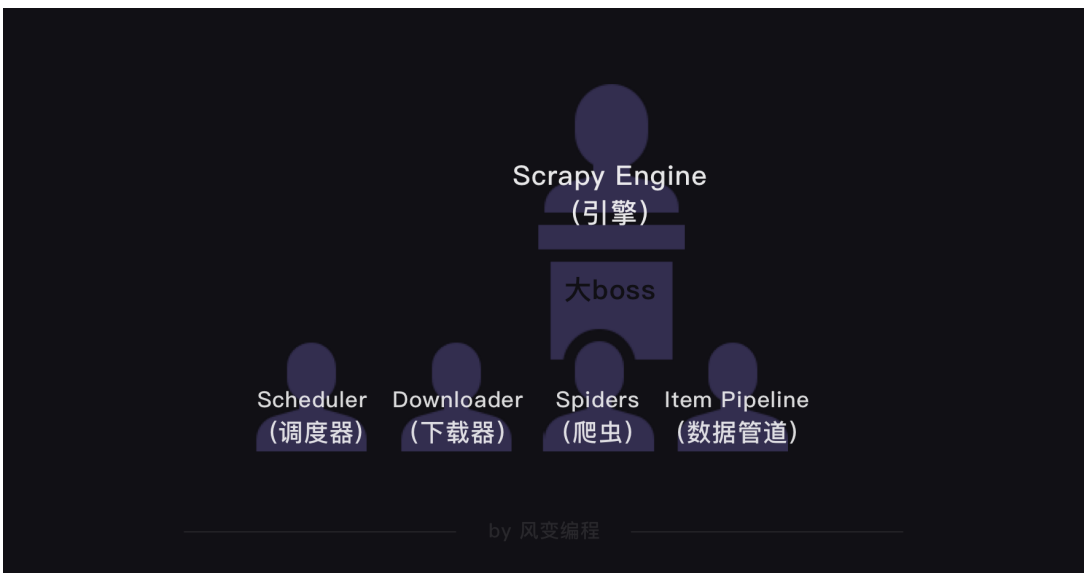
第 13 关、Scrapy 框架

1、Scrapy

Scrapy 框架可自动实现麻烦的异步，导入和操作不同的模块，比如requests模块、gevent库、csv模块等。

1-1、Scrapy 的结构



1-1-1、Scrapy Engine（引擎）

收集其下四个模块的反馈数据，并对其下达操作指令，是框架中的核心点。

1-1-2、Scheduler（调度器）

主要负责处理引擎发送过来的 requests 对象（即网页请求的相关信息集合，包括 params，data，cookies，request headers… 等），会把请求的 url 以有序的方式排列成队，并等待引擎来提取（功能上类似于 gevent 库的 queue 模块）。

1-1-3、Downloader（下载器）

负责处理引擎发送过来的 requests，进行网页爬取，并将返回的 response（爬取到的内容）交给引擎。它对应的是爬虫流程【获取数据】这一步。

1-1-4、Spiders（爬虫）

主要任务是创建 requests 对象和接受引擎发送过来的 response（Downloader 部门爬取到的内容），从中解析并提取出有用的数据。它对应的是爬虫流程【解析数据】和【提取数据】这两步。

1-1-5、Item Pipeline（数据管道）

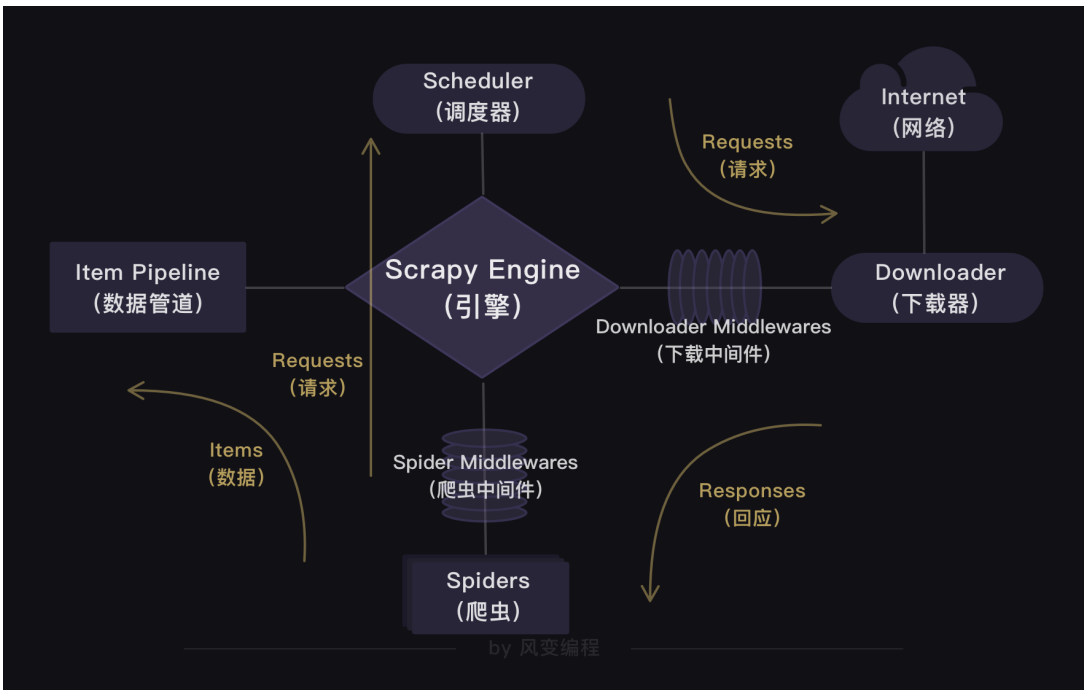
只负责存储和处理 Spiders 部门提取到的有用数据。这个对应的是爬虫流程【存储数据】这一步。

1-1-6、Downloader Middlewares（下载中间件）

会提前对引擎发送的诸多 requests 做出处理。

1-1-7、Spider Middlewares（爬虫中间件）

提前接收并处理引擎发送来的 response，过滤掉一些重复无用的东西。



1-2、Scrapy 的工作原理

- Scrapy 框架的工作原理——引擎是中心，其他组成部分由引擎调度。
- Scrapy 中的程序全部都是异步模式，所有的请求或返回的响应都由引擎自动分配去处理。哪怕有某个请求出现异常，程序也会做异常处理，跳过报错的请求，继续往下运行程序。

2、Scrapy 的用法

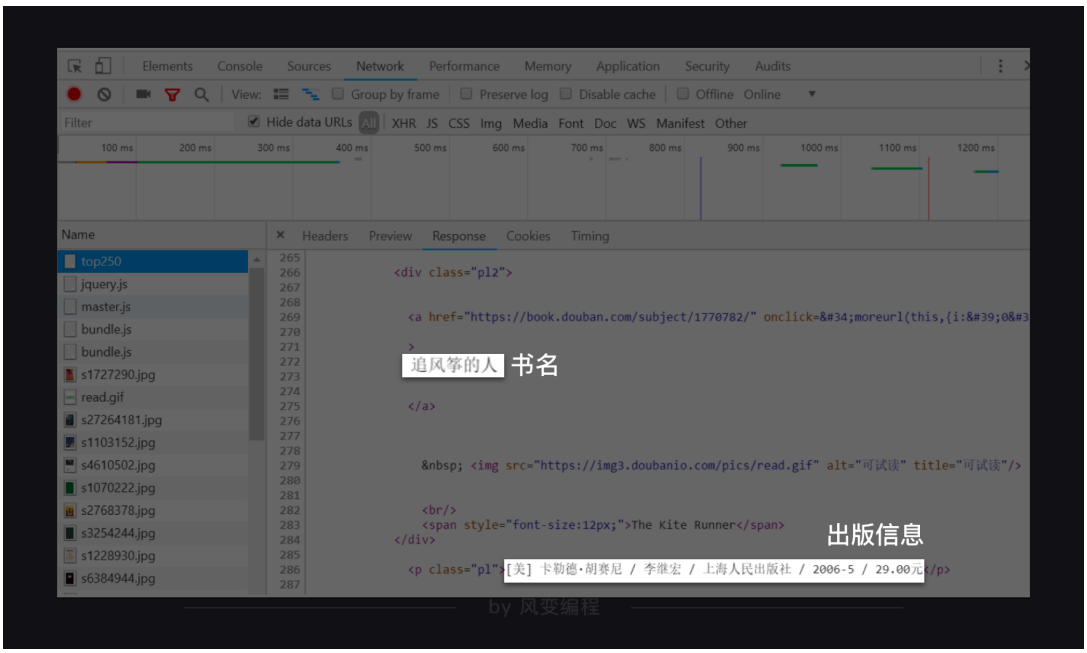
2-1、明确目标

（1）目标网站：https://book.douban.com/top250；
（2）项目目标：豆瓣 Top250 图书爬取前三页书籍的信息，也就是爬取前 75 本书籍的信息（包含书名、出版信息和书籍评分）。

2-2、过程分析

（1）确定数据所在页面

右击打开“检查”工具，点开 Network，刷新页面，然后点击第 0 个请求 top250，看 Response，在里面翻找到书名、出版信息，说明我们想要的书籍信息就藏在这个网址的 HTML 里。



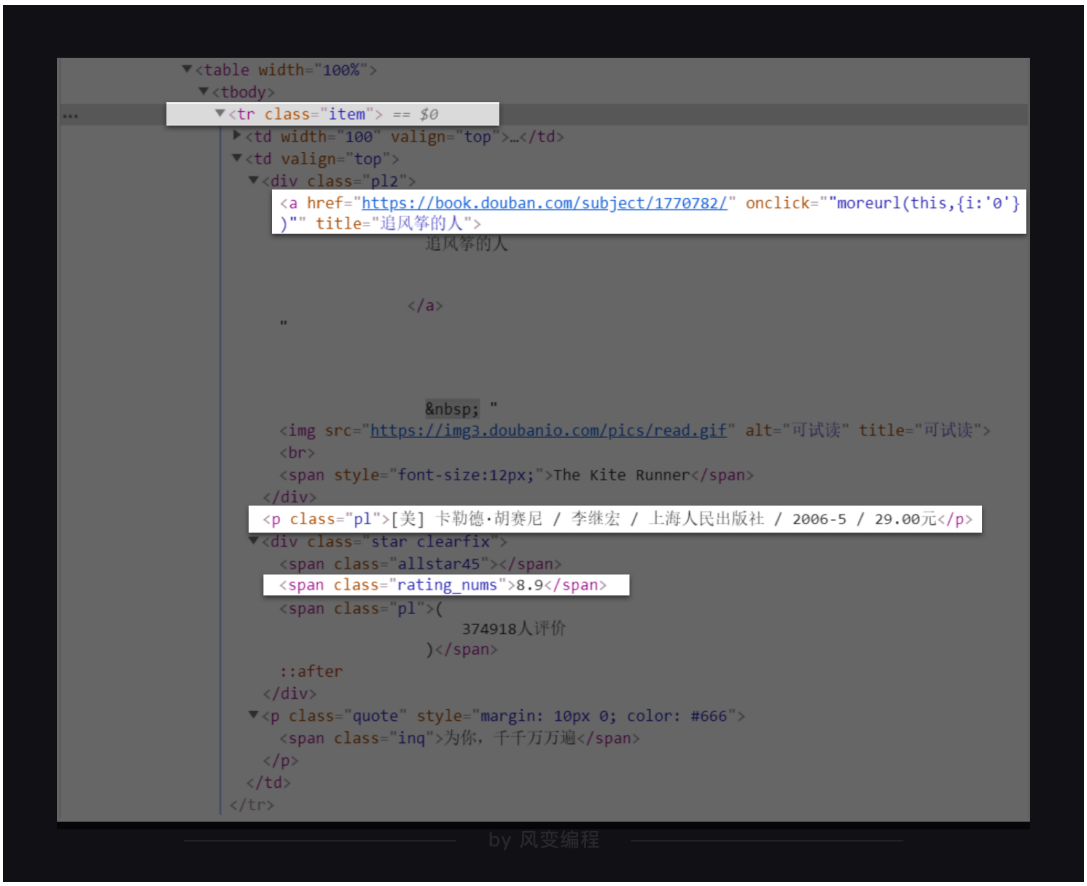
（2）确定数据所在位置

①、数据规律

每翻一页，网址后面的数字都会增加 25，说明这个 start 的参数就是代表每页的 25 本书籍，只要改变 ?start = 后面的数字（翻一页加25），我们就能得到每一页的网址。

②、数据位置

书名、出版信息和评分的数据分别在 <tr class=”item”> 元素下 <a> 元素的 title 属性的值、<p class=”pl”> 元素、 元素。



2-3、代码实现

2-3-1、创建项目

（1）安装 Scrapy 模块

```
Windows：在终端输入命令：pip install scrapy；
mac：在终端输入命令：pip3 install scrapy
```

（2）创建 Scrapy 项目

- 在本地电脑打开终端（windows：Win+R，输入cmd；mac：command+空格，搜索“终端”），然后跳转到你想要保存项目的目录下。比如：我们要在电脑的【下载】文件夹下保存，需要我们在 cmd 里输入 cd C:\Users\user\Downloads ；
- 再输入一行能帮我们创建 Scrapy 项目的命令：scrapy startproject douban，douban 就是 Scrapy 项目的名字。按下 enter 键，一个 Scrapy 项目就创建成功了。

（3）Scrapy 项目结构

settings.py 是 scrapy 里的各种设置。items.py 是用来定义数据的，pipelines.py 是用来处理数据的，它们对应的就是 Scrapy 的结构中的 Item Pipeline（数据管道）。

2-3-2、编辑爬虫

spiders 是放置爬虫的目录。我们可以在 spiders 这个文件夹里创建爬虫文件 top250，后面的大部分代码都需要在这个 top250.py 文件里编写。

（1）导入模块

```
import scrapy
import bs4
```

- 导入 BeautifulSoup 用于解析和提取数据；
- 导入 scrapy 是待会我们要用创建类的方式写这个爬虫，我们所创建的类将直接继承 scrapy 中的 scrapy.Spider 类。

（2）核心代码

```
class DoubanSpider(scrapy.Spider):
    name = 'douban'
    allowed_domains = ['book.douban.com']
```

```
    start_urls = ['https://book.douban.com/top250?start=0']

    def parse(self, response):
        print(response.text)
```

- 第 1 行代码：定义一个爬虫类 DoubanSpider（ DoubanSpider 类继承自 scrapy.Spider 类）；
- 第 2 行代码：name 是定义爬虫的名字，这个名字是爬虫的唯一标识。name = 'douban' 意思是定义爬虫的名字为 douban（我们启动爬虫的时候，要用到这个名字）；
- 第 3 行代码：allowed_domains 是定义允许爬虫爬取的网址域名（不需要加 https:// ）。如果网址的域名不在这个列表里，就会被过滤掉；
- 第 4 行代码：start_urls 是定义起始网址，就是爬虫从哪个网址开始抓取；
- 第 6 行代码：parse 是 Scrapy 里默认处理 response 的一个方法，中文是解析。

（3）根据数据的规律，我们的完善代码

```
class DoubanSpider(scrapy.Spider):
    name = 'douban'
    allowed_domains = ['book.douban.com']
    start_urls = []
    for x in range(3):
        url = 'https://book.douban.com/top250?start=' + str(x * 25)
        start_urls.append(url)
```

2-3-3、定义数据

（1）item.py 文件

Item Pipeline（数据管道）处理定义数据：

```
import scrapy
#导入scrapy
class DoubanItem(scrapy.Item):
#定义一个类DoubanItem，它继承自scrapy.Item
    title = scrapy.Field()
    #定义书名的数据属性
    publish = scrapy.Field()
    #定义出版信息的数据属性
    score = scrapy.Field()
    #定义评分的数据属性
```

- 第 1 行代码，导入 scrapy 的目的是让创建的类将直接继承 scrapy 中的 scrapy.Item 类；
- 第 3 行代码定义 DoubanItem 类继承自 scrapy.Item 类；
- 第 5、7、9 行代码：定义了书名、出版信息和评分三种数据。scrapy.Field() 实现数据能以类似字典的形式记录。

（2）top250.py 文件

```
import scrapy
import bs4
from ..items import DoubanItem
# 需要引用DoubanItem，它在items里面。因为是items在top250.py的上一级目录，所以要用..items，这是一个固定用法。

```

```
class DoubanSpider(scrapy.Spider):
#定义一个爬虫类DoubanSpider。
    name = 'douban'
    #定义爬虫的名字为douban。
    allowed_domains = ['book.douban.com']
    #定义爬虫爬取网址的域名。
    start_urls = []
    #定义起始网址。
    for x in range(3):
        url = 'https://book.douban.com/top250?start=' + str(x * 25)
        start_urls.append(url)
        #把豆瓣Top250图书的前3页网址添加进start_urls。

    def parse(self, response):
    #parse是默认处理response的方法。
        bs = bs4.BeautifulSoup(response.text,'html.parser')
        #用BeautifulSoup解析response。
        datas = bs.find_all('tr',class_="item")
        #用find_all提取<tr class="item">元素，这个元素里含有书籍信息。
        for data in  datas:
        #遍历data。
            item = DoubanItem()
            #实例化DoubanItem这个类。
            item['title'] = data.find_all('a')[1]['title']
            #提取出书名，并把这个数据放回DoubanItem类的title属性里。
            item['publish'] = data.find('p',class_='pl').text
            #提取出出版信息，并把这个数据放回DoubanItem类的publish里。
            item['score'] = data.find('span',class_='rating_nums').text
            #提取出评分，并把这个数据放回DoubanItem类的score属性里。
            print(item['title'])
            #打印书名。
            yield item
            #yield item是把获得的item传递给引擎。
```

- 第 3 行，需要引用 DoubanItem，它在 items 里面。因为是 items 在 top250.py 的上一级目录，所以要用 ..items ，这是一个固定用法；
- 第 37 行的 yield 语句类似 return，不过区别 return 的点在于不会结束函数，且能多次返回信息。

2-3-4、设置

点击 settings.py 文件修改默认设置代码：

```
# Crawl responsibly by identifying yourself (and your website) on the
user-agent
USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36'

# Obey robots.txt rules
ROBOTSTXT_OBEY = False
```

- 第 2 行把 USER _AGENT 的注释取消（删除 # ），然后替换掉 user-agent 的内容，就是修改了请求头；

- 第 5 行把 ROBOTSTXT_OBEY = True 改成 ROBOTSTXT_OBEY = False，就是把遵守 robots 协议换成无需遵从 robots 协议，这样 Scrapy 就能不受限制地运行。

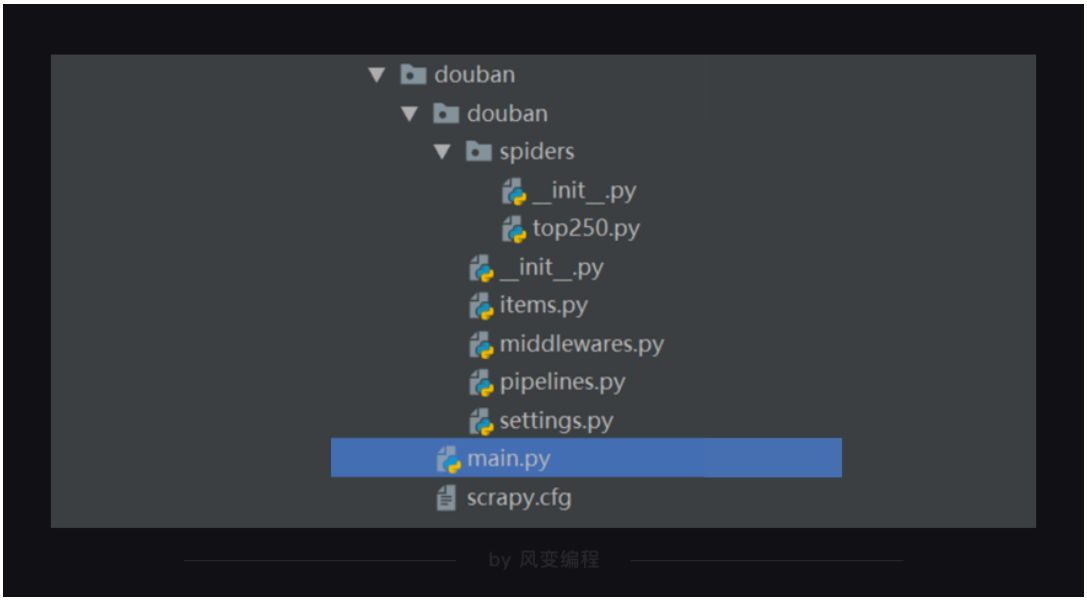
2-3-5、运行

（1）方法一：本地终端运行项目

在本地电脑的终端跳转到 scrapy 项目的文件夹（跳转方法：cd + 文件夹的路径名），然后输入命令行：scrapy crawl douban（douban 就是我们爬虫的名字）。

（2）方法二：创建 main 文件运行

→ 在最外层的大文件夹里新建一个 main.py 文件（与 scrapy.cfg 同级）；



→ main.py 文件里，输入以下代码，点击运行，Scrapy 的程序就会启动：

```
1 from scrapy import cmdline
2 #导入cmdline模块,可以实现控制终端命令行。
3 cmdline.execute(['scrapy','crawl','douban'])
4 #用execute（）方法，输入运行scrapy的命令。
```

- 第 1 行代码：在 Scrapy 中有一个可以控制终端命令的模块 cmdline 。导入了这个模块，我们就能操控终端；
- 第 3 行代码：在 cmdline 模块中，有一个 execute 方法能执行终端的命令行，不过这个方法需要传入列表的参数。我们想输入运行 Scrapy 的代码 scrapy crawl douban，就需要写成 ['scrapy','crawl','douban'] 这样。