如何用 Python 爬取网页制作电子书

0 前言

有人爬取数据分析黄金周旅游景点，有人爬取数据分析相亲，有人大数据分析双十一，连小学生写论文都用上了大数据。

我们每个人每天都在往网上通过微信、微博、淘宝等上传我们的个人信息，现在就连我们的钱都是放在网上，以后到强人工智能，我们连决策都要依靠网络。网上的数据就是资源和宝藏，我们需要一把铲子来挖掘它。

最近，AI 的兴起让 Python 火了一把。实际上 Python 拥有庞大的第三方支持，生态系统非常完整，可以适用各种场景和行业。这次，我们准备通过 Python 学习爬虫的开发，既简单有趣，而且是数据采集重要一环。同时脱离应用谈技术就是耍流氓，通过制作电子书学习数据的收集与整理，即能学到东西又有实用价值。

我们将通过爬取网页信息这个很小的应用场景来体会数据预处理的思想，并从中学习了解数据处理中抓取、处理、分组、存储等过程的实现。我这次分享主要分为以下几个部分：

Python 语法的讲解，通过分享掌握简单的 Python 开发语法和思路，侧重于后面爬虫开发的需要用的内容
Scrapy 爬虫开发，通过分享了解基本的 Scrapy 开发，并实现从网络爬取数据，使用 Sigil 制作 epub 电子书
最后，我希望通过分享能够入门，并喜欢上 Python 开发，并且掌握 Scrapy 爬虫开发的思路和方法。

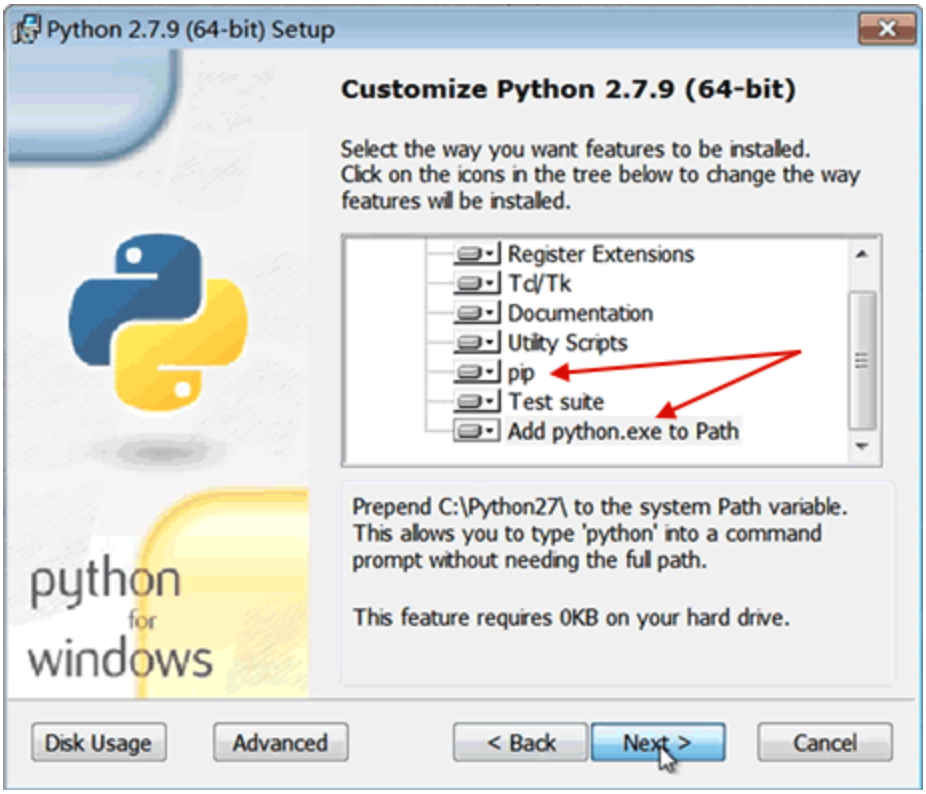
1 Python 开发

1.1 Windows下环境安装

熟悉 Windows 的安装 Python 不难，首先官网下载，地址请单击这里。

有两个版本，根据需要选择自己的版本，现在越来越多的库开始支持3，所以建议下载3，这里我们下载2。

双击下载的安装文件，一路 Next 即可，但是要注意勾选 `__pip__` 和 **Add python.exe to Path**



pip 是 Python 生态体系里面的包管理工具，很多第三方库可以通过它方便的管理。

安装 Finish 之后，打开命令行窗口，输入 Python：

如果出现这个界面说明安装成功了，如果出现下面的情况：

> ‘python’不是内部或外部命令，也不是可运行的程序或批处理文件。

需要把 python.exe 的目录添加到 path 中，一般是 C:/Python27。

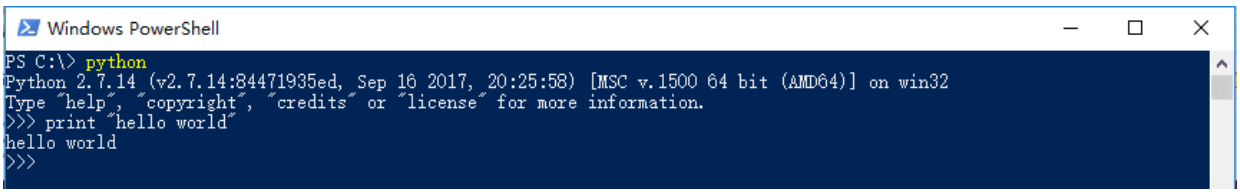
1.2 Python 之 HelloWorld

目前我所接触过的所有编程语言都只有掌握三个内容就可以了：就是输入、处理、输出。我们已经安装好了 Python，可以来一个最俗套的程序。

首先我们打开 windows 的控制台，然后输入 python 回车，然后输入如下代码：

```
print 'Hello world!'
```

我就问你俗不俗，好了看结果：



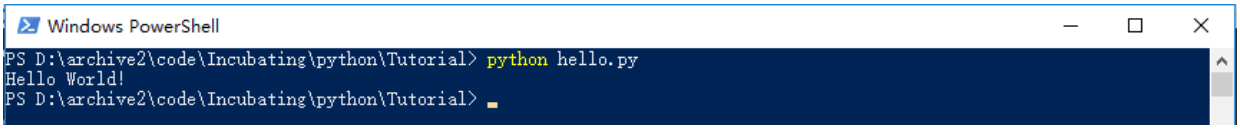
根据我上面的说法，这个程序的输入就是 Hello World 字符串，处理使系统内部的输出处理，输出结果就是'Hello World'。

我们还可以把代码写在文件里面：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

print 'Hello World!'
```

执行效果:



我们说这个不是单纯的秀一下，以前没有用户界面的时候print可以作为人机交互用途，现在多数是用于调试，可以在程序运行的时候快速的输出程序结果或者过程结果。

1.3 做菜与编程

现在有个很有意思的说法生数据（原始数据）就是没有处理过的数据，熟数据（Cooked Data）是指原始数据经过加工处理后的数据，处理包括解压缩、组织，或者是分析和提出，以备将来使用。

这就像做菜生菜是输入，菜谱是程序，洗、切、烹饪等处理是程序执行过程，最后输出的熟菜。但不管生菜、熟菜都是菜，或者都是物质。

准备食材

在程序世界里的物质组成就是数据，就像有萝卜白菜等不同的品种一样，数据也有不同的类型。我目前所接触到的数据类型主要有以下几种：

- 物理类：数据在物理内存中的表达存储方式
  - 位
  - 字
  - 字节
- 数据类：数据类中的具体类型代表了不同精度和内存中不同的存储结构
  - 整数
  - 浮点数

    - 长整型
    - 双精度

- 字符类：就是文本字符相关的数据类型
    - 字符
    - 字符串

- 逻辑类：就是逻辑真与逻辑假
    - 布尔值

- 复合类：由各基本的数据类型按照一定的结构组合而成的数据
    - 结构体
    - 类
    - 集合
    - 字典
    - 列表
    - 序列
    - Hash表
    - ……

这里我强调几点：

- 首先，这个分类不是某种语言特有，目前大多数编程语言都差不多，你理解这个思想就把自己的编程能力扩展了。
- 其次，这个东西不用专门记忆，编程是程序性的知识，运用的知识，是一种技能，你要做什么菜，你来这个分类查查需要什么原材料，再去具体研究，慢慢就会了，不做你记住了也没用。用多深，研究多深，不用就别研究浪费时间。比如说，我们一般性应用不会去考虑数据的内存模型，但是涉及到精度、性能或者边界值时我们就需要小心，研究得深一些。

器皿

食材已准备好了，可以下锅，可锅在哪里，你不能放在手里加工。程序里我们用变量、常量来盛各种数据，还有个作用域的问题，严格的厨房红案和白案是分开的，有时候砧板是不能互用的。

- 空值：四大皆空，什么也不是，不是0，不是长度为0的字符串，不是false，什么都不是
- 变量：学过数学的人都应该有这个概念，反正差不多
- 常量：固定不变的量，比如说π

烹饪手法

刚查了下，我大天朝常用的烹饪手法多达20多种，我归纳了一下，编程大概就那么几种：

- 数值计算——加减乘除、位移等

- 逻辑计算——逻辑真假判断
- 过程计算——循环、嵌套、递归等
- 数据处理——字符串、对象的操作

菜谱与炒菜

菜都准备好了，下锅怎么炒，全靠菜谱，这个就是程序，而我们按照菜谱炒菜这个过程就是程序的执行。Python 或任何一种编程语言都是博大精深，同时又是一种技能，不可能在使用之前完全掌握，也没必要。我们需要知道的是我们想吃什么（程序要输出什么），然后再去菜市场买时才找菜谱（搜索引擎查资料），最后按照我们的需求加工（编程）。

1.4 Python 简单实践

首先我们来写三个 Python 文件：

> hello.py
> ——事情的处理有落点，程序执行有入口，例如：main，这个文件可以看作程序的入口


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import pkg
print 'Hello World!'
pkg.test()
p = pkg.Person("Mike", 23)
p.showInfo()
```

> pkg.py
> ——程序可以分块编写，这样层次更分明，易于理解和维护，我们在pkg.py中编写一部分功能，作为演示模块

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

def test():
    print "Here is pkg's test"

class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age = age
        pass
```

```
    def showInfo(self):
        print self.name
        print self.age
```

> **init**.py
> ——这是一个空文件，也可以写代码，表明当前路径是包。

接下来，我们来运行一下：

```
python hello.py
```

显示结果如下：

> Hello World!
>
> Here is pkg's test
>
> Mike
>
> 23

我们运行了 hello.py 文件，然后 hello.py 导入了包 pkg；包 pkg 定义了一个方法和一个类，我们在 hello.py 文件里面调用了外部的方法和类。

2 使用 Scrapy 抓取电子书

2.1 写在爬去数据之前

虽然我们这里的数据都是从公开的网络获取，但也不能确定其版权问题，因此获取的数据仅用于编程练习，严禁分享或用于其他用途。

好了，现在我们找一个在线看书的网站，找一本书把它下载到本地。首先，我们准备下载工具，就是 Python 的爬虫框架 Scrapy。

2.2 Scrapy 安装

安装完 Python 后可以用以下的命令按照 Scrapy，有些版本的 Python 没有带 pip 需要手动安装。

```
pip install scrapy
```

pip 是Python的包管理器，大量的第三方包或者说功能可以通过这个工具来管理，所谓包就是模块化的功能集合，基本的技术参考实践里面的包。

我安装成功显示如下信息：

```
> Collecting scrapy
  Downloading Scrapy-1.5.0-py2.py3-none-any.whl (251kB)
    100% |████████████████████████████████| 256kB 181kB/s
Collecting service-identity (from scrapy)
  Downloading service_identity-17.0.0-py2.py3-none-any.whl
Collecting parsel>=1.1 (from scrapy)
  Downloading parsel-1.3.1-py2.py3-none-any.whl
Collecting six>=1.5.2 (from scrapy)
  Downloading six-1.11.0-py2.py3-none-any.whl
Collecting w3lib>=1.17.0 (from scrapy)
  Downloading w3lib-1.18.0-py2.py3-none-any.whl
Collecting lxml (from scrapy)
  Downloading lxml-4.1.1-cp27-cp27m-win_amd64.whl (3.6MB)
    100% |████████████████████████████████| 3.6MB 142kB/s
Collecting Twisted>=13.1.0 (from scrapy)
  Downloading Twisted-17.9.0-cp27-cp27m-win_amd64.whl (3.2MB)
    100% |████████████████████████████████| 3.2MB 169kB/s
Collecting pyOpenSSL (from scrapy)
  Downloading pyOpenSSL-17.5.0-py2.py3-none-any.whl (53kB)
    100% |████████████████████████████████| 61kB 313kB/s
Collecting PyDispatcher>=2.0.5 (from scrapy)
  Downloading PyDispatcher-2.0.5.tar.gz
Collecting queuelib (from scrapy)
  Downloading queuelib-1.4.2-py2.py3-none-any.whl
Collecting cssselect>=0.9 (from scrapy)
  Downloading cssselect-1.0.3-py2.py3-none-any.whl
Collecting pyasn1 (from service-identity->scrapy)
  Downloading pyasn1-0.4.2-py2.py3-none-any.whl (71kB)
    100% |████████████████████████████████| 71kB 328kB/s
Collecting attrs (from service-identity->scrapy)
  Downloading attrs-17.4.0-py2.py3-none-any.whl
Collecting pyasn1-modules (from service-identity->scrapy)
  Downloading pyasn1_modules-0.2.1-py2.py3-none-any.whl (60kB)
    100% |████████████████████████████████| 61kB 347kB/s
Collecting hyperlink>=17.1.1 (from Twisted>=13.1.0->scrapy)
  Downloading hyperlink-17.3.1-py2.py3-none-any.whl (73kB)
```

```
    100% |████████████████████████████████| 
81kB 407kB/s
Collecting Automat>=0.3.0 (from Twisted>=13.1.0->scrapy)
  Downloading Automat-0.6.0-py2.py3-none-any.whl
Collecting constantly>=15.1 (from Twisted>=13.1.0->scrapy)
  Downloading constantly-15.1.0-py2.py3-none-any.whl
Collecting zope.interface>=3.6.0 (from Twisted>=13.1.0->scrapy)
  Downloading zope.interface-4.4.3-cp27-cp27m-win_amd64.whl
(137kB)
    100% |████████████████████████████████| 
143kB 279kB/s
Collecting incremental>=16.10.1 (from Twisted>=13.1.0->scrapy)
  Downloading incremental-17.5.0-py2.py3-none-any.whl
Collecting cryptography>=2.1.4 (from pyOpenSSL->scrapy)
  Downloading cryptography-2.1.4-cp27-cp27m-win_amd64.whl (1.3MB)
    100% |████████████████████████████████| 
1.3MB 220kB/s
Requirement already satisfied: setuptools in
c:\python27\lib\site-packages (from zope.interface>=3.6.0-
>Twisted>=13.1.0->scrapy)
Collecting idna>=2.1 (from cryptography>=2.1.4->pyOpenSSL-
>scrapy)
  Downloading idna-2.6-py2.py3-none-any.whl (56kB)
    100% |████████████████████████████████| 
61kB 311kB/s
Collecting cffi>=1.7; platform_python_implementation != "PyPy"
(from cryptography>=2.1.4->pyOpenSSL->scrapy)
  Downloading cffi-1.11.2-cp27-cp27m-win_amd64.whl (163kB)
    100% |████████████████████████████████| 
163kB 183kB/s
Collecting enum34; python_version < "3" (from
cryptography>=2.1.4->pyOpenSSL->scrapy)
  Downloading enum34-1.1.6-py2-none-any.whl
Collecting asn1crypto>=0.21.0 (from cryptography>=2.1.4-
>pyOpenSSL->scrapy)
  Downloading asn1crypto-0.24.0-py2.py3-none-any.whl (101kB)
    100% |████████████████████████████████| 
102kB 194kB/s
Collecting ipaddress; python_version < "3" (from
cryptography>=2.1.4->pyOpenSSL->scrapy)
  Downloading ipaddress-1.0.19.tar.gz
Collecting pycparser (from cffi>=1.7;
platform_python_implementation != "PyPy"->cryptography>=2.1.4-
>pyOpenSSL->scrapy)
  Downloading pycparser-2.18.tar.gz (245kB)
    100% |████████████████████████████████| 
256kB 264kB/s
Installing collected packages: pyasn1, six, idna, pycparser,
cffi, enum34, asn1crypto, ipaddress, cryptography, pyOpenSSL,
attrs, pyasn1-modules, service-identity, w3lib, cssselect, lxml,
parsel, hyperlink, Automat, constantly, zope.interface,
incremental, Twisted, PyDispatcher, queuelib, scrapy
```

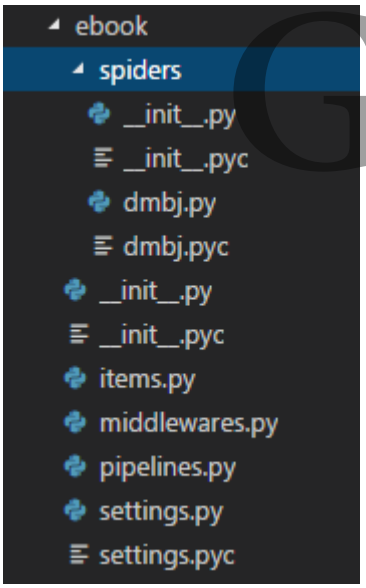
```
  Running setup.py install for pycparser ... done
  Running setup.py install for ipaddress ... done
  Running setup.py install for PyDispatcher ... done
Successfully installed Automat-0.6.0 PyDispatcher-2.0.5 Twisted-
17.9.0 asn1crypto-0.24.0 attrs-17.4.0 cffi-1.11.2 constantly-
15.1.0 cryptography-2.1.4 cssselect-1.0.3 enum34-1.1.6 hyperlink-
17.3.1 idna-2.6 incremental-17.5.0 ipaddress-1.0.19 lxml-4.1.1
parsel-1.3.1 pyOpenSSL-17.5.0 pyasn1-0.4.2 pyasn1-modules-0.2.1
pycparser-2.18 queuelib-1.4.2 scrapy-1.5.0 service-identity-
17.0.0 six-1.11.0 w3lib-1.18.0 zope.interface-4.4.3
```

2.3 新建 Scrapy 爬虫项目

Scrapy 是 Python 程序，同时也是一套框架，提供了一系列工具来简化开发，因此我们按照 Scrapy 的模式来开发，先新建一个 Scrapy 项目，如下：

```
scrapy startproject ebook
```

Scrapy 项目包含一些基础框架代码，我们在次基础上开发，目录结构类似下面这样子：



2.4 新建 Scrapy 爬虫

这个时候，Scrapy 还不知道我们要爬取什么数据，所以我们要用 Scrapy 工具新建一个爬虫，命令如下：

```
scrapy genspider example example.com
```

下面实操，我们在起点中文网上找一篇免费小说的完本，这里选择是修真小主播

我们就在前面建立的Scrapy项目ebook下新建一个*修真小主播*的爬虫，命令如下：

```
cd ebook
scrapy genspider xzxzb qidian.com
```

执行成功之后，在项目的 spider 目录下就多了一个**xzxzb.py**的文件。

2.5 爬虫思路

怎么抓取数据，首先我们要看从哪里取，打开修真小主播的页面，如下：

有个目录页签，点击这个页签可以看见目录，使用浏览器的元素查看工具，我们可以定位到目录和每一章节的相关信息，根据这些信息我们就可以爬取到具体的页面：

2.6 获取章节地址

现在我们打开 xzxzb.py 文件，就是我们刚刚创建的爬虫：


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


class XzxzbSpider(scrapy.Spider):
    name = 'xzxzb'
    allowed_domains = ['qidian.com']
    start_urls = ['http://qidian.com/']

    def parse(self, response):
        pass
```

start_urls 就是目录地址，爬虫会自动爬这个地址，然后结果就在下面的parse中处理。现在我们就来编写代码处理目录数据，首先爬取小说的主页，获取目录列表：

```
    def parse(self, response):
        pages = response.xpath('//div[@id="j-
catalogWrap"]//ul[@class="cf"]/li')
        for page in pages:
            url =
page.xpath('./child::a/attribute::href').extract()
            print url
        pass
```

获取网页中的DOM数据有两种方式，一种是使用CSS选择子，另外一种是使用XML的xPath查询。这里我们用xPath，相关知识请自行学习，看以上代码，首先我们通过ID获取目录框，获取类cf获取目录列表：

```
pages = response.xpath('//div[@id="j-
catalogWrap"]//ul[@class="cf"]/li')
```

接着，遍历子节点，并查询li标签内a子节点的href属性，最后打印出来：

```
for page in pages:
            url =
page.xpath('./child::a/attribute::href').extract()
            print url
```

这样，可以说爬取章节路径的小爬虫就写好了，使用如下命令运行xzxzb爬虫查看结果：

```
scrapy crawl xzxzb
```

这个时候我们的程序可能会出现如下错误：

> ...
> ImportError: No module named win32api
> ...

运行下面的语句即可：

```
pip install pypiwin32
```

屏幕输出如下：

```
> ...
>
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/wrrduN6auIlO
BDFlr9quQA2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/Jh-
J5usgyW62uJcMpdsVgA2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/5YXHdBvg1Ima
GfXRMrUjdw2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/fw5EBeKat-
76ItTi_ILQ7A2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/KsFh5VutI6Pw
rjbX3WA1AA2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/-
```

```
mpKJ01gPp1p4rPq4Fd4KQ2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/MlZSeYOQxSPM
5j8_3RRvhw2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/5TXZqGvLi-
3M5j8_3RRvhw2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/sysD-
JPiugv4p8iEw--PPw2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/xGckZ01j64-
aGfXRMrUjdw2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/72lHOJcgmedO
BDFlr9quQA2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/cZkHZEYnPl22
uJcMpdsVgA2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/vkNh45O3JsRM
s5iq0oQwLQ2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/ge4m8RjJyPH6
ItTi_ILQ7A2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/Y33PuxrKT4dp
4rPq4Fd4KQ2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/MDQznkrkiyXw
rjbX3WA1AA2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/A2r-
YTzWCYj6ItTi_ILQ7A2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/Ng9CuONRKei2
uJcMpdsVgA2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/Q_AxWAge14pM
s5iq0oQwLQ2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/ZJshvAu8TVVp
4rPq4Fd4KQ2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/hYD2P4c5UB2a
GfXRMrUjdw2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/muxiWf_jpqTg
n4SMoDUcDQ2']
[u'//read.qidian.com/chapter/MuRzJqCY6MyoLoerY3WDhg2/OQQ5jbADJjVp
4rPq4Fd4KQ2']
> ...
```


爬取章节路径的小爬虫就写好了，但我们的目的不仅于此，我们接下来使用这些地址来抓取内容：

2.7 章节页面分析

我们接下来分析一下章节页面，从章节页面我们要获取标题和内容。
如果说章节信息爬取使用的 parser 方法，那么我们可以给每一个章节内容的爬取写一个方法，比如：parser_chapter，先看看章节页面的具体情况：

可以看到，章节的整个内容在类名为 main-text-wrap 的 div 标签内，标题是其中类名为 j_chapterName 的h3标签，具体内容是类名为 read-content j_readContent 的 div 标签。我们试着把这些内容打印出来：


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


class XzxzbSpider(scrapy.Spider):
    name = 'xzxzb'
    allowed_domains = ['qidian.com']
    start_urls = ['https://book.qidian.com/info/1010780117/']

    def parse(self, response):
        pages = response.xpath('//div[@id="j-
catalogWrap"]//ul[@class="cf"]/li')
        for page in pages:
            url =
page.xpath('./child::a/attribute::href').extract_first()
            # yield scrapy.Request('https:' + url,
callback=self.parse_chapter)
            yield response.follow(url,
callback=self.parse_chapter)
        pass

    def parse_chapter(self, response):
        title = response.xpath('//div[@class="main-text-
wrap"]//h3[@class="j_chapterName"]/text()').extract_first().strip
```

```
()
        content = response.xpath('//div[@class="main-text-
wrap"]//div[@class="read-content
j_readContent"]').extract_first().strip()
        print title
        # print content
        pass
```

上一步，我们获取到了一个章节地址，从输出内容来看是相对路径，因此我们使用了 `yield response.follow(url, callback=self.parse_chapter)` ，第二个参数是一个回调函数，用来处理章节页面，爬取到章节页面后我们解析页面和标题保存到文件。

```
next_page = response.urljoin(url)
yield scrapy.Request(next_page, callback=self.parse_chapter)
```

scrapy.Request 不同于使用 response.follow ，需要通过相对路径构造出绝对路径，response.follow 可以直接使用相对路径，因此就不需要调用 urljoin 方法了；注意，response.follow 直接返回一个 Request 实例，可以直接通过 yield 进行返回。

数据获取了之后是存储，由于我们要的是html页面，因此，我们就按标题存储即可，代码如下：


```
    def parse_chapter(self, response):
        title = response.xpath('//div[@class="main-text-
wrap"]//h3[@class="j_chapterName"]/text()').extract_first().strip
()
        content = response.xpath('//div[@class="main-text-
wrap"]//div[@class="read-content
j_readContent"]').extract_first().strip()
        # print title
        # print content

        filename = './down/%s.html' % (title)
        with open(filename, 'wb') as f:
            f.write(content.encode('utf-8'))
        pass
```

至此，我们已经成功的抓取到了我们的数据，但还不能直接使用，需要整理和优化。

2.8 数据整理

首先，我们爬取下来的章节页面排序不是很好，如果人工去排需要太多的时间精力；另外，章节内容包含许多额外的东西，阅读体验不好，我们需要优化内容的排版和可读性。

我们先给章节排个序，因为目录中的章节列表是按顺序排列的，所以只需要给下载页面名称添加一个顺序号就行了。可是保存网页的代码是回调函数，顺序只是在处理目录的时候能确定，回调函数怎么能知道顺序呢？因此，我们要告诉回调函数它处理章节的顺序号，我们要给回调函数传参，修改后的代码是这样的：

```
    def parse(self, response):
        pages = response.xpath('//div[@id="j-
catalogWrap"]//ul[@class="cf"]/li')
        for page in pages:
            url =
page.xpath('./child::a/attribute::href').extract_first()
            idx = page.xpath('./attribute::data-
rid').extract_first()
            # yield scrapy.Request('https:' + url,
callback=self.parse_chapter)
            req = response.follow(url,
callback=self.parse_chapter)
            req.meta['idx'] = idx
            yield req
        pass

    def parse_chapter(self, response):
        idx = response.meta['idx']
        title = response.xpath('//div[@class="main-text-
wrap"]//h3[@class="j_chapterName"]/text()').extract_first().strip
()
        content = response.xpath('//div[@class="main-text-
wrap"]//div[@class="read-content
j_readContent"]').extract_first().strip()
        # print title
        # print content

        filename = './down/%s_%s.html' % (idx, title)
        cnt = '<h1>%s</h1> %s' % (title, content)
        with open(filename, 'wb') as f:
            f.write(cnt.encode('utf-8'))
        pass
```


不知道大家注意到没有，前面的分析中目录已经提供了一个 data_rid 可以作为排序号，我们在目录分析页面获取这个序号，然后通过 request 的 meta 传入 parse_chapter ，在 parse_chapter 中通过 response 的 meta 获取传入的参数，然后文件名中加入这个顺序好完成了排序。

另外，Sigil找那个通过H1标签来生成目录，需要目录的话，我们需要给内容添加一个h1标签

还有可读性差的问题，也许我们下载的网页可能会包含一些乱七八糟的东西，我们有很多办法，也可以使用readbility等第三方库，这里就不深入了。

3 使用 Sigil 制作电子书

电子书的制作，完全就是工具的应用，非常简单，这里把流程过一下，大家根据兴趣自行深入。

3.1 Sigil简介

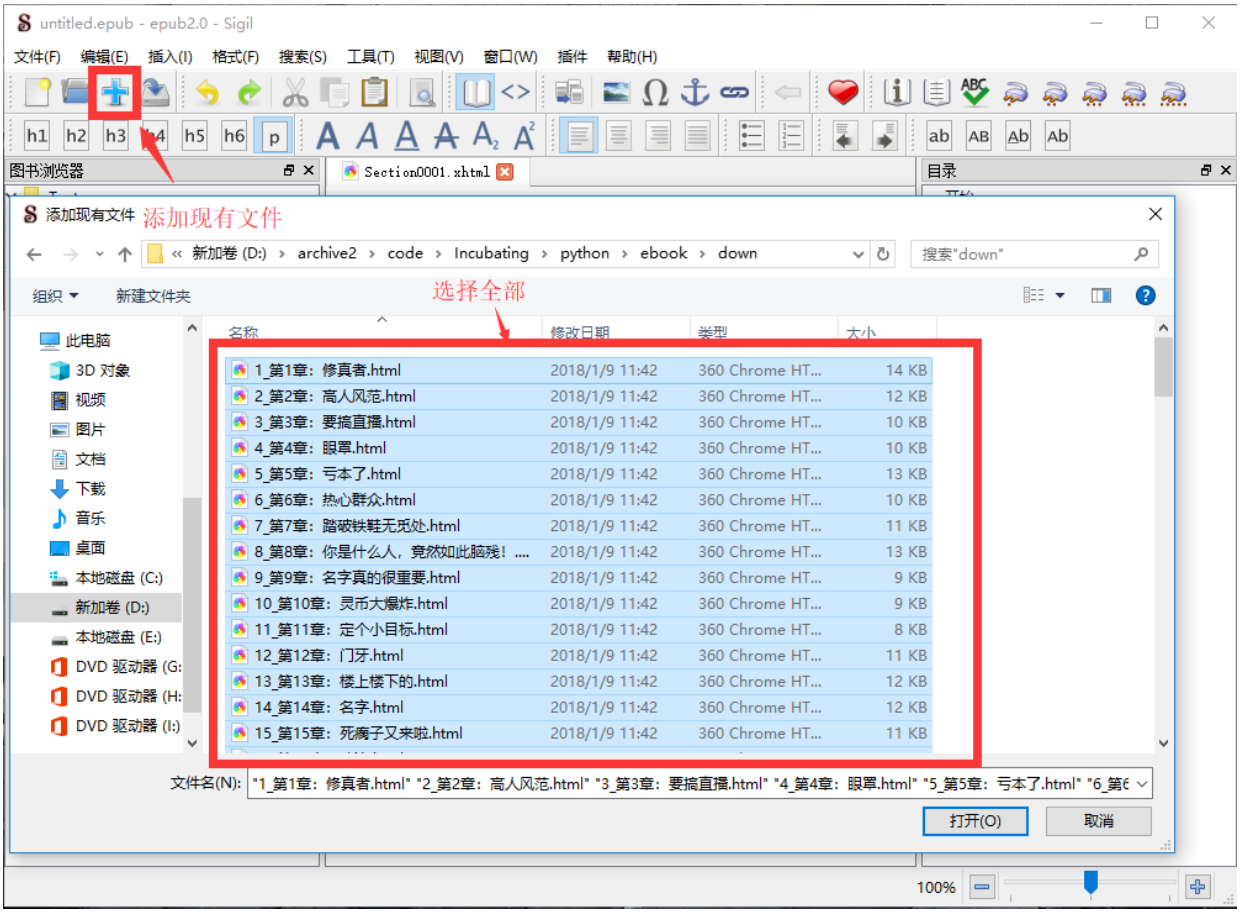
Sigil 是一个多平台的ePub电子书编辑器。官方网站，下载页面在GitHub，根据自己的需求下载，安装很简单就不啰嗦了。

3.2 ePub 电子书简介

ePub（Electronic Publication 的缩写，意为：电子出版），是一个自由的开放标准，属于一种可以“自动重新编排”的内容；也就是文字内容可以根据阅读设备的特性，以最适于阅读的方式显示。ePub 档案内部使用了 XHTML 或 DTBook（一种由 DAISY Consortium 提出的 XML 标准）来展现文字、并以 zip 压缩格式来包裹档案内容。EPub 格式中包含了数位版权管理（DRM）相关功能可供选用。

3.3 加载 html 文件

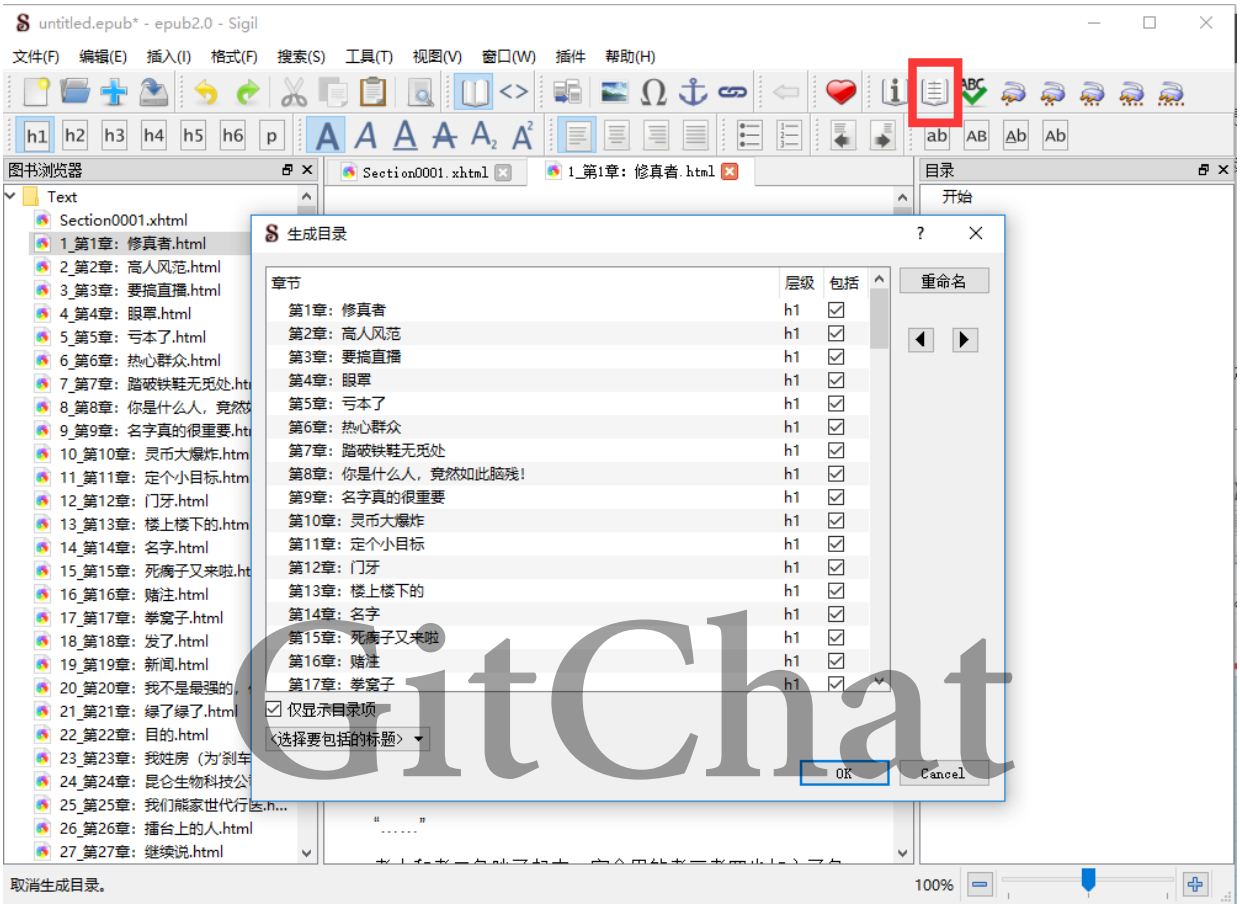
要制作 ePub 电子书，我们首先通过 Sigil 把我们的抓取的文件加载到程序中，在添加文件对话框中我们全选所有文件：



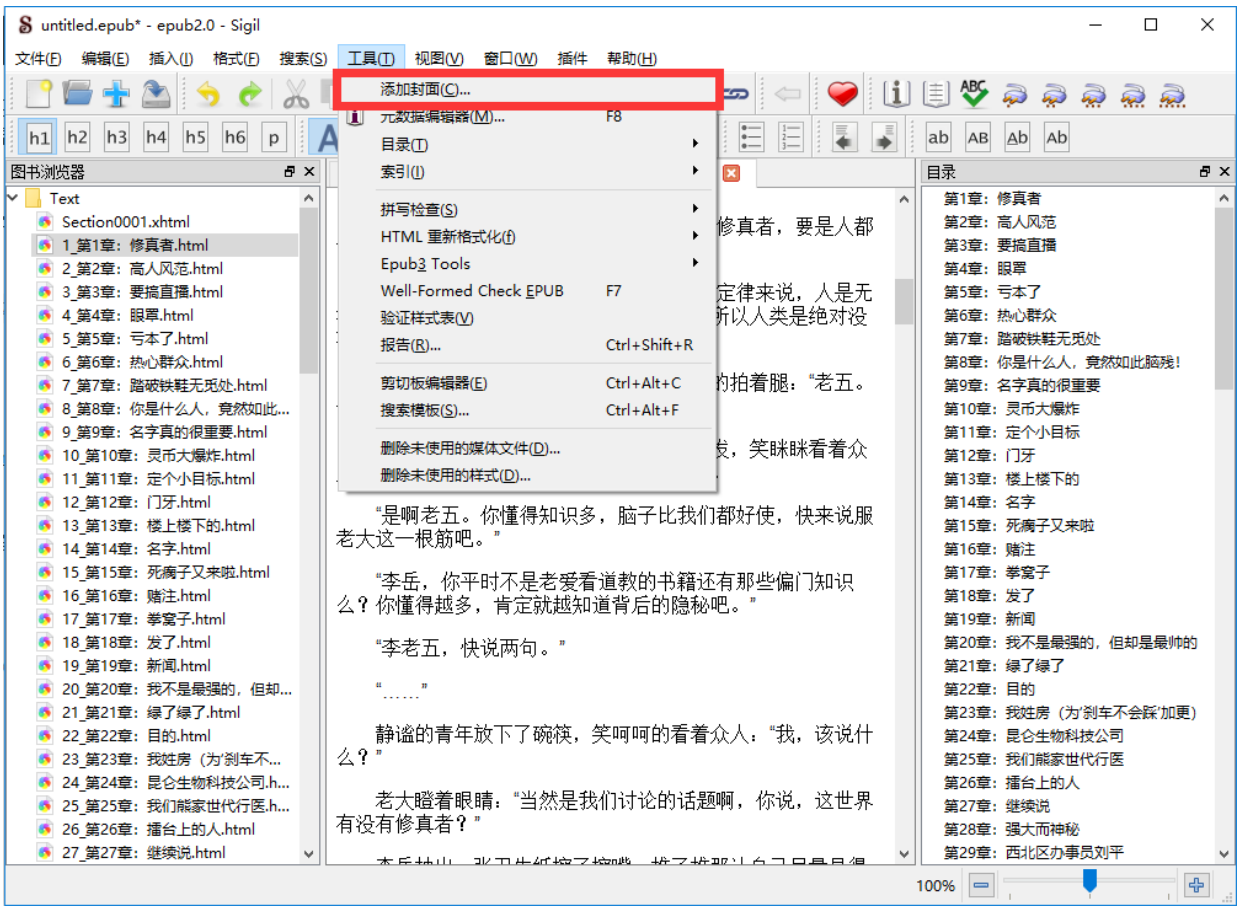
内容都是 HTML 文件，所以编辑、排版什么的学习下 HTML。

3.4 制作目录

文件中存在 HTML 的 h 标签时，点击生成目录按钮就可以自动生成目录，我们在前面数据抓取时已经自动添加了h1标签：



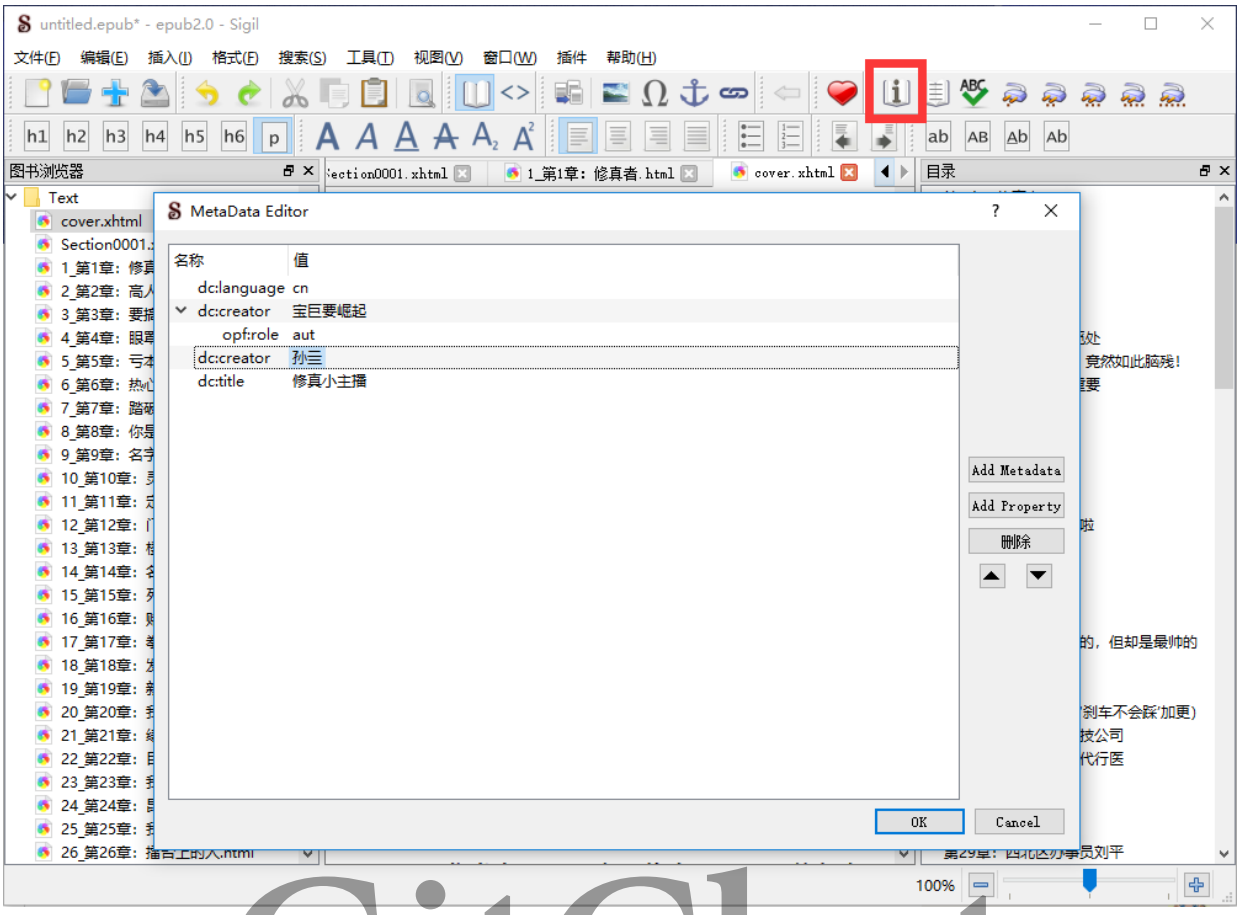
3.5 制作封面



封面本质上也是 HTML，可以编辑，也可以从页面爬取，就留给大家自己实现吧。

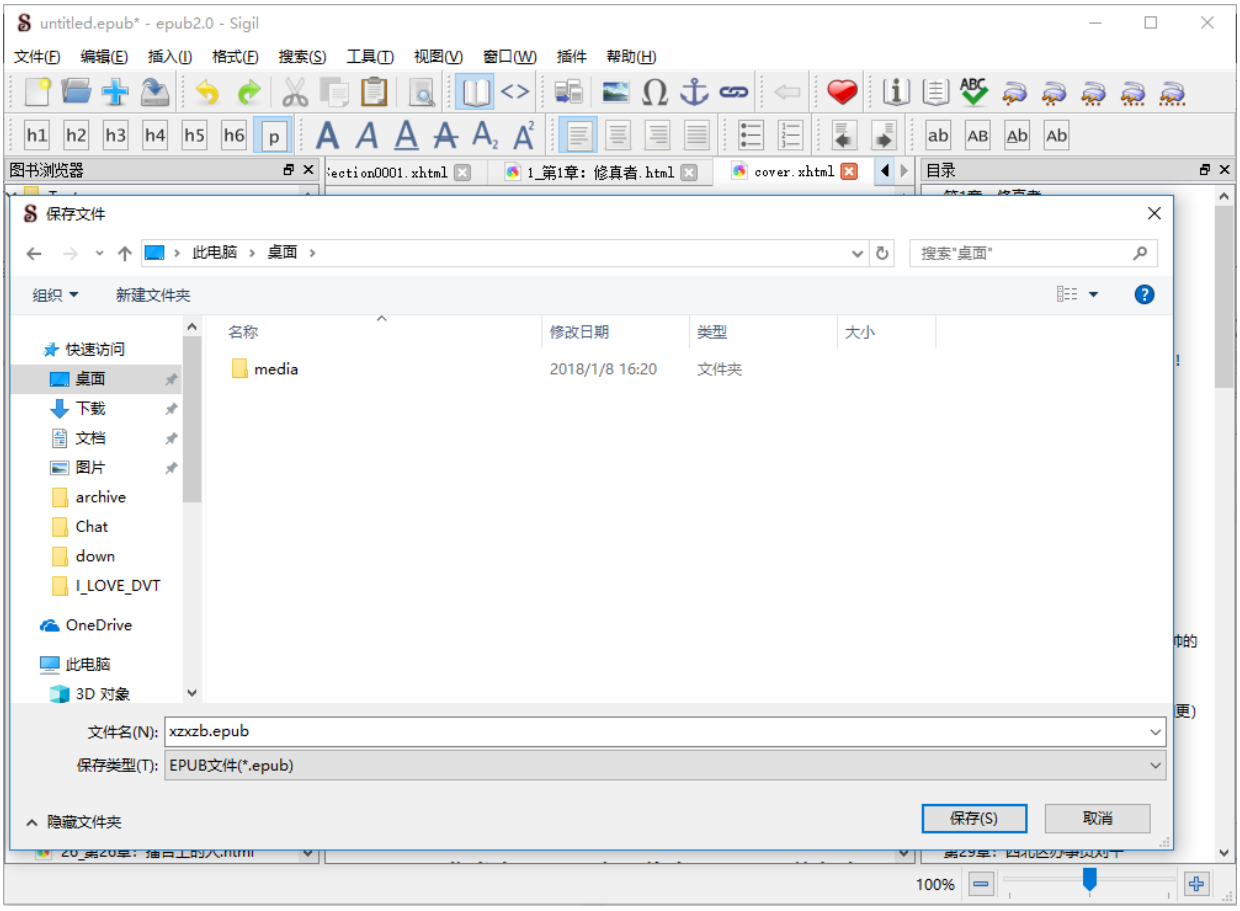
3.6 编辑元数据

编辑书名、作者等信息：

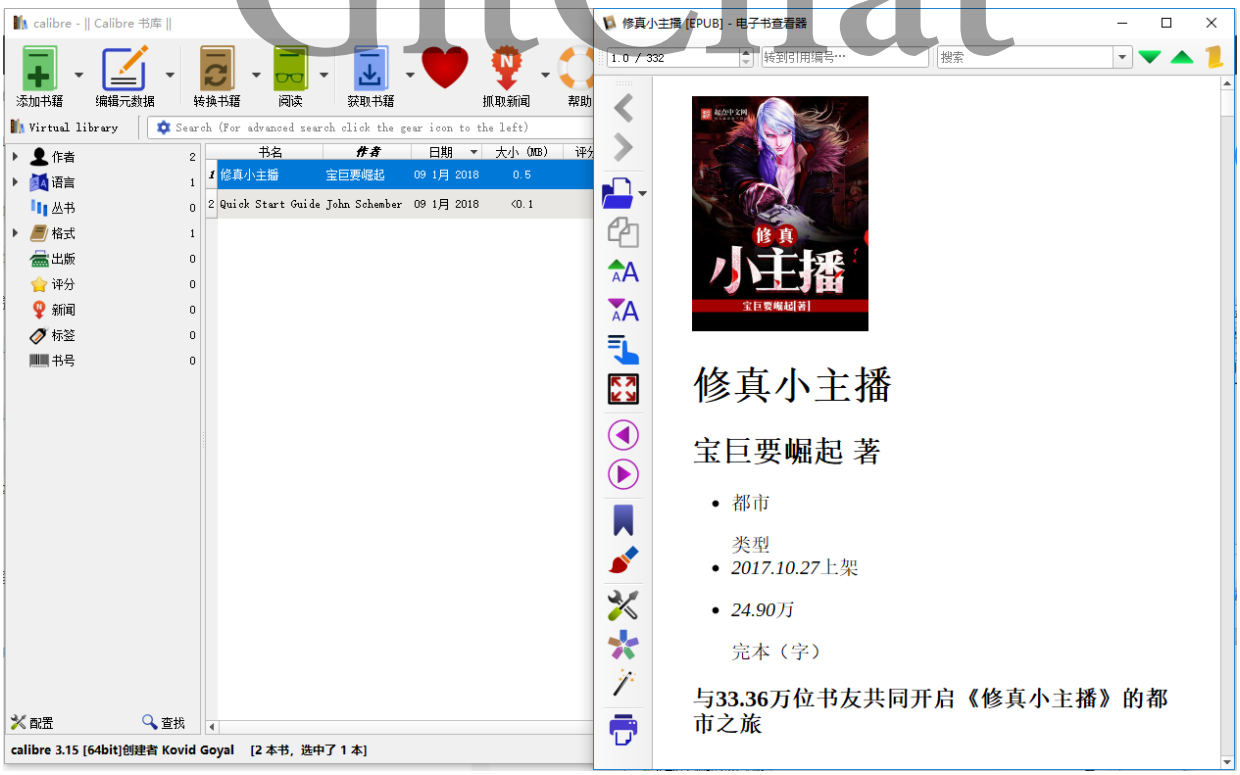


3.6 输出ePub

编辑完成后保存，取个名字：



输出可以使用电子书阅读软件打开查看，我用的是 Calibre，还可以方便的转换为相应的格式装到 Kindle 中阅读。



整个过程就结束了，文章内代码提交到码云，接下来自由发挥，请开始你的表演。

参考资料

爬虫 Scrapy 学习系列之一：Tutorial