



广西师范大学

大数据采集与融合技术课程 设计报告

题 目：飞卢小说书库-小说信息的爬取

学院（部）：计算机科学与工程学院/软件学院

专 业：数据科学与大数据技术

年 级：2021 级

学 号：202112304049

学生姓名：刘玉镇

任课教师：甘炎灵

完成日期：2022 年 6 月 20 日

目录

题目：飞卢小说书库-小说信息的爬取	2
一、 选题背景	2
二、数据采集分析和总体方案设计	2
(一) 数据采集分析	2
2.1robots 协议的分析	2
2.2 飞卢小说网站的分析	3
2.3 网页结构特征分析	4
2.4 提取数据特征的分析	5
2.5 数据清晰方法分析	5
2.6 存储数据方法分析	6
(二) 总体方案设计	7
2.7 爬虫的基本思路	7
2.8Scrapy 框架爬虫基本步骤	8
2.9 算法的实现分析	8
三、爬虫程序详细设计和实现	9
(一) 实验环境及配置	9
3.1 操作系统	9
3.2 集成开发环境	9
3.3Python 版本	9
3.4Scrapy 框架所使用到的库	10
(二) 实验的步骤、分析和运行结果	10
3.5 完整的实验步骤与分析	10
3.6 关键技术方法的使用	17
3.7 实验运行结果	18
四、总结	19

题目：飞卢小说书库-小说信息的爬取

一、选题背景

随着互联网的快速发展，人们对于信息获取的需求越来越高。小说作为人们喜爱的文学形式之一，其阅读需求也日益增长。在众多小说网站中，飞卢小说以其海量的书库和优质的服务吸引了大量的读者。然而，飞卢小说的书库信息对于读者来说并不可见，读者需要手动搜索并选择自己感兴趣的小说。因此，本课题旨在通过爬取飞卢小说书库中的小说信息，将其整理成一个小说信息数据库，为用户提供更加便捷的小说阅读体验。

主要爬取小说的小说名、小说类型、作者、出版时间、评分、阅读量、最新章节标题以及小说简介。

二、数据采集分析和总体方案设计

（一）数据采集分析

通过分析小说的标题、作者、更新时间等字段，来了解当前小说网站上的热门小说类型、作者分布、更新速度等情况，为读者提供更加全面的阅读指导。使用 scrapy 爬虫框架完成爬取飞卢小说书库的信息，比如：小说名，小说类型，作者，出版时间，评分，阅读量，最新章节标题以及小说简介等信息的爬取。在 settings 文件中全局配置 robots 协议，并且遵守 robots 协议，将其设置为 True。

2.1 robots 协议的分析

Robots.txt 是一种协议，用于在网站中定义网络爬虫（也称为机器人或爬行者）可以访问和抓取的内容。它是一个文本文件，通常位于网站根目录下，并使用 .txt 扩展名。

Robots.txt 文件包含一系列指令，称为 robots exclusion rules，用于指定机器人应该遵循的规则。这些规则可以指定哪些页面可以被抓取，哪些页面不能被抓取，哪些页面需要使用特定的抓取权限等。

在使用网络爬虫抓取网站内容时，机器人应该尊重 Robots.txt 协议，遵循其中的规则。如果机器人不遵循 Robots.txt 协议，网站所有者可能会认为这是一种恶意行为，并采取相应的措施来阻止该机器人的访问。

需要注意的是，Robots.txt 只是一种约定俗成的协议，并非法律文件，因此如果违反了 Robots.txt 协议，网站所有者并不能直接采取法律行动。但是，如果机器人抓取网站内容时违反了 Robots.txt 协议，可能会导致网站所有者采取技术措施来阻止该机器人的访问。

一个好的爬虫不仅需要遵守 robots 协议，还得设计一个爬虫代理，防止过度爬虫导致 IP 被封，同时给服务器带来一个有好的爬虫环境。robots 协议通过 Allow 与 Disallow 的搭配使用，对爬虫的抓取实行限制或放行。在网址搜索框输

入” `https://b.faloo.com/robots.txt`”则可以查看豆瓣电影网站的 `robots.txt` 的相关信息。相关信息如下：



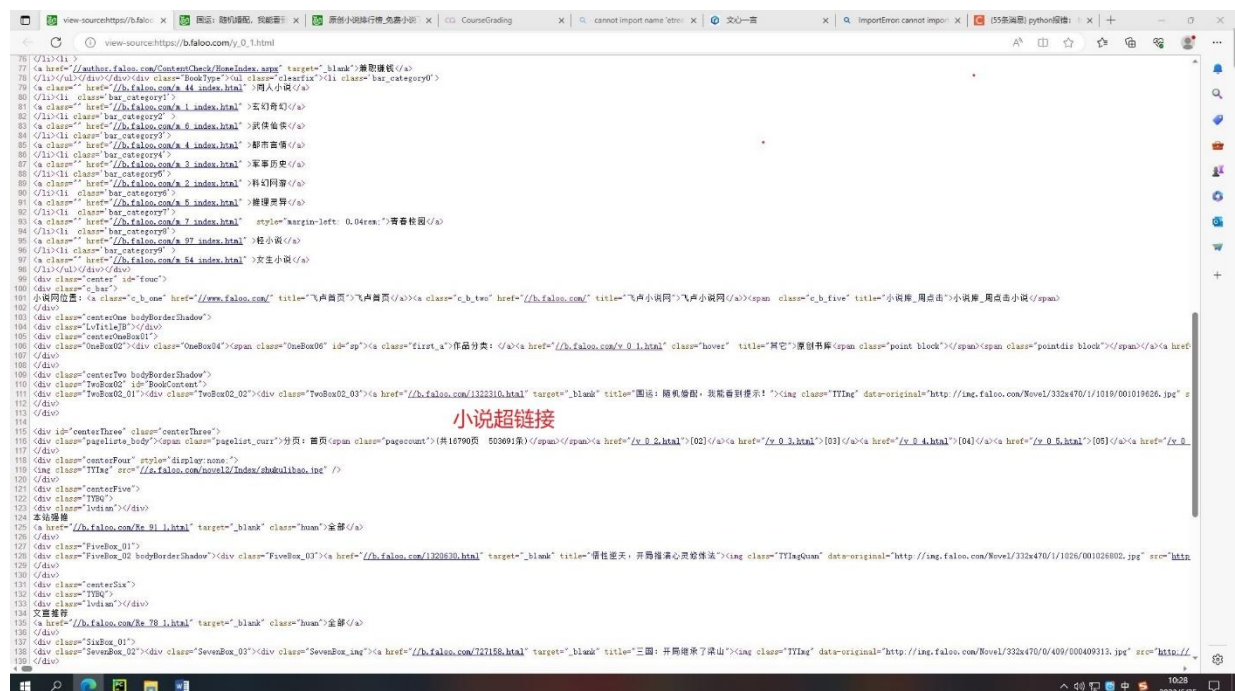
2.2 飞卢小说网站的分析

1. 页面跳转：为了避免弹出窗口对用户的干扰，建议在页面跳转时使用 HTML 链接或 JavaScript 异步加载的方式，而不是使用弹出窗口。这样可以让用户在不离开当前页面的情况下，快速获取所需内容。
2. 页面长度：飞卢小说网站的每个页面非常短，一般两屏，不超过三屏，这样的设计可以让用户更快地获取信息，更容易地阅读和理解内容。因此，建议在页面设计时，尽量保持页面的简短，避免冗长和不必要的页面。
3. 按钮设计：为了提高用户体验，建议在页面上使用明确的按钮，避免使用过多的按钮或复杂的按钮设计。同时，按钮的标签和意义应该清晰明了，易于理解。
4. 布局：飞卢小说网站采用上中下的布局方式，最上面为导航，中间为左右两列的方式，左边为页面核心内容，右面为辅助、相关功能及信息，下面是站务导航。这种布局方式整洁清晰，易于用户理解和操作。建议在页面设计时，采用类似的布局方式，保持页面的统一性和可读性。
5. 技术栈：为了提高页面的性能和可维护性，建议在页面技术栈的选择上，考虑使用现代化的前端框架和工具，例如 React、Vue 等。同时，为了提高页面的可访问性和兼容性，页面使用 `div+css` 的页面技术，避免使用过多的 `table` 和 `font` 等标签。



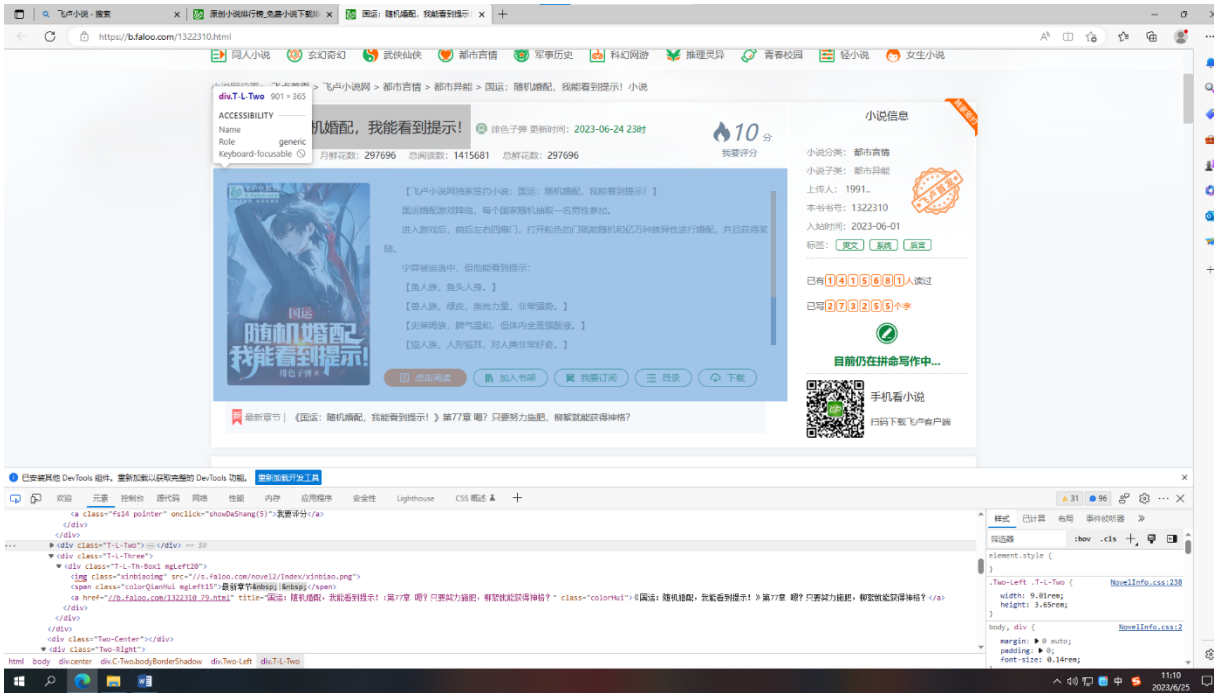
2.3 网页结构特征分析

在飞卢小说书库这个页面，页面排版清晰而干净，一个页面有 30 篇小说的排行信息，页面底端设计有翻页按钮，方便用户翻页，节省时间，拥有更好的体验感，属于列表型页面。每一篇小说的超链接都放在标签<div class="TwoBox02_03">中的<a>内，小说的详细内容都在小说对应的超链接中，所以在爬取小说详细内容的时候需要跳转到相关页面进行爬取，在爬取对应的标签的 text 文件，即小说对应的小说名、小说类型、作者等信息，每一部小说的爬取步骤都是先提取每一部小说的超链接，然后请求超链接，跳转至超链接的页面，对小说的相关信息信息进行爬取。



2.4 提取数据特征的分析

利用网页的开发者工具查看数据的分布以及结构，小说的超链接都是放在 `a` 标签的 `href` 属性中，小说名都放在 `h1` 标签的文本中，作者、最新章节标题都放在 `a` 标签的文本中，小说分类、出版时间、阅读量、评分放在 `span` 标签中，小说简介的都放在 `p` 标签下，但在不同的 `div` 标签下。数据特征信息明了，便于我们去分析以及理解。我们可以在指定的页面使用开发者工具查看页面的布局以及元素特征。以飞卢小说的《国运：随机婚配，我能看到提示》为例，如下图所示：



2.5 数据清晰方法分析

XPath (XML Path Language) 是一种用于在 XML 文档中查找和定位信息的语言。在爬虫中，XPath 选择器是一种常用的技术，用于从 HTML 或 XML 文档中提取数据。XPath 选择器可以帮助我们根据元素的位置、属性、文本内容等来选择元素。通过使用 XPath 表达式，可以轻松地定位和提取所需的元素和数据，而无需手动查找和复制粘贴元素 ID 或类名等。使用 Scrapy 框架中的 XPath 表达式从 HTML 页面中提取数据，并将数据存储在一个字典类型的变量 `item` 中。该函数的目的是解析小说的评论页面，并提取小说的名称、风格、作者、出版日期、评分、阅读次数、新章节标题和简介等信息。该函数接受一个参数 `response`，表示要解析的 HTML 页面。该函数返回一个包含所有提取信息的字典变量 `item`。

```

60     def parse_comment(self, response):
61         item = CommentItem()
62
63         Novel = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[1]/div[1]/h1/text()').extract()[0]
64         Novel_style = response.xpath('/html/body/div[3]/div[2]/div[5]/div[1]/div[2]/div[1]/span/span/a/text()').extract()[0]
65         Author = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[1]/div[1]/a/text()').extract()[0]
66         Publication_date = response.xpath('/html/body/div[3]/div[2]/div[5]/div[1]/div[2]/div[5]/span/span/text()').extract()[0]
67         Score = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[2]/span[1]/text()').extract()[0]
68         Read_times = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[1]/div[2]/span[3]/span/text()').extract()[0]
69         New_page = response.xpath('/html/body/div[3]/div[2]/div[3]/div[3]/div/a/text()').extract()[0]
70         # Introduction = response.xpath('//div[@class = "T-L-T-C-Box1"]/p/text()').extract()[0] + response.xpath('//div[@class = "T-L-
71         # print(Introduction)
72         p = response.xpath('//div[@class = "T-L-T-C-Box1"]/p')
73         n = len(p)
74         Introduction = ''
75         for i in range(1, n - 1):
76             Introduce = response.xpath('//div[@class = "T-L-T-C-Box1"]/p/text()').extract()[i]
77             Introduction = Introduction + '\n' + Introduce
78
79         item['Novel'] = Novel
80         item['Novel_style'] = Novel_style
81         item['Author'] = Author
82         item['Publication_date'] = Publication_date
83         item['Score'] = Score
84         item['Read_times'] = Read_times
85         item['New_page'] = New_page
86         item['Introduction'] = Introduction
87
88         return item

```

2.6 存储数据方法分析

本次实验将爬取的数据保存在本地文本文件当中，以追加的方式（a，表示以追加模式打开文件，如果文件不存在，则创建文件。）写入文本文件，便于我们将爬取到完整的信息，以及多次爬取获得的信息有哪些变化，都可以在文件中观察可得。为保障数据的正确性和对应性，我们采取了以字典的方式进行写入。

```

43     with open(f'第{self.file_number+1}页.txt', 'a', encoding='utf-8') as f:
44         # file_path = item['Novel']+'.txt'
45         # f = open(file_path, "wb")
46         f.write('小说名: ' + str(Novel) + '\n')
47         f.write('小说类型: ' + str(Novel_style))
48         f.write('作者: ' + str(Author) + '\n')
49         f.write('出版时间: ' + str(Publication_date) + '\n')
50         f.write('评分: '+str(Score)+'分'+'\n')
51         f.write('总阅读人数: '+str(Read_times)+'\n')
52         f.write('最新章节: '+str(New_page)+'\n')
53         f.write('简介: ' + str(Introduction) + '\n')
54         f.write('\n')
55

```

图 1 将数据保存为文本文件的代码

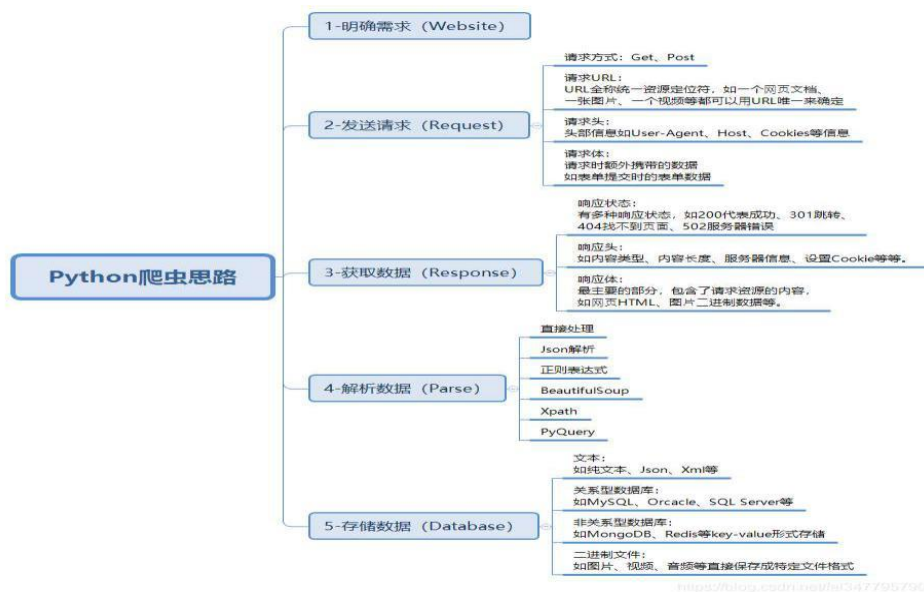


图 2 保存在文件内容部分截图

（二）总体方案设计

2.7 爬虫的基本思路

无论做什么事情我们都会有一个大概的框架，爬虫也不例外，了解爬虫的思路是我们着手爬虫的第一步。网络爬虫框架主要由控制器、解析器和索引库三大部分组成，而爬虫工作原理主要是解析器这个环节，解析器的主要工作是下载网页，进行页面的理，主要是将一些 JS 脚本标签、CSS 代码内容、空格字符、HTML 标签等内容处理掉，爬虫的基本工作是由解析器完成。以下是 python 爬虫的基本流程图，可以帮助我们快速的找到爬虫的入手点：



2.8 Scrapy 框架爬虫基本步骤

- 1) 爬虫中起始的 URL 构造成 Requests 对象 \Rightarrow 爬虫中间件 \Rightarrow 引擎 \Rightarrow 调度器;
- 2) 调度器把 Requests \Rightarrow 引擎 \Rightarrow 下载中间件 \Rightarrow 下载器;
- 3) 下载器发送请求, 获取 Responses 响应 \Rightarrow 下载中间件 \Rightarrow 引擎 \Rightarrow 爬虫中间件 \Rightarrow 爬虫;
- 4) 爬虫提取 URL 地址, 组装成 Requests 对象 \Rightarrow 爬虫中间件 \Rightarrow 引擎 \Rightarrow 调度器, 重复步骤 2;
- 5) 爬虫提取数据 \Rightarrow 引擎 \Rightarrow 管道处理和保存数据。

2.9 算法的实现分析

本次实验是从 "https://b.faloo.com/y_0_"+str(i)+".html" (其中 i 的范围为 1-101) 网页为入口爬取电影信息。

```
class DemoSpider(scrapy.Spider):
    name = "demo" #定义唯一名称, 与spider区别开来, 爬虫项目名称
    allowed_domains = ["b.faloo.com"] #爬取的域名

    finish_url = []
    scrawl_urls = []
    for i in range(1,2):
        scrawl_url = []
        start_urls = "https://b.faloo.com/y_0_"+str(i)+".html" #从哪个页面开始爬取
        # print(start_urls)
```

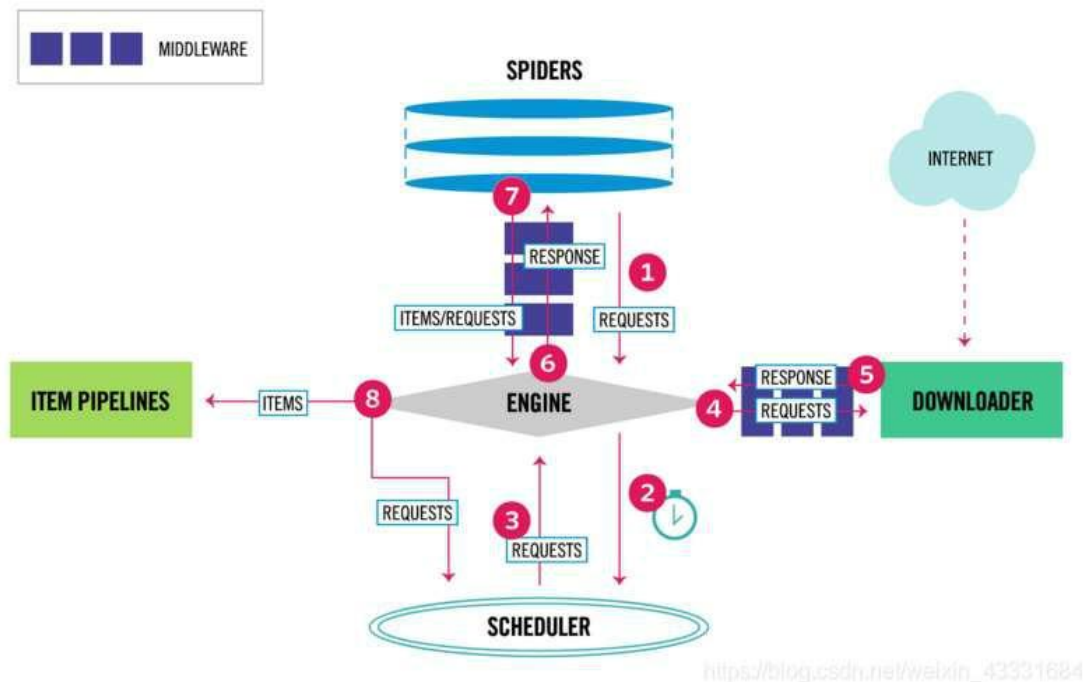
通过使用 **xpath** 技术爬取下一页的链接以及拼接技术模拟广度搜索算法。当然也可通过递归技术实现翻页, 但对于本次实验还是拼接技术比较可靠。

```
def start_requests(self):
    list = []
    while self.scrawl_urls.__len__():
        url = self.scrawl_urls.pop()
        self.finish_url.append(url)
        print(url)
        try:
            yield scrapy.Request(url=url, callback=self.parse_comment, dont_filter=True, headers=self.headers)
        except:
            print('出错!')
```

项目框架图:



Scrapy 框架实现流程图:



三、爬虫程序详细设计和实现

(一) 实验环境及配置

3.1 操作系统

Windows 10 家庭中文版 , 64 位操作系统, 基于 x64 的处理器

3.2 集成开发环境

pycharm 学习集成开发环境

3.3 Python 版本

Python 3.11

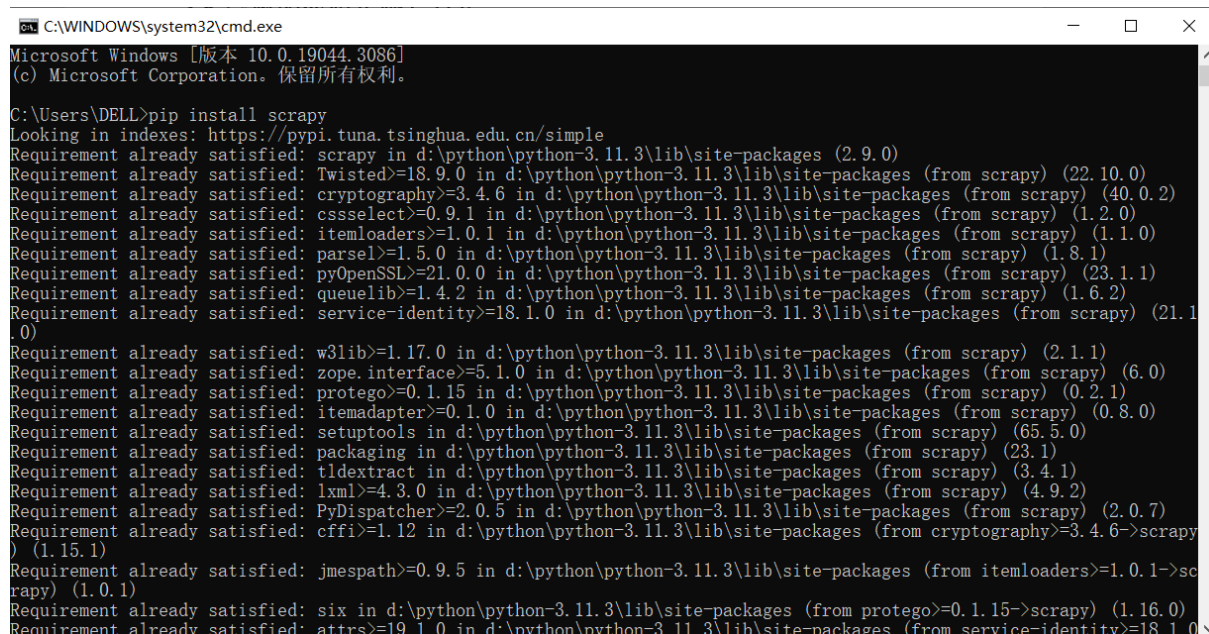
3.4 Scrapy 框架所使用到的库

- 1) satrt 文件: `from scrapy.cmdline import execute`、`import sys`、`import os`
- 2) demo 文件 (爬虫文件): `import scrapy` from `scrapy.selector` `import Selector` from `scrapy.http` `import Request`、`from myScrapy.items` `import DoubanItem`
- 3) Pipeline 文件: `import json`
- 4) Item 文件: `import scrapy`

(二) 实验的步骤、分析和运行结果

3.5 完整的实验步骤与分析

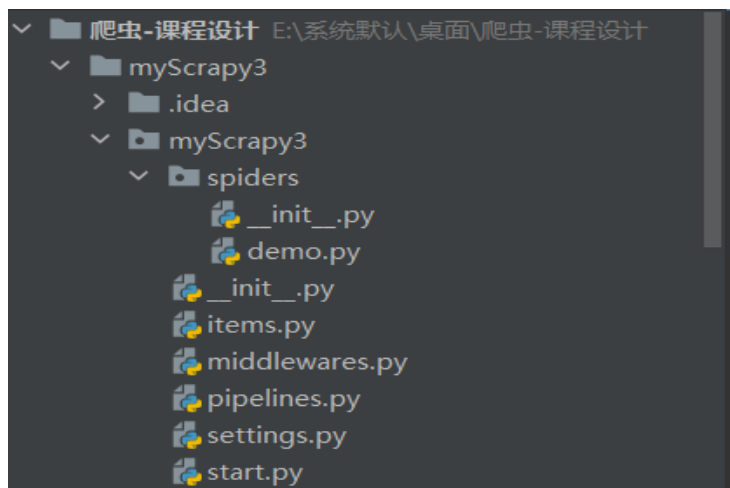
- (1) 安装 Scrapy 框架: 在命令行窗口运行指令: `pip install scrapy`, 如下图:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.19044.3086]
(c) Microsoft Corporation. 保留所有权利。

C:\Users\DELL>pip install scrapy
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Requirement already satisfied: scrapy in d:\python\python-3.11.3\lib\site-packages (2.9.0)
Requirement already satisfied: Twisted>=18.9.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (22.10.0)
Requirement already satisfied: cryptography>=3.4.6 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (40.0.2)
Requirement already satisfied: cssselect>=0.9.1 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (1.2.0)
Requirement already satisfied: itemloaders>=1.0.1 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (1.1.0)
Requirement already satisfied: parsel>=1.5.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (1.8.1)
Requirement already satisfied: pyOpenSSL>=21.0.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (23.1.1)
Requirement already satisfied: queuelib>=1.4.2 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (1.6.2)
Requirement already satisfied: service-identity>=18.1.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (21.1.0)
Requirement already satisfied: w3lib>=1.17.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (2.1.1)
Requirement already satisfied: zope.interface>=5.1.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (6.0)
Requirement already satisfied: protego>=0.1.15 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (0.2.1)
Requirement already satisfied: itemadapter>=0.1.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (0.8.0)
Requirement already satisfied: setuptools in d:\python\python-3.11.3\lib\site-packages (from scrapy) (65.5.0)
Requirement already satisfied: packaging in d:\python\python-3.11.3\lib\site-packages (from scrapy) (23.1)
Requirement already satisfied: tldextract in d:\python\python-3.11.3\lib\site-packages (from scrapy) (3.4.1)
Requirement already satisfied: lxml>=4.3.0 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (4.9.2)
Requirement already satisfied: PyDispatcher>=2.0.5 in d:\python\python-3.11.3\lib\site-packages (from scrapy) (2.0.7)
Requirement already satisfied: cffi>=1.12 in d:\python\python-3.11.3\lib\site-packages (from cryptography>=3.4.6->scrapy) (1.15.1)
Requirement already satisfied: jmespath>=0.9.5 in d:\python\python-3.11.3\lib\site-packages (from itemloaders>=1.0.1->scrapy) (1.0.1)
Requirement already satisfied: six in d:\python\python-3.11.3\lib\site-packages (from protego>=0.1.15->scrapy) (1.16.0)
Requirement already satisfied: attrs>=19.1.0 in d:\python\python-3.11.3\lib\site-packages (from service-identity>=18.1.0->scrapy) (23.1.0)
```

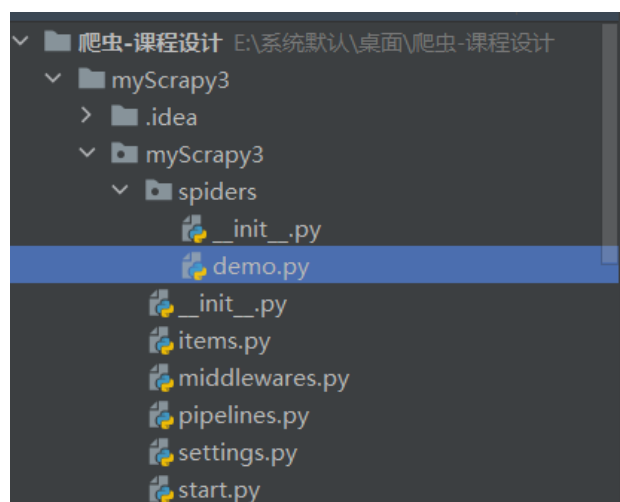
- (2) 创建 Scrapy 工程: 在命令行窗口, 通过 `cd` 指令, 切换到保存项目的位置。运行指令: `scrapy startproject 工程名`。假设工程名设置为 `myScrapy3`, 则运行 `scrapy startproject myScrapy3`, 这样就创建了一个 `myScrapy3` 工程, 如下图:



(3) 创建爬虫文件

1) 运行指令：cd myScrapy，转到 myScrapy 目录下。

2) 运行指令：scrapy genspider 爬虫名称 “url” 创建爬虫。其中，url 是要爬网站的网址。爬虫名称是 demo，飞卢小说书库的网址是：https://b.faloo.com/。那么运行的指令是：scrapy genspider demo “https://b.faloo.com/”。结果如下图（看蓝色部分，多了一个 demo 模块）：



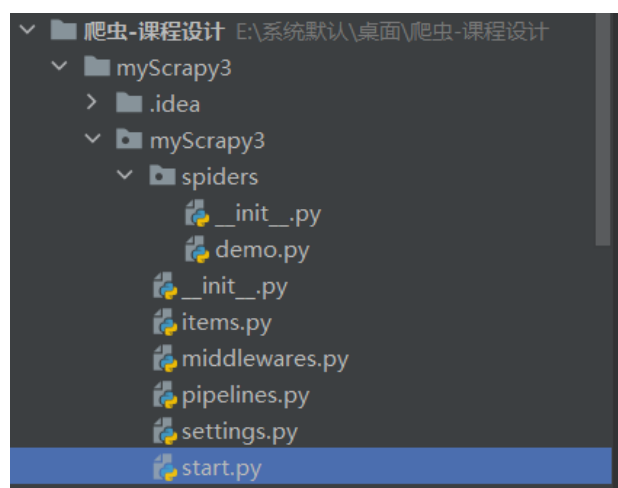
(4) demo 模块里面的内容如下（看蓝色部分，爬虫的名字是 demo）

```
8 class DemoSpider(scrapy.Spider):
9     name = "demo"           #定义唯一名称，与spider区别开来，爬虫项目名称
10    allowed_domains = ["b.faloo.com"]  #爬取的域名
11
12    finish_url = []
13    scrawl_urls = []
14    for i in range(1,2):...
15
16    def start_requests(self):...
17
18    def parse_comment(self, response):...
```

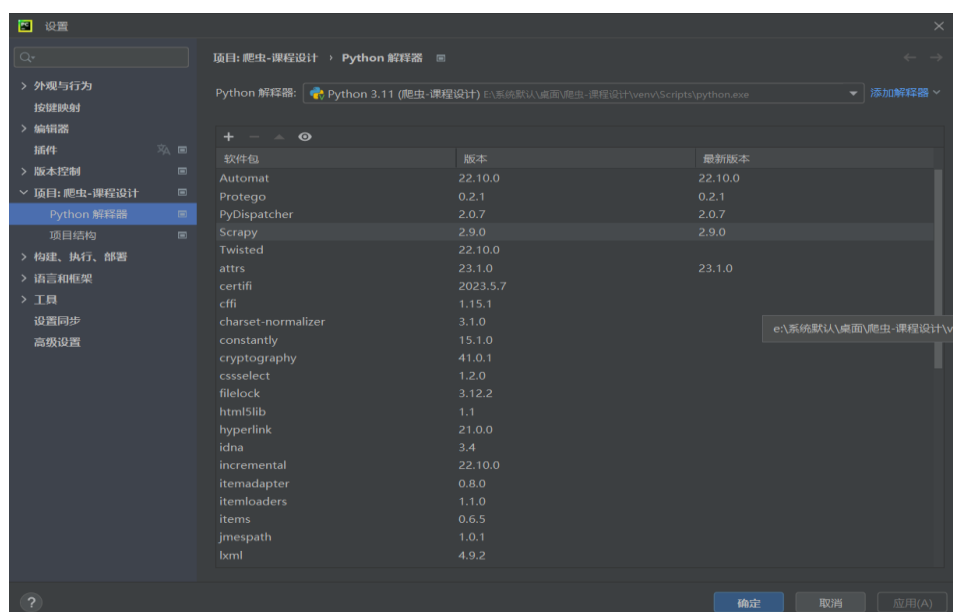
(5) 编写一个“start.py”，此文件要跟 **cfg** 文件在同一个目录下，因为这个文件的内容会用到 **cfg** 文件的功能，用来执行爬虫程序，如下图：

```
1 import os.path
2 from scrapy.cmdline import execute
3 import sys
4 import os
5 sys.path.append(os.path.dirname(__file__))
6 execute(["scrapy", "crawl", "demo"])
7
```

(6) “start.py”放的位置如下，直接运行“start.py”，就可以开始爬虫：



(7) 导入实验所需要的包，没有成功导入实验所需要的包，实验就不能成功运行。



(8) 在 **dome** 文件中爬虫逻辑程序编写，如下图：

```
import html5lib as html5lib
import requests
import scrapy
from ..items import CommentItem
from ..pipelines import Myscrapy3Pipeline

class DemoSpider(scrapy.Spider):
    name = "demo" #定义唯一名称, 与spider区别开来, 爬虫项目名称
    allowed_domains = ["b.faloo.com"] #爬取的域名

    finish_url = []
    scrawl_urls = []
    for i in range(1,2):
        scrawl_url = []
        start_urls = "https://b.faloo.com/v_0_"+str(i)+".html" #从哪个页面开始爬取
        # print(start_urls)
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/113.0.0.0'
        }

        res = requests.get(url=start_urls, headers=headers)
        res.encoding = res.apparent_encoding
        html_txt = res.text
        # print(res.encoding)
        # print(html_txt)

        content = html5lib.parse(html_txt, treebuilder="lxml", namespaceHTMLElements=False)
        # print(content)
        new_urls = content.xpath('//div[@class = "TwoBox02_03"]/a[@href and @target = "_blank" and @title]/@href')
        new_urls = set(new_urls)
        # print(new_urls)
        scrawl_url.extend(new_urls)
        # print(scrawl_url)
        for url in scrawl_url:
            url = 'https:' + url
            scrawl_urls.append(url)
            # print(scrawl_urls)

        ...

    def start_requests(self):
        list = []
        while self.scrawl_urls.__len__():
            url = self.scrawl_urls.pop()
            self.finish_url.append(url)
            print(url)
            try:
                yield scrapy.Request(url=url,callback=self.parse_comment,dont_filter=True,headers=self.headers)
            except:
                print('出错! ')

    def parse_comment(self,response):
        item = CommentItem()

        Novel = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[1]/div[1]/h1/text()').extract()[0]
        Novel_style = response.xpath('/html/body/div[3]/div[2]/div[5]/div[1]/div[2]/div[1]/span/span/a/text()').extract()[0]
        Author = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[1]/div[1]/a/text()').extract()[0]
        Publication_date = response.xpath('/html/body/div[3]/div[2]/div[5]/div[1]/div[2]/div[5]/span/span/text()').extract()[0]
        Score = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[2]/span[1]/text()').extract()[0]
        Read_times = response.xpath('/html/body/div[3]/div[2]/div[3]/div[1]/div[1]/div[2]/span[3]/span/text()').extract()[0]
        New_page = response.xpath('/html/body/div[3]/div[2]/div[3]/div[3]/div[3]/div/a/text()').extract()[0]
        # Introduction = response.xpath('//div[@class = "T-L-T-C-Box1"]/p/text()').extract()[0] + response.xpath('//div[@class = "T-L-T-C-Box1"]/p/text()').extract()[1]
        # print(Introduction)
        p = response.xpath('//div[@class = "T-L-T-C-Box1"]/p')
        n = len(p)
        Introduction = ''
        for i in range(1, n - 1):
            Introdunct = response.xpath('//div[@class = "T-L-T-C-Box1"]/p/text()').extract()[i]
            Introduction = Introduction + '\n' + Introdunct
        item['Novel'] = Novel
        item['Novel_style'] = Novel_style
        item['Author'] = Author
        item['Publication_date'] = Publication_date
        item['Score'] = Score
        item['Read_times'] = Read_times
        item['New_page'] = New_page
        item['Introduction'] = Introduction

        return item
```

(9) itmes 文件的设置，定义爬取数据持久化的数据结构，如下图：

```
import scrapy

class Myscrapy3Item(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    pass

class CommentItem(scrapy.Item):

    Novel = scrapy.Field()
    Novel_style = scrapy.Field()
    Author = scrapy.Field()
    Publication_date = scrapy.Field()
    Score = scrapy.Field()
    Read_times = scrapy.Field()
    New_page = scrapy.Field()
    Introduction = scrapy.Field()
```

(10) pipelines.py:管道,持久化存储相关,用来将 items 的模型存储到本地磁盘。

```
class Myscrapy3Pipeline:
    def __init__(self):
        self.counter = 0
        self.file_number = 0

    def process_item(self, item, spider):

        self.counter += 1

        if self.counter == 31:
            self.close_file()
            self.file_number += 1
            self.counter = 0

        Novel = item['Novel']
        Novel_style = item['Novel_style']
        Author = item['Author']
        Publication_date = item['Publication_date']
        Score = item['Score']
        Read_times = item['Read_times']
        New_page = item['New_page']
        Introduction = item['Introduction']

        print(Novel)
        print(Novel_style)
        print(Author)
        print(Publication_date)
        print(Score)
        print(Read_times)
        print(New_page)
        print(Introduction)

        with open(f'第{self.file_number+1}页.txt', 'a', encoding='utf-8') as f:
            # file_path = item['Novel']+'.txt'
            # f = open(file_path, "wb")
            f.write('小说名: ' + str(Novel) + '\n')
            f.write('小说类型: ' + str(Novel_style) + '\n')
            f.write('作者: ' + str(Author) + '\n')
            f.write('出版时间: ' + str(Publication_date) + '\n')
            f.write('评分: ' + str(Score) + '分' + '\n')
            f.write('总阅读人数: ' + str(Read_times) + '\n')
            f.write('最新章节: ' + str(New_page) + '\n')
            f.write('简介: ' + str(Introduction) + '\n')
            f.write('\n')

        return item

    def close_file(self):
        if hasattr(self, "file") and not self.file_number:
            self.file.close()
```

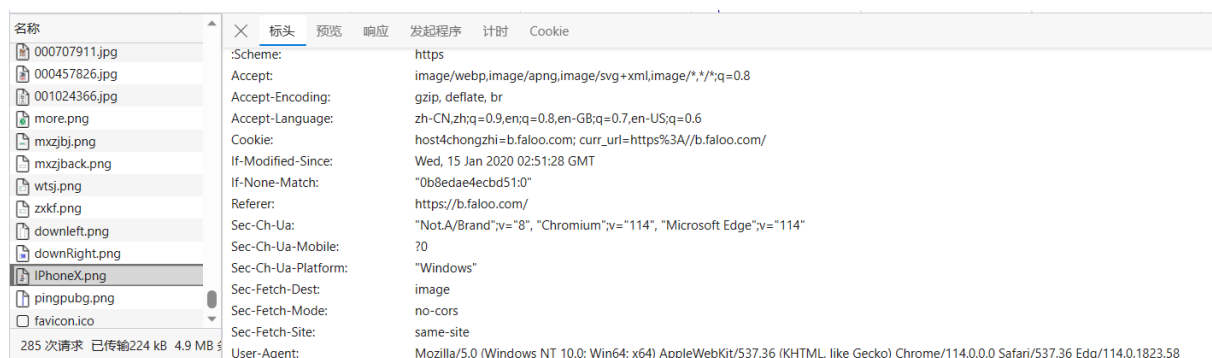

(11) **settings.py**:配置文件, 将 **robots** 协议设置为 **True**, 即遵守网页的 **robots** 协议。

```
# Obey robots.txt rules
ROBOTSTXT_OBEY = True
LOG_LEVEL = 'ERROR'
```

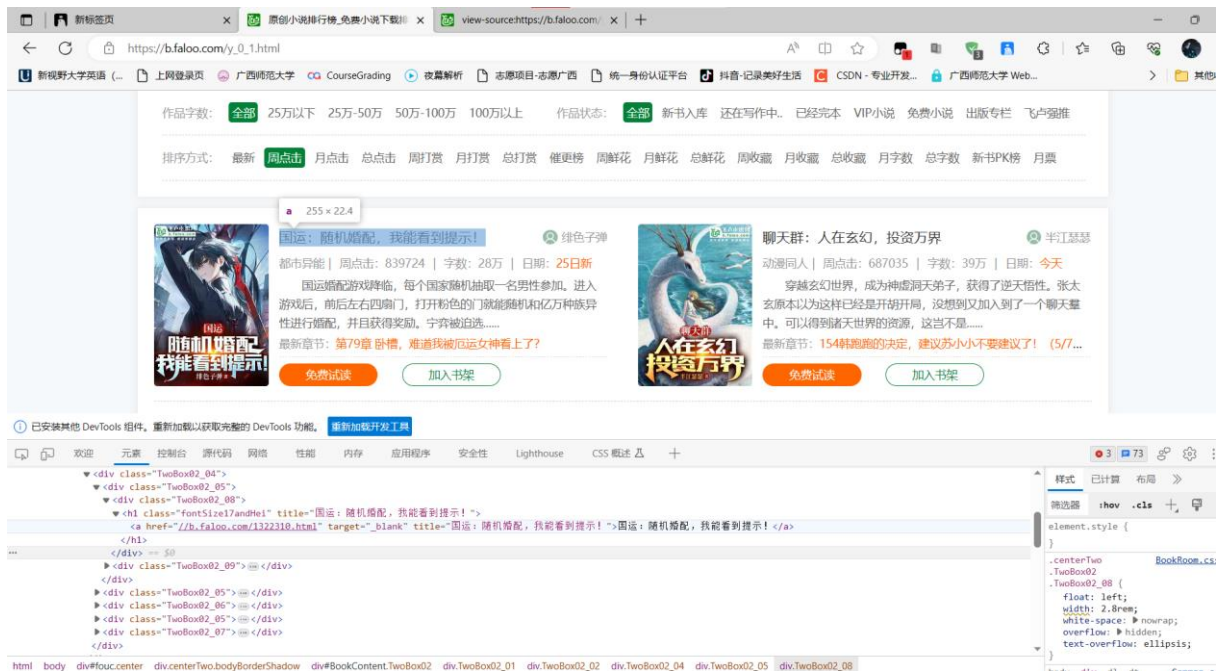
(12) 豆瓣网站的 **robots** 协议



我们需要根据 **robots** 协议的限制来设置我们的爬虫代理, 设置爬虫代理如下:



(13) 观察网页的源代码以及提取爬取相关的路径信息



以下是飞卢小说书库第一页的电影网页源代码，从中我们可以观察出每一页的小说超链接都存在于网页源代码

```
<div class="TwoBox02_03"><a href="//b.faloo.com/1322310.html" target="_blank" title="国运：随机婚配，我能看到提示!"></a></div>
```

(14) 实现爬虫翻页

实现翻页有多种方法 1) 利用网址拼接技术递归爬取下一页。2) 对于有下一页标签的网页，可以利用 Xpath 技术提取下一页的网址，再利用拼接技术即可实现翻页，由每一部小说的超链接也不是完整的，所以在请求每一部小说的 url 时要拼接完整。在本实验当中，我使用的是第 2 种方法。

第 16 页共 48 页

```
<a href="//b.faloo.com/1322310.html" target="_blank" title="国运：随机婚配，我能看到提示！">国运：随机婚配，我能看到提示
</a>
</div>
<div class="TwoBox02_09">
</div>
<div class="TwoBox02_05">
<div class="TwoBox02_06">
<div class="TwoBox02_05">
<div class="TwoBox02_07">
</div>
</div>
<div class="TwoBox02_02">
<div class="TwoBox02_03">
<a href="//b.faloo.com/1324481.html" target="_blank" title="聊天群：人在玄幻，投资万界">
</a>
</div>
<div class="TwoBox02_04">
</div>
</div>
<div class="TwoBox02_01">
<div class="TwoBox02_02">
<div class="TwoBox02_02">
</div>
</div>
<div class="TwoBox02_01">
<div class="TwoBox02_02">
<div class="TwoBox02_03">
<a href="//b.faloo.com/1322032.html" target="_blank" title="我祁同伟，刑案之王！">
</a>
</div>
<div class="TwoBox02_04">
</div>
</div>
```

3.6 关键技术方法的使用

1) 第 i 页的小说网址的拼接

```
finish_url = []
scrawl_urls = []
for i in range(1,2):
    scrawl_url = []
    start_urls = "https://b.faloo.com/y_0_"+str(i)+".html" #从哪个页面开始爬取
    # print(start_urls)
```

2) 把小说超链接的 url 拼接完整

```
content = html5lib.parse(html_txt, treebuilder="lxml", namespaceHTMLElements=False)
# print(content)
new_urls = content.xpath('//div[@class = "TwoBox02_03"]/a[@href and @target = "_blank" and @title]/@href')
new_urls = set(new_urls)
# print(new_urls)
scrawl_url.extend(new_urls)
# print(scrawl_url)
for url in scrawl_url:
    url = 'https:' + url
    scrawl_urls.append(url)
# print(scrawl_urls)
```

3.7 实验运行结果

运行结果截图，共有 30x100 条记录：

```
运行: start
第二题：“乃是人而不可以不知鸟乎？诗云，穆穆文王，请写一篇简论。”
.....
看到这些题目，考生们直接笑了！
奇怪，这都是什么考题啊？这是人能答上来的题目？
科举结束后，考生们一个个笑着离开了贡院。
接着，万千学子求见朱元璋。
“陛下，求求您千万不要让叶轩出题了，这答案谁能想得到啊？”
朱元璋则大喜，“叶发卿真是我大明国之栋梁啊！”
热思地下甜甜果实，为我疯狂打榜
都市言情
甜宠文
2023-05-03
10
2619745
《热思地下甜甜果实，为我疯狂打榜》第一百四十六章 圣刀时空的奇幻旅程！新魔刀冰轮九！

苏尘穿越到了诸天降临、逐渐离谱的世界。
并且绑定诸天科普系统。
这个世界，有甜宠果实，新魔刀、武功秘籍.....
但这一切，只有苏尘知道它们的来历！
于是，苏尘索性当起了科普主播。
“水友们，这叫甜甜果实，食用者可以冻结一切！”
“没错，这叫新魔刀，看我的，流刀若火，花解！”
“降龙十八掌？打出千米龙影有没有见过？”
PDD，我是古代种豌豆形态能力者，谁敢与我一战？
噫巴新德，家人们谁懂啊？我居然变成蛋蛋骑士了！
臭小妹，不会有人还不是超凡者吧？不会吧？
看着大夏遍地的超凡者，苏尘表示，我只是一名搞了那么亿点点的科普主播！

进程已结束，退出代码0
```

将数据保存为 txt 将文件，文件中呈现的内容是将每一部小说的元素一一显示，爬取数据分别为：小说名、小说类型、出版时间、评分、总阅读人数、最新章节的标题以及小说简介。本次实验成功的爬取预期目标的数据，并成功保存在 txt 文件中。

```
demo.py × 第1页.txt × items.py × start.py × pipelines.py × settings.py ×
1 小说名：开局约会杨老板，她转我百万巨款
2 小说类型：都市言情作者：很高兴
3 出版时间：2022-10-01
4 评分：10分
5 总阅读人数：3871564
6 最新章节：《开局约会杨老板，她转我百万巨款》992 学习哄蜜宝绝招
7 简介：
8 林风重生，家里还没破产，还没
9 他还激活了超凡投资系统，
10 下定决心改变人生
11 当他打开股票交易界面，准备开启富可敌国的一生时.....
12 老妈：“就知道玩电脑，给我去相亲！”
13 林风不情不愿地赴约，却惊讶发现一
14 包间内端坐的，竟然是美艳大明星杨蜜！
15 林风默默反锁了屋门。
16 杨蜜惊恐地捂住胸口，无力看着男人一点一点凑近自己.....
17 只听林风缓缓开口：
18 “一起搞钱吗？”
19
20 小说名：融合白银大超，打造万界仙秦帝国
21 小说类型：武侠修真作者：四合院李云龙
22 出版时间：2022-08-20
23 评分：10分
24 总阅读人数：3934368
25 最新章节：《融合白银大超，打造万界仙秦帝国》第一千一百九十四章 终生跟随秦王
26 简介：
27 嬴政穿越大秦，成为了嬴政的第七子，因为母亲的原因，所以非常得嬴政的喜爱，但他就是一条咸鱼，每天只是晒太阳，啥也不干.....
28 嬴政恨铁不成钢，但却不知道，嬴政有着10%的氪星人血统.....只要晒太阳就能变强！
29 而嬴政实在受不了了，让嬴政负责诸子百家之事！
30 【捉回天明，盖聂，奖励2%氪星人血统！】
31 而只要完成系统发布的任务，嬴政就可以获得血统奖励，当血统奖励到极致，白银大超，毁灭日超人？亦或是.....祖国人！
32 成立锦衣卫，一人踏平机关城，威压小圣贤庄.....
33 而这仅仅只是开始，从秦时开始，打造万界仙秦帝国，让里龙旗飘扬在诸天万界！

1:1 CRLF UTF-8 4
```

四、总结

通过这次对飞卢小说书库小说基本信息的爬取，我对 Scrapy 框架有了更深的理解，以及对爬虫步骤也有了进一步的熟悉，同时能够熟练地运用 xpath 选择器对所需要提取的内容进行爬取，更加熟练地掌握了对拼接技术以及对提取内容的保存，还学会了如何观察网页源代码，并找到所要提取内容所在的标签，还学会了如何用网页检查工具定位所要爬取内容的具体位置，通过这一次实验，更加激发了我对爬虫的兴趣，在爬虫的过程中我也学会了要抓取一个网站的数据时首先要观察该网站的 robots 协议，了解哪些数据可以抓取，哪些不可以抓取，在以后的爬虫实验中，我也会遵守这个规则，更加努力地学习新的爬虫知识，这次实验对于我来说仅仅是爬虫的开始，但也让我收获甚多，受益匪浅，使得我更有信心去学习这门技术。