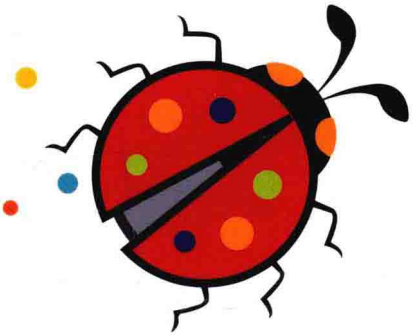
资深Python工程师10年开发经验分享

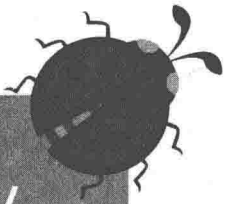
精通Scrapy网络爬虫

刘 硕 编著

- 深入讲解Scrapy的核心技术与知识点
- 从简单爬虫到分布式爬虫，涉及面广泛
- 注重实践，范例丰富，代码注释详尽



清華大學出版社



精通Scrapy网络爬虫

刘硕 编著

清华大学出版社
北京

内 容 简 介

本书深入系统地介绍了 Python 流行框架 Scrapy 的相关技术及使用技巧。全书共 14 章，从逻辑上可分为基础篇和高级篇两部分，基础篇重点介绍 Scrapy 的核心元素，如 spider、selector、item、link 等；高级篇讲解爬虫的高级话题，如登录认证、文件下载、执行 JavaScript、动态网页爬取、使用 HTTP 代理、分布式爬虫的编写等，并配合项目案例讲解，包括供练习使用的网站，以及京东、知乎、豆瓣、360 爬虫案例等。

本书案例丰富，注重实践，代码注释详尽，适合有一定 Python 语言基础，想学习编写复杂网络爬虫的读者使用。

图书在版编目（CIP）数据

精通 Scrapy 网络爬虫/刘硕编著. —北京：清华大学出版社，2017
ISBN 978-7-302-48493-6

Ⅰ. ①精… Ⅱ. ①刘… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字（2017）第 230263 号

责任编辑：王金柱
封面设计：王　翔
责任校对：闫秀华
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：180mm×230mm　　**印　　张**：14.5　　**字　　数**：325 千字
版　　次：2017 年 10 月第 1 版　　**印　　次**：2017 年 10 月第1 次印刷
印　　数：1～3000
定　　价：59.00 元

产品编号：074439-01

前　言

关于本书

如今是互联网的时代，而且正在迈入智能时代。人们早已意识到互联网中的数据是有待开采的巨大金矿，这些数据将会改善我们的生活，网络爬虫开发工作岗位的出现和不断增加正是基于对数据价值的重视。优秀的爬虫框架就像是开采金矿的强力挖掘机，如果你能娴熟地驾驶它们，就能大幅提高开采效率。

本书讲解目前最流行的 Python 爬虫框架 Scrapy，它简单易用、灵活易拓展、文档丰富、开发社区活跃，使用 Scrapy 可以高效地开发网络爬虫应用。本书的读者只需要有 Python 语言基础即可，我们从零基础、逐步由浅入深进行讲解。第 1~8 章讲解 Scrapy 开发的核心基础部分，其中包括：

- 初识 Scrapy
- 编写 Spider
- 使用 Selector 提取数据
- 使用 Item 封装数据
- 使用 Item Pipeline 处理数据
- 使用 Link Extractor 提取链接
- 使用 Exporter 导出数据
- 项目练习

第 9~14 章讲解实际爬虫开发中使用频率最高的一些实用技术，其中包括：

- 下载文件和图片
- 模拟登录
- 爬取动态页面
- 存入数据库
- 使用 HTTP 代理
- 分布式爬取

本书特色

本书的宗旨是以实用和实战为教学目标，主要特色是：

- 所有基础部分的讲解都配有代码示例，而不仅仅是枯燥的文档。
- 案例选材方面以讲解知识点为核心，尽量选择专门供练习爬虫技术的网站（不易变动）或贴近日常生活的网站（京东、知乎、豆瓣、360）进行演示。
- 在讲解某些知识点时，对 Scrapy 源码进行分析，让读者能够“知其然并知其所以然”。

另外，Python 是一门简单易学、功能强大、开发效率极高的语言，近年来在网络爬虫、数据分析、机器学习等领域得到广泛认可。虽然 Python 很容易上手，但想灵活恰当地运用它也并不简单。作者在慕课网（www.imooc.com）上推出了一套《Python 高级进阶实战》课程，可供有需求的读者进行参考：http://coding.imooc.com/class/62.html。

致谢

感谢康烁和陈渝老师在清华大学信息研究院工作期间对我在专业方面的耐心指导。
感谢清华大学出版社的王金柱编辑给予我这次写作的机会以及在写作方面的指点。
感谢赵佳音同事认真阅读全书并提出了许多的宝贵建议。
感谢剑超和任怡同学认真审阅全书并对书中代码在多个 Python 版本上进行测试。
感谢女儿刘真，她的笑容化解了写作本书时偶尔的小烦躁。

编　者
2017 年 8 月 8 日

目　录

第 1 章

初识 Scrapy

本章首先介绍爬虫的基本概念、工作流程，然后介绍 Scrapy 的安装和网络爬虫项目的实现流程，使读者对网络爬虫有一个大致的了解，并且建立起网络爬虫的编写思路。本章重点讲解以下内容：

- 网络爬虫及爬虫的工作流程。
- Scrapy 的介绍与安装。
- 网络爬虫编写步骤。

1.1 网络爬虫是什么

网络爬虫是指在互联网上自动爬取网站内容信息的程序，也被称作网络蜘蛛或网络机器人。大型的爬虫程序被广泛应用于搜索引擎、数据挖掘等领域，个人用户或企业也可以利用爬虫收集对自身有价值的数据。举一个简单的例子，假设你在本地新开了一家以外卖生意为主的餐馆，现在要给菜品定价，此时便可以开发一个爬虫程序，在美团、饿了么、百度外卖这些外卖网站爬取大量其他餐馆的菜品价格作为参考，以指导定价。

一个网络爬虫程序的基本执行流程可以总结为以下循环：

1. 下载页面

一个网页的内容本质上就是一个 HTML 文本，爬取一个网页内容之前，首先要根据网页的 URL 下载网页。

2. 提取页面中的数据

当一个网页（HTML）下载完成后，对页面中的内容进行分析，并提取出我们感兴趣的数据，提取到的数据可以以多种形式保存起来，比如将数据以某种格式（CSV、JSON）写入文件中，或存储到数据库（MySQL、MongoDB）中。

3. 提取页面中的链接

通常，我们想要获取的数据并不只在一个页面中，而是分布在多个页面中，这些页面彼此联系，一个页面中可能包含一个或多个到其他页面的链接，提取完当前页面中的数据后，还要把页面中的某些链接也提取出来，然后对链接页面进行爬取（循环 1-3 步骤）。

设计爬虫程序时，还要考虑防止重复爬取相同页面（URL 去重）、网页搜索策略（深度优先或广度优先等）、爬虫访问边界限定等一系列问题。

从头开发一个爬虫程序是一项烦琐的工作，为了避免因制造轮子而消耗大量时间，在实际应用中我们可以选择使用一些优秀的爬虫框架，使用框架可以降低开发成本，提高程序质量，让我们能够专注于业务逻辑（爬取有价值的数据）。接下来，本书就带你学习目前非常流行的开源爬虫框架 Scrapy。

1.2 Scrapy 简介及安装

Scrapy 是一个使用 Python 语言（基于 Twisted 框架）编写的开源网络爬虫框架，目前由 Scrapinghub Ltd 维护。Scrapy 简单易用、灵活易拓展、开发社区活跃，并且是跨平台的。在 Linux、MaxOS 以及 Windows 平台都可以使用。Scrapy 应用程序也使用 Python 进行开发，目前可以支持 Python 2.7 以及 Python 3.4+版本。

在任意操作系统下，可以使用 pip 安装 Scrapy，例如：

```
$ pip install scrapy
```

为确认 Scrapy 已安装成功，首先在 Python 中测试能否导入 Scrapy 模块：

```
>>> import scrapy
>>> scrapy.version_info
(1, 3, 3)
```

然后，在 shell 中测试能否执行 Scrapy 这条命令：

```
$ scrapy
Scrapy 1.3.3 - no active project

Usage:
  scrapy  [options] [args]

Available commands:
  bench          Run quick benchmark test
  commands
  fetch          Fetch a URL using the Scrapy downloader
  genspider      Generate new spider using pre-defined templates
  runspider      Run a self-contained spider (without creating a project)
  settings       Get settings values
  shell          Interactive scraping console
  startproject   Create new project
  version        Print Scrapy version
  view           Open URL in browser, as seen by Scrapy

  [ more ]       More commands available when run from project directory

Use "scrapy  -h" to see more info about a command
```

通过了以上两项检测，说明 Scrapy 安装成功了。如上所示，我们安装的是当前最新版本 1.3.3。

1.3　编写第一个 Scrapy 爬虫

为了帮助大家建立对 Scrapy 框架的初步印象，我们使用它完成一个简单的爬虫项目。

1.3.1　项目需求

在专门供爬虫初学者训练爬虫技术的网站（http://books.toscrape.com）上爬取书籍信息，如图 1-1 所示。

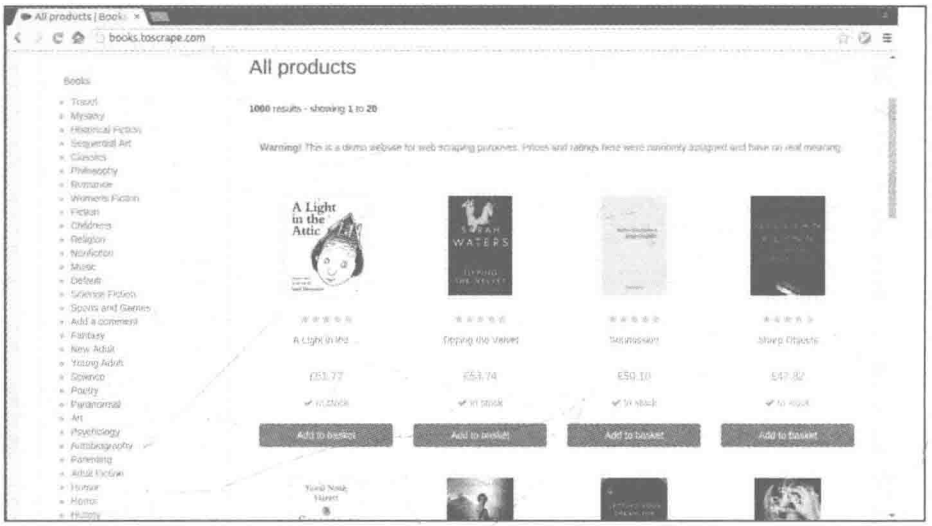


图 1-1

该网站中，这样的书籍列表页面一共有 50 页，每页有 20 本书，第一个例子应尽量简单。我们下面仅爬取所有图书（1000 本）的书名和价格信息。

1.3.2　创建项目

首先，我们要创建一个 Scrapy 项目，在 shell 中使用 scrapy startproject 命令：

```
$ scrapy startproject example
New Scrapy project 'example', using template directory
'/usr/local/lib/python3.4/dist-packages/scrapy/templates/project', created in:
/home/liushuo/book/example

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

创建好一个名为 example 的项目后，可使用 tree 命令查看项目目录下的文件，显示如下：

```
$ tree example
example/
├── example
│   ├── __init__.py
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       └── __init__.py
└── scrapy.cfg
```

随着后面逐步深入学习，大家会了解这些文件的用途，此处不做解释。

1.3.3　分析页面

编写爬虫程序之前，首先需要对待爬取的页面进行分析，主流的浏览器中都带有分析页面的工具或插件，这里我们选用 Chrome 浏览器的开发者工具（Tools→Developer tools）分析页面。

1. 数据信息

在 Chrome 浏览器中打开页面 http://books.toscrape.com，选中其中任意一本书并右击，然后选择“审查元素”，查看其 HTML 代码，如图 1-2 所示。

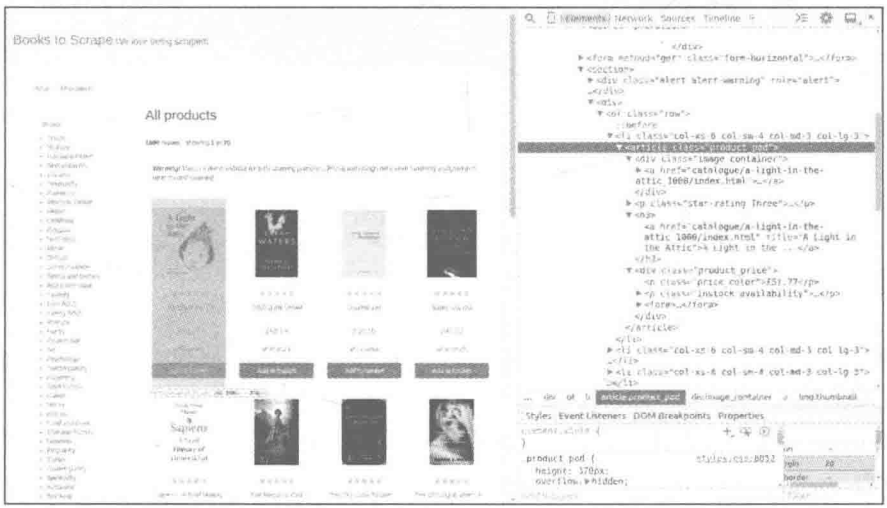


图 1-2

可以看到，每一本书的信息包裹在<article class="product_pod">元素中：书名信息在其下 h3 > a 元素的 title 属性中，如A Light in the ...；书价信息在其下 元素的文本中，如 £51.77 。

2. 链接信息

图 1-3 所示为第一页书籍列表页面，可以通过单击 next 按钮访问下一页，选中页面下方的 next 按钮并右击，然后选择“审查元素”，查看其 HTML 代码，如图 1-3 所示。

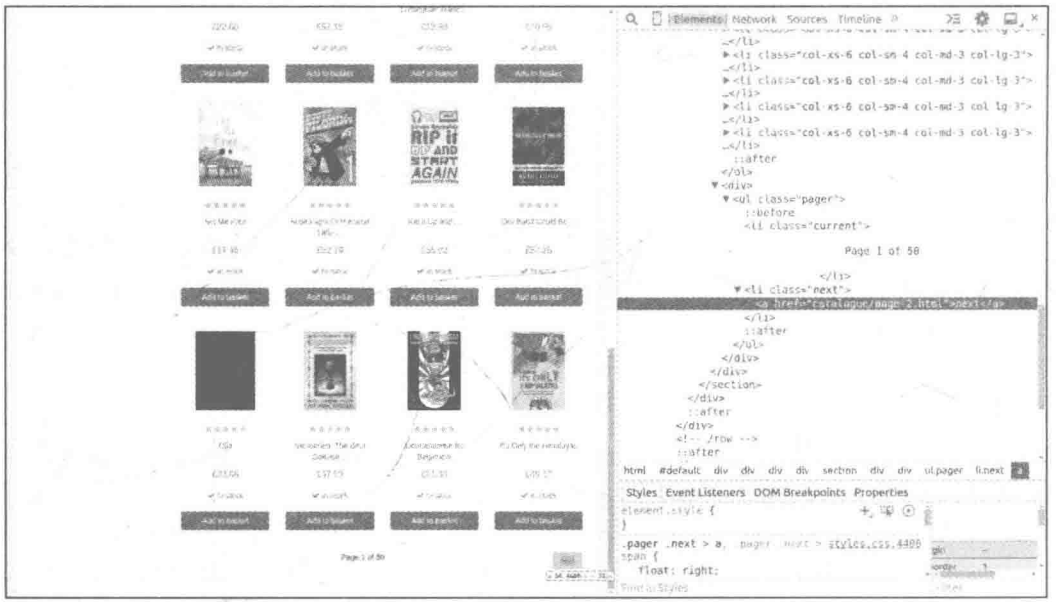


图 1-3

可以发现，下一页的 URL 在 ul.pager > li.next > a 元素的 href 属性中，是一个相对 URL 地址，如<li class="next">next。

1.3.4　实现 Spider

分析完页面后，接下来编写爬虫。在 Scrapy 中编写一个爬虫，即实现一个 scrapy.Spider 的子类。

实现爬虫的 Python 文件应位于 exmaple/spiders 目录下，在该目录下创建新文件 book_spider.py。然后，在 book_spider.py 中实现爬虫 BooksSpider，代码如下：

```
# -*- coding: utf-8 -*-
import scrapy

class BooksSpider(scrapy.Spider):
    # 每一个爬虫的唯一标识
    name = "books"

    # 定义爬虫爬取的起始点，起始点可以是多个，这里只有一个
    start_urls = ['http://books.toscrape.com/']

    def parse(self, response):
        # 提取数据
        # 每一本书的信息在<article class="product_pod">中，我们使用
        # css()方法找到所有这样的 article 元素，并依次迭代
        for book in response.css('article.product_pod'):
            # 书名信息在 article > h3 > a 元素的 title 属性里
            # 例如: <a title="A Light in the Attic">A Light in the ...</a>
            name = book.xpath('./h3/a/@title').extract_first()

            # 书价信息在 <p class="price_color">的 TEXT 中。
            # 例如: <p class="price_color">£51.77</p>
            price = book.css('p.price_color::text').extract_first()
            yield {
                'name': name,
                'price': price,
            }

        # 提取链接
        # 下一页的 url 在 ul.pager > li.next > a 里面
        # 例如: <li class="next"><a href="catalogue/page-2.html">next</a></li>
        next_url = response.css('ul.pager li.next a::attr(href)').extract_first()
        if next_url:
            # 如果找到下一页的 URL，得到绝对路径，构造新的 Request 对象
            next_url = response.urljoin(next_url)
            yield scrapy.Request(next_url, callback=self.parse)
```

如果上述代码中有看不懂的部分，大家不必担心，更多详细内容会在后面章节学习，这里只要先对实现一个爬虫有个整体印象即可。

下面对 BooksSpider 的实现做简单说明。

- name 属性
  一个 Scrapy 项目中可能有多个爬虫，每个爬虫的 name 属性是其自身的唯一标识，在一个项目中不能有同名的爬虫，本例中的爬虫取名为'books'。
- start_urls 属性
  一个爬虫总要从某个（或某些）页面开始爬取，我们称这样的页面为起始爬取点，start_urls 属性用来设置一个爬虫的起始爬取点。在本例中只有一个起始爬取点'http://books.toscrape.com'。
- parse 方法
  当一个页面下载完成后，Scrapy 引擎会回调一个我们指定的页面解析函数（默认为 parse 方法）解析页面。一个页面解析函数通常需要完成以下两个任务：
  - 提取页面中的数据（使用 XPath 或 CSS 选择器）。
  - 提取页面中的链接，并产生对链接页面的下载请求。

页面解析函数通常被实现成一个生成器函数，每一项从页面中提取的数据以及每一个对链接页面的下载请求都由 yield 语句提交给 Scrapy 引擎。

1.3.5 运行爬虫

完成代码后，运行爬虫爬取数据，在 shell 中执行 scrapy crawl <SPIDER_NAME>命令运行爬虫'books'，并将爬取的数据存储到 csv 文件中：

```
$ scrapy crawl books -o books.csv
2016-12-27 15:19:53 [scrapy] INFO: Scrapy 1.3.3 started (bot: example)
2016-12-27 15:19:53 [scrapy] INFO: INFO: Overridden settings: {'BOT_NAME': 'example',
'SPIDER_MODULES': ['example.spiders'], 'ROBOTSTXT_OBEY': True, 'NEWSPIDER_MODULE':
'example.spiders'}
2016-12-27 15:19:53 [scrapy] INFO: Enabled extensions:
['scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.feedexport.FeedExporter',
 'scrapy.extensions.logstats.LogStats']
2016-12-27 15:19:53 [scrapy] 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.chunked.ChunkedTransferMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2016-12-27 15:19:53 [scrapy] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2016-12-27 15:19:53 [scrapy] INFO: Enabled item pipelines:
[]
2016-12-27 15:19:53 [scrapy] INFO: Spider opened
2016-12-27 15:19:53 [scrapy] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0
items/min)
2016-12-27 15:19:53 [scrapy] DEBUG: Telnet console listening on 127.0.0.1:6023
2016-12-27 15:20:01 [scrapy] DEBUG: Crawled (404)  (referer: None)
2016-12-27 15:20:02 [scrapy] DEBUG: Crawled (200)  (referer: None)
2016-12-27 15:20:02 [scrapy] DEBUG: Scraped from <200 http://books.toscrape.com/>
{'name': 'A Light in the Attic', 'price': '£51.77'}
2016-12-27 15:20:02 [scrapy] DEBUG: Scraped from <200 http://books.toscrape.com/>
{'name': 'Tipping the Velvet', 'price': '£53.74'}
2016-12-27 15:20:02 [scrapy] DEBUG: Scraped from <200 http://books.toscrape.com/>
{'name': 'Soumission', 'price': '£50.10'}

... <省略中间部分输出> ...

2016-12-27 15:21:30 [scrapy] DEBUG: Scraped from <200
http://books.toscrape.com/catalogue/page-50.html>
{'name': '1,000 Places to See Before You Die', 'price': '£26.08'}
2016-12-27 15:21:30 [scrapy] INFO: Closing spider (finished)
2016-12-27 15:21:30 [scrapy] INFO: Stored csv feed (1000 items) in: books.csv
2016-12-27 15:21:30 [scrapy] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 14957,
 'downloader/request_count': 51,
 'downloader/request_method_count/GET': 51,
```