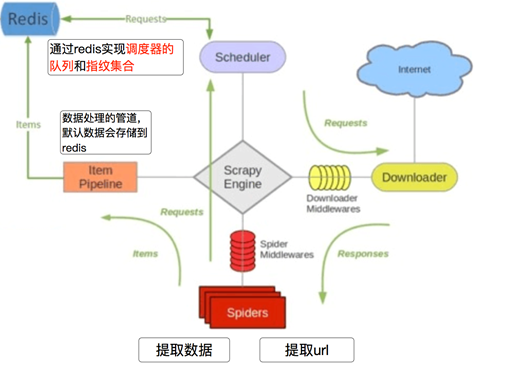
# 京东图书数据爬取(LOG)

## 爬取框架（scrapy\_redis）

**使用scrapy\_redis框架的作用是什么？**

1. 它能够实现request的去重，也就是每个url只爬取一次,这样的话就不会导致爬取的数据有重复。
2. 它能够实现持久化，何为持久化，就是说，你爬取一个网站爬了一段时间，还没有爬取完，你想让电脑休息下，停止了爬取这个网站，如果你用的是scrapy\_redis，那么当你下一次爬取数据的时候，会在前一次爬取的状态之后，继续爬取数据，也就是不会导致爬取的数据重新爬取。
3. 很容易实现分布式爬虫，提高爬虫的效率。通过celery也可以实现异步分布式爬虫，但是有点麻烦。而使用scrapy\_redis，很轻松的实现分布式爬虫
4. 爬取的数据可以存在redis中或者存在其他的数据库中（mongodb,mysql…）

**Scrapy\_redis的原理结构图：**



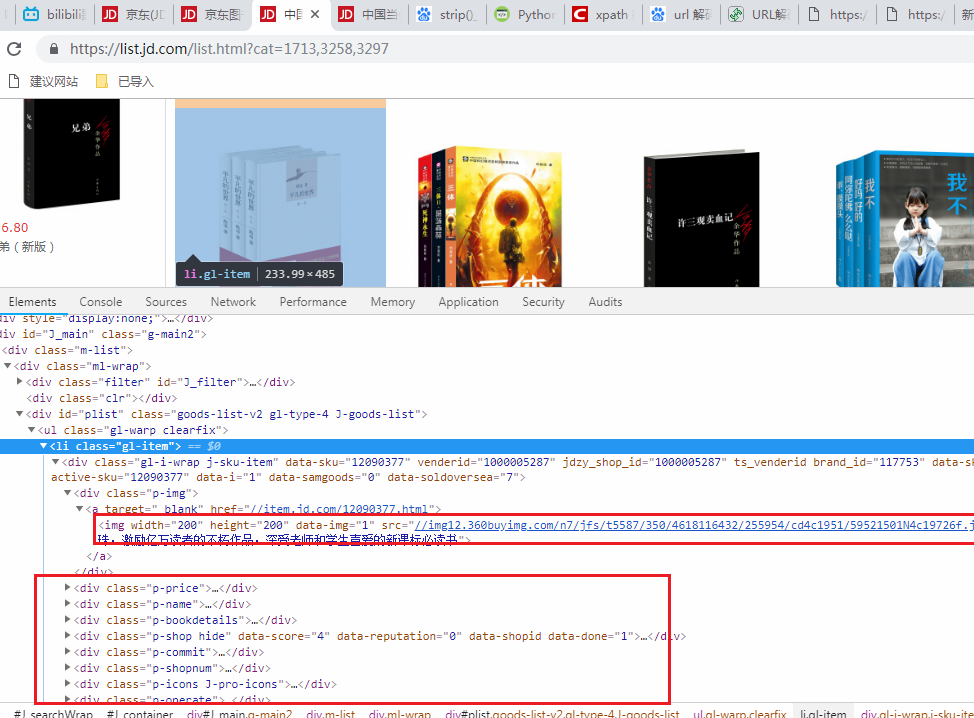
具体的细节就详说了，可以参考CSDN或者其他的技术网站。

## 爬虫思路

1.通过分析京东图书这个页面，来分析出大分类，和小分类的数据爬取模型:



2.通过分析小分类中的页面，来分析出每本书的详细内容（包括书名、作者、出版社、出版日期、价格），并的出爬取数据的思路,注意一定要弄清楚是份响应中和Element中的内容都对应，不能排除有些数据是通过ajax或者js等构成的（比如价格就是通过ajax获取的）:

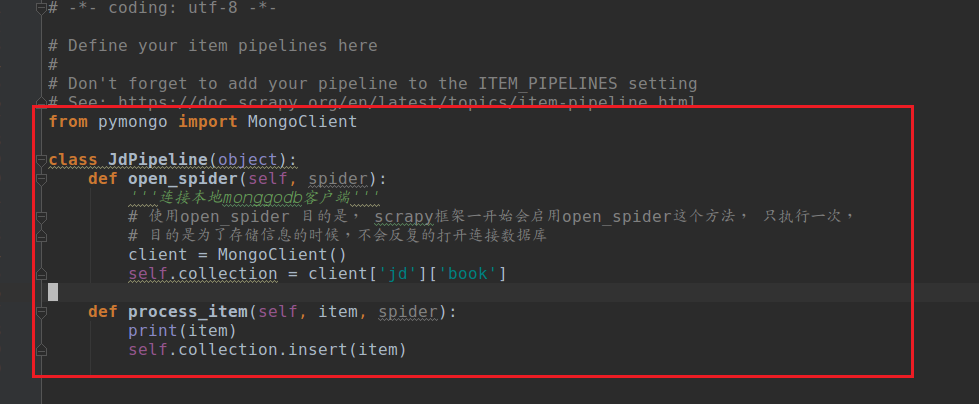


3.获取下一页链接，并获得下一页页面的数据，京东的下一页比较容易，直接是通过响应中获取，（而有些下一页是隐藏的通过js实现页面的跳转，并且还要带有参数，因此这类的url分析是有点难度的）：



## 获取数据的存储

获取到的数据可以默认的存储在redis中，以列表的格式。也可以把数据存储在mongodb中，这里我是放在mongodb中的，缓解redis压力。

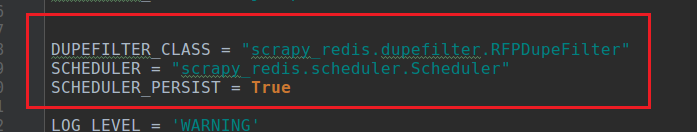


获取的数据的格式为：



## 陌生知识点：

1. scrapy\_redis配置：



1. 爬取数据的时候，获取作者或者其它的数据可能会遇到首尾有空格，处理方法使用strip()方法
2. 在获取价格这个页面的时候域名与jd.com不符，因此在allowed-domain中添加价格页面的域名。
3. url补充完整。获取到的url地址一般都是不完整的缺少方案或者其它等，可以导入urllib 中的parse，调用parse中的urljoin()方法，自动补充完整：

eg:

parse.urljoin(response.url, next\_url)