.com
Created spider 'book' using template 'basic' in module:
  books.spiders.book

(spider_py3) E:\爬虫-左文彬\day09-scrapy框架\02-课堂代码\books>
```

```python
import scrapy


class BookSpider(scrapy.Spider):
    name = 'book'
    allowed_domains = ['book.kongfz.com']
    start_urls = ['https://book.kongfz.com/Cxianzhuang/cat_8001/']

    def parse(self, response):
        # 先分组
        div_list = response.xpath("//div[@id='listBox']/div")
        # 遍历获取数据
        for div in div_list:
            # 准备一个字典 接收数据
            item = {}
            # 图书的标题
            item["title"] = div.xpath(".//a[@class='link']/text()").extract_first()
            # 图书的详情页的url地址
            item["detail_href"] = div.xpath(".//a[@class='link']/@href").extract_first()
            # 几成新
            item["new_num"] = div.xpath(".//div[@class='quality bold gray3']/text()").extract_first()
            # 价格
            item["price"] = div.xpath(".//span[@class='bold']/text()").extract_first()
            # 进入到详情页中 获取图书的图片
            # 将详情页的url地址构造成request对象
            yield scrapy.Request(
                url=item["detail_href"],
                callback=self.parse_detail,
                meta={"item": item}
            )

    def parse_detail(self, response):
        # 接收一下 从列表页中传递过来的数据
        item = response.meta["item"]

        item["img_url_list"] = response.xpath("//ul[@id='figure-info-box']/li//img/@src").extract()
        print(item)
```

想要在列表页面中 去进行到详情页中

- 获取到详情页的url地址
- 将详情页的url地址构造成request对象返回给引擎

能不能将 寻找url地址和构造request对象的操作简化， 简化成一个步骤。 这个时候就要使用到 crawspider

crawlspider: 可以根据我们指定的一些规则，自动的从页面中找到符合规则的url地址，并且自动的将获取到的url地址构造成request对象

如何使用crawlspider:

可以使用命令来创建一个crawlspider 爬虫

```
创建普通爬虫
scrapy genspider 爬虫的名字 爬虫允许的域名范围

创建crawlspider爬虫
scrapy genspider -t crawl 爬虫的名字 爬虫允许的域名范围
```

```python
import scrapy
# LinkExtractor 连接提取器 （从页面中去提取连接  url地址)
from scrapy.linkextractors import LinkExtractor
# CrawlSpider 创建的爬虫父类
# Rule 规则
from scrapy.spiders import CrawlSpider, Rule


# 爬虫类  继承了  CrawlSpider
class CrawlBookSpider(CrawlSpider):
    name = 'crawl_book'
    allowed_domains = ['book.kongfz.com']
    start_urls = ['https://book.kongfz.com/Cxianzhuang/cat_8001/']

    # 规则
    rules = (
        # 规则对象，三个参数
        # LinkExtractor(allow=r'Items/') :  连接提取器对象，在连接提取器对象中去指定提取url地址的规则、
        # allow=r'Items/'  allow=正则表达式，  表示 连接提取器根据指定的正则表达式的规则 去获取 符合规则的url地址。
        # callback  回调:  表示的是根据连接提取器提取出来的新的url地址的响应交给哪一个解析函数去处理。
        # follow=True  是否跟进=True  表示跟进，跟进的意思就是，从连接提取器提取出来的url地址的响应中继续使用连接提取器提取符合规则的url地址，直到获取不到
        Rule(LinkExtractor(allow=r'Items/'), callback='parse_item', follow=True),
    )

    def parse_item(self, response):
        item = {}
        #item['domain_id'] = response.xpath('//input[@id="sid"]/@value').get()
        #item['name'] = response.xpath('//div[@id="name"]').get()
        #item['description'] = response.xpath('//div[@id="description"]').get()
        return item
```

具体使用

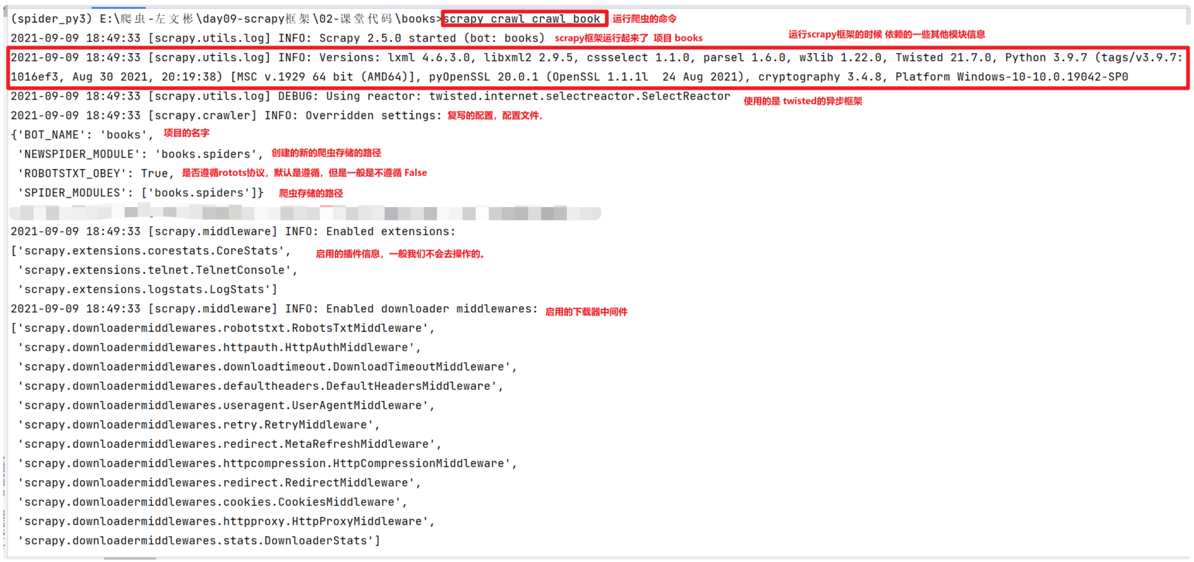
```python
import scrapy
# LinkExtractor 连接提取器  (从页面中去提取连接  url地址)
from scrapy.linkextractors import LinkExtractor
# CrawlSpider 创建的爬虫父类
# Rule 规则
from scrapy.spiders import CrawlSpider, Rule
from pprint import pprint


# 爬虫类  继承了  CrawlSpider
class CrawlBookSpider(CrawlSpider):
    name = 'crawl_book'
    allowed_domains = ['book.kongfz.com']
    start_urls = ['https://book.kongfz.com/Cxianzhuang/cat_8001/']

    # 规则
    rules = (
        # 规则对象，三个参数
        # LinkExtractor(allow=r'Items/') :  连接提取器对象，在连接提取器对象中去指定提取url地址的规则、
        # allow=r'Items/'  allow=正则表达式，  表示 连接提取器根据指定的正则表达式的规则 去获取 符合规则的url地址。、
        # restrict_xpaths   此参数接收的是一个 xpath  会根据 xpath去获取url地址
        # callback  回调:  表示的是根据连接提取器提取出来的新的url地址的响应交给哪一个解析函数去处理。
        # follow=True  是否跟进=True  表示跟进，跟进的意思就是，从连接提取器提取出来的url地址的响应中继续使用连接提取器提取符合规则的url地址，直到获取不到
        Rule(LinkExtractor(restrict_xpaths="//a[@class='link']"), callback='parse_item'),
    )

    def parse_item(self, response):
        # response参数的值 就是 详情页的response
        item = {}
        item["title"] = response.xpath("//h1/text()").extract_first().strip()
        item["price"] = response.xpath("//i[@class='now-price-text']/text()").extract_first().strip()
        item["img_url_list"] = response.xpath("//ul[@id='figure-info-box']/li/a/@href").extract()
        pprint(item)

```

2. Scrapy的日志和配置文件

2.1 Scrapy日志

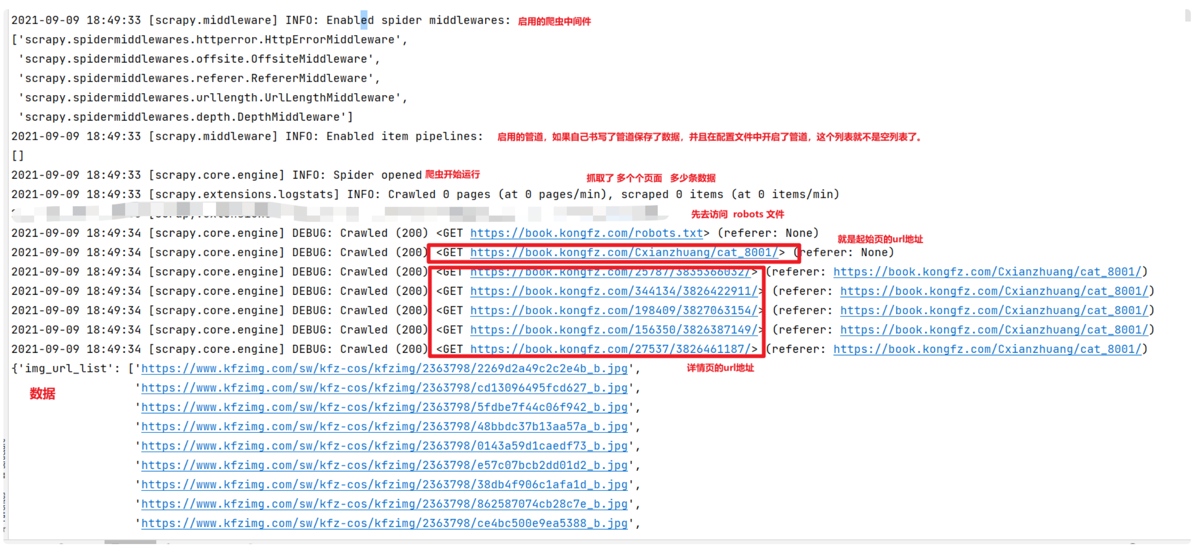
日志一般在实际的开发中比较重要，记录程序在运行过程中的状态以及输出的一些信息。方便我们去查看程序的运行状态，以及如果程序在运行过程中出现了bug，我们能够根据日志信息能够快速的定位到错误的所在，以及去解决这个问题。

在使用scrapy框架的时候，运行爬虫的时候 `scrapy crawl 爬虫名` 会在控制台输出很多的信息，这些信息就是日志。



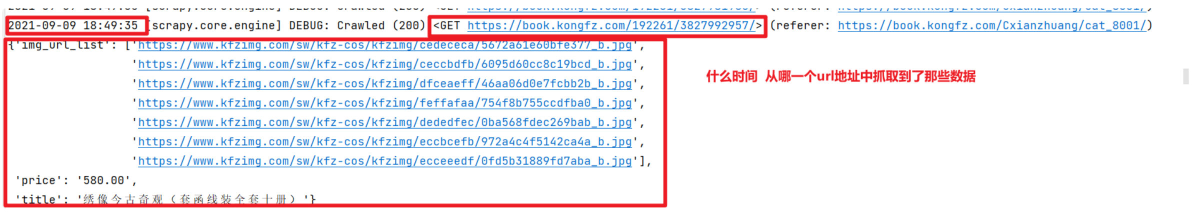
```
(spider_py3) E:\爬虫-左文彬\day09-scrapy框架\02-课堂代码\books>scrapy crawl crawl_book
2021-09-09 18:49:33 [scrapy.utils.log] INFO: Scrapy 2.5.0 started (bot: books)
2021-09-09 18:49:33 [scrapy.utils.log] INFO: Versions: lxml 4.6.3.0, libxml2 2.9.5, cssselect 1.1.0, parsel 1.6.0, w3lib 1.22.0, Twisted 21.7.0, Python 3.9.7 (tags/v3.9.7:
1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)], pyOpenSSL 20.0.1 (OpenSSL 1.1.1l  24 Aug 2021), cryptography 3.4.8, Platform Windows-10-10.0.19042-SP0
2021-09-09 18:49:33 [scrapy.utils.log] DEBUG: Using reactor: twisted.internet.selectreactor.SelectReactor
2021-09-09 18:49:33 [scrapy.crawler] INFO: Overridden settings:
{'BOT_NAME': 'books',
 'NEWSPIDER_MODULE': 'books.spiders',
 'ROBOTSTXT_OBEY': True,
 'SPIDER_MODULES': ['books.spiders']}
[redacted]
2021-09-09 18:49:33 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.logstats.LogStats']
2021-09-09 18:49:33 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware',
 'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
```

运行爬虫的命令
scrapy框架运行起来了 项目 books
运行scrapy框架的时候 依赖的一些其他模块信息
使用的是 twisted的异步框架
复写的配置，配置文件，
项目的名字
创建的新的爬虫存储的路径
是否遵循rotots协议，默认是遵循，但是一般是不遵循 False
爬虫存储的路径
启用的插件信息，一般我们不会去操作的。
启用的下载器中间件



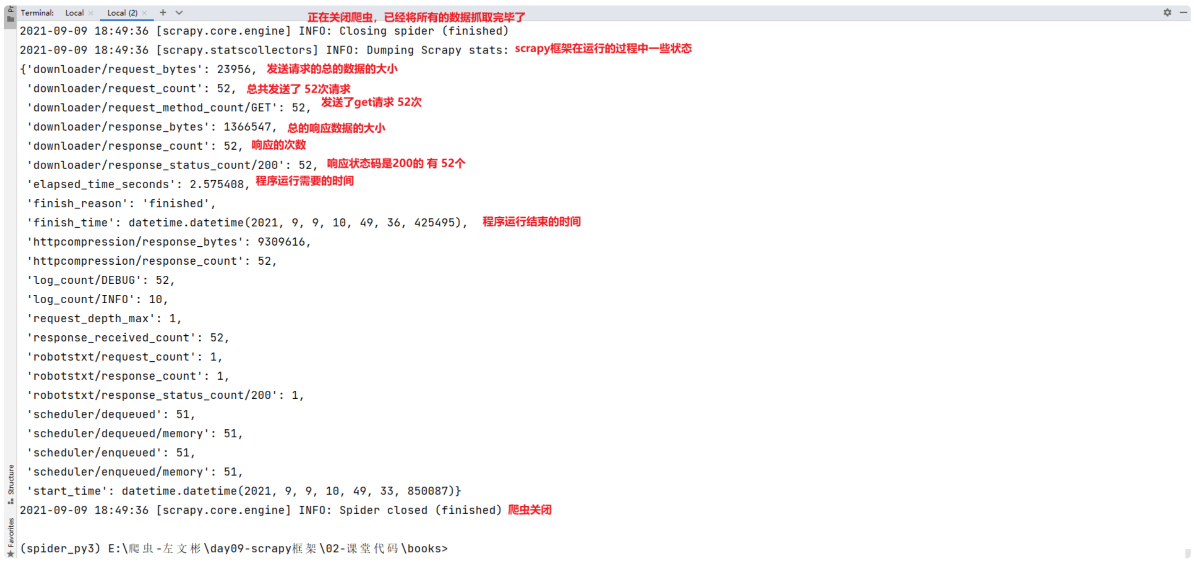
```
2021-09-09 18:49:33 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2021-09-09 18:49:33 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2021-09-09 18:49:33 [scrapy.core.engine] INFO: Spider opened
2021-09-09 18:49:33 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
[redacted]
2021-09-09 18:49:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/robots.txt> (referer: None)
2021-09-09 18:49:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/Cxianzhuang/cat_8001/> (referer: None)
2021-09-09 18:49:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/25787/3835366032/> (referer: https://book.kongfz.com/Cxianzhuang/cat_8001/)
2021-09-09 18:49:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/344134/3826422911/> (referer: https://book.kongfz.com/Cxianzhuang/cat_8001/)
2021-09-09 18:49:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/198409/3827063154/> (referer: https://book.kongfz.com/Cxianzhuang/cat_8001/)
2021-09-09 18:49:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/156350/3826387149/> (referer: https://book.kongfz.com/Cxianzhuang/cat_8001/)
2021-09-09 18:49:34 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/27537/3826461187/> (referer: https://book.kongfz.com/Cxianzhuang/cat_8001/)
{'img_url_list': ['https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/2269d2a49c2c2e4b_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/cd13096495fcd627_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/5fdbe7f44c06f942_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/48bbdc37b13aa57a_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/0143a59d1caedf73_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/e57c07bcb2dd01d2_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/38db4f906c1afa1d_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/862587074cb28c7e_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/2363798/ce4bc500e9ea5388_b.jpg',
```

启用的爬虫中间件
启用的管道，如果自己书写了管道保存了数据，并且在配置文件中开启了管道，这个列表就不是空列表了。
爬虫开始运行
抓取了 多个个页面 多少条数据
先去访问 robots 文件
就是起始页的url地址
详情页的url地址
数据



```
2021-09-09 18:49:35 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://book.kongfz.com/192261/3827992957/> (referer: https://book.kongfz.com/Cxianzhuang/cat_8001/)
{'img_url_list': ['https://www.kfzimg.com/sw/kfz-cos/kfzimg/cedececa/5672a61e60bfe377_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/ceccbdfb/6095d60cc8c19bcd_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/dfceaeff/46aa06d0e7fcbb2b_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/feffafaa/754f8b755ccdfba0_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/dededfec/0ba568fdec269bab_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/eccbcefb/972a4c4f5142ca4a_b.jpg',
                  'https://www.kfzimg.com/sw/kfz-cos/kfzimg/ecceeedf/0fd5b31889fd7aba_b.jpg'],
 'price': '580.00',
 'title': '绣像今古奇观（套函线装全套十册）'}
```

什么时间 从哪一个url地址中抓取到了那些数据



正在关闭爬虫，已经将所有的数据抓取完毕了

```
2021-09-09 18:49:36 [scrapy.core.engine] INFO: Closing spider (finished)
2021-09-09 18:49:36 [scrapy.statscollectors] INFO: Dumping Scrapy stats: scrapy框架在运行的过程中一些状态
{'downloader/request_bytes': 23956, 发送请求的总的数据的大小
 'downloader/request_count': 52, 总共发送了 52次请求
 'downloader/request_method_count/GET': 52, 发送了get请求 52次
 'downloader/response_bytes': 1366547, 总的响应数据的大小
 'downloader/response_count': 52, 响应的次数
 'downloader/response_status_count/200': 52, 响应状态码是200的 有 52个
 'elapsed_time_seconds': 2.575408, 程序运行需要的时间
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2021, 9, 9, 10, 49, 36, 425495),  程序运行结束的时间
 'httpcompression/response_bytes': 9309616,
 'httpcompression/response_count': 52,
 'log_count/DEBUG': 52,
 'log_count/INFO': 10,
 'request_depth_max': 1,
 'response_received_count': 52,
 'robotstxt/request_count': 1,
 'robotstxt/response_count': 1,
 'robotstxt/response_status_count/200': 1,
 'scheduler/dequeued': 51,
 'scheduler/dequeued/memory': 51,
 'scheduler/enqueued': 51,
 'scheduler/enqueued/memory': 51,
 'start_time': datetime.datetime(2021, 9, 9, 10, 49, 33, 850087)}
2021-09-09 18:49:36 [scrapy.core.engine] INFO: Spider closed (finished) 爬虫关闭

(spider_py3) E:\爬虫-左文彬\day09-scrapy框架\02-课堂代码\books>
```

2.2 Scrapy配置文件

```python
# 项目的名字
BOT_NAME = 'books'

# 爬虫文件存储的路径
SPIDER_MODULES = ['books.spiders']
# 创建一个新的爬虫。要拆创建在哪个位置
NEWSPIDER_MODULE = 'books.spiders'

# 设置日志保存到文件中。
LOG_FILE = "spider.log"

# 设置 User-Agent的
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'

# 是否遵循 robots 协议，我们是不遵循的
ROBOTSTXT_OBEY = False

# 设置scrapy在发送请求的时候的最大并发数 一次同时发送多少次请求 (default: 16)
# 如果说 想要让程序 一次性访问更多的url地址，可以去修改下面的数字
CONCURRENT_REQUESTS = 100

# 设置下载延迟， 每个请求之间的 间隔时间，不建议设置，
#DOWNLOAD_DELAY = 3

# 设置请求头信息、
#DEFAULT_REQUEST_HEADERS = {
#   'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
#   'Accept-Language': 'en',
#}

# 设置爬虫中间件
#SPIDER_MIDDLEWARES = {
```

```
#    'books.middlewares.BooksSpiderMiddleware': 543,
#}

# 设置下载器中间件
#DOWNLOADER_MIDDLEWARES = {
#    'books.middlewares.BooksDownloaderMiddleware': 543,
#}

# 启用管道
ITEM_PIPELINES = {
   'books.pipelines.BooksPipeline': 300,
}
```