## 爬取的流程图

item

item

item

item

parse

parse\_booklist

parse\_book\_info

parse\_booklist

scrapy引擎

Pipline

Mongodb

item

item

## 爬取苏宁易购日志

### 第一部分

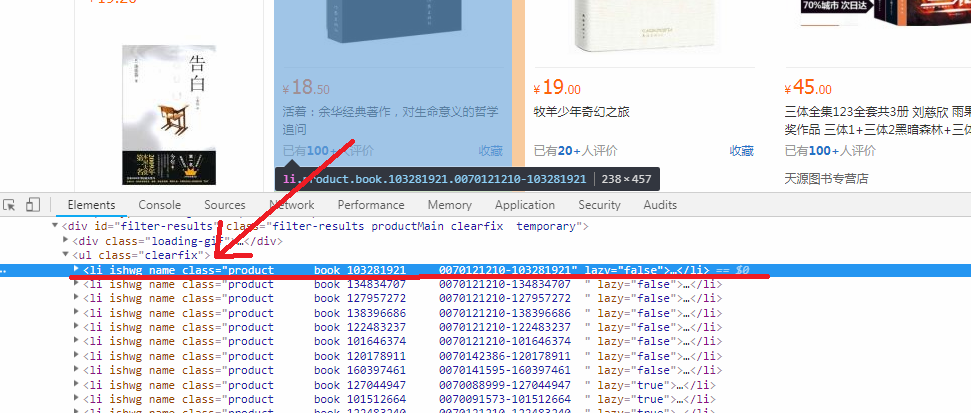
访问suning.com，选择全部分类中的图书这个大分类，进入之后就是book.suning.com，鼠标移到左侧的menu中，会弹出子弹框，这里要抓取的是橙色文字和红色文字，以及红色文字的href,也就是大分类是文学艺术，小分类是小说，并且抓取小分类的href。



### 第二部分

访问第一部分中抓取的href，进入 <https://list.suning.com/1-264003-0.html> 这个页面，其中264003是这个小分类的编号，这个编号也要抓取，目的是为了下一页的跳转而做准备的。

在这个网页中，抓取的是每个ul元素下的li中的a标签中的href，以及img的href,如下图所示，但是注意，这里一定要对比一下Elements和Network中的response进行对比，防止有些内容是通过js等生成的或者通过ajax动态获取，这里一定要注意，后面点抓取价格也是通过ajax动态获取一个json。



### 第三部分

访问第二部分抓取到的url，进入到每本书的详细页面，这里要抓取的是标题，作者，内容简介。如

<https://product.suning.com/0070121210/103281921.html?safp=d488778a_46602_resultsRblock_10>

示例如图所示，当然，也可以爬取出版社，出本日期，累计评价的数据等。这些都可以增加。

注意，在这个网页中爬取不到价格这个数据，这也是爬取苏宁易购的**一个难点，**我找了好久，终于在一个推荐的页面中发现它的价格数据。



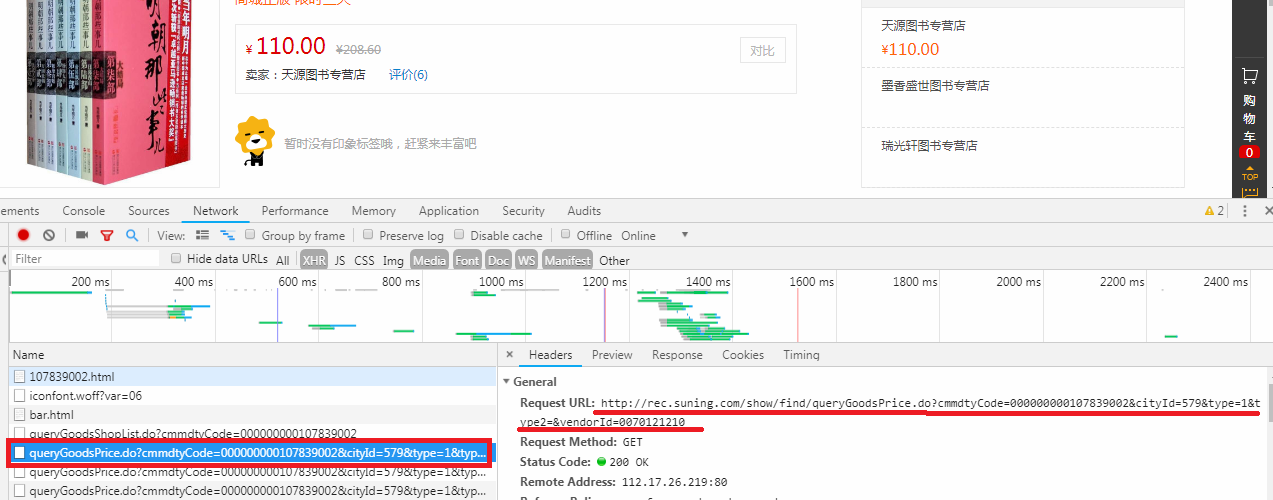


### 第四部分

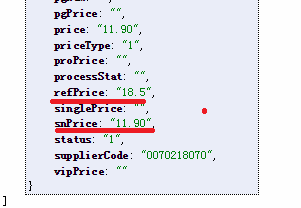
爬取书本的价格，这是整个项目中的**难点之一，在爬取数据的时候，价格并不在第三部分中的url地址中，并且通过Chrome中的Network进行抓包，也没有发现价格的踪影。这单很让人诡异。**

不过研究了老半天苏宁的这个网站，终于发现它的价格藏在哪个地方了。

它居然藏在一个反馈的页面中。诡异的离谱了。如图所示中的红线部分



**当你访问红线的url地址时，就会发现它得到的响应是一个数组中嵌套着json,这个明显是ajax动态请求的。如图所示**

****

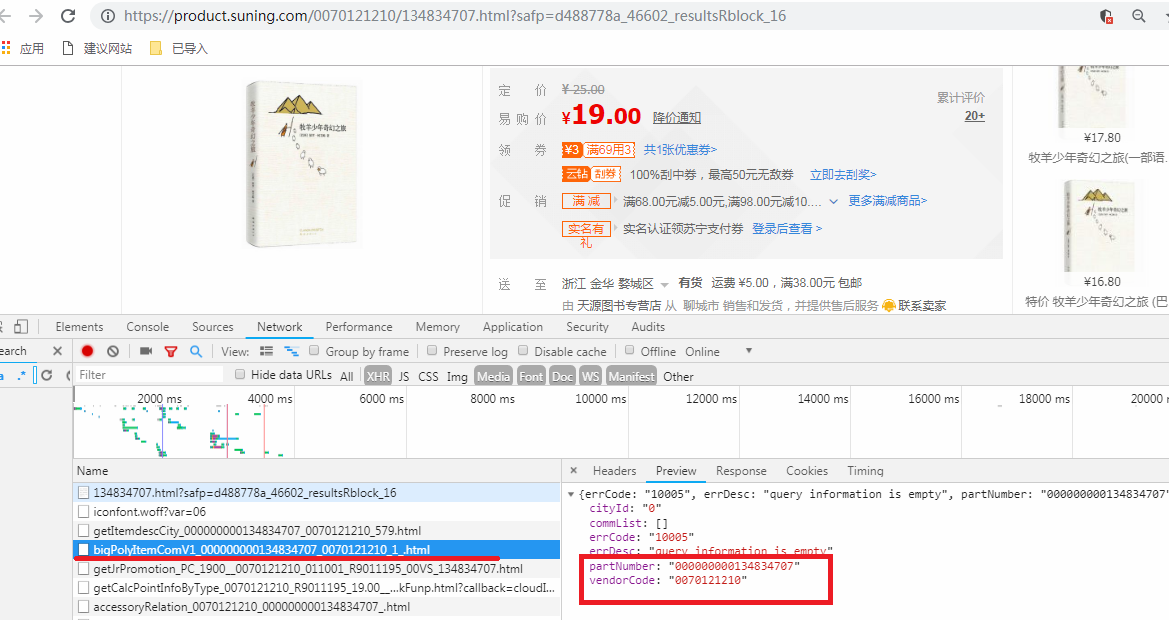
**既然知道它的价格藏身之处，那就简单多了，只要研究下它的ajax请求的url地址即可：**

**‘http://rec.suning.com/show/find/queryGoodsPrice.do?cmmdtyCode=000000010569087665&cityId=025&type=1&type2=&vendorId=0070218070’，研究发现只需要获取两个数据即可，那么问题来了这两个数据从何而来。**

### 第五步

只要你查看访问每个url中的响应，并且打开开发工具中的Network进行抓包，仔细分析就会得出结果的。

**这里就可以通过第三个步骤中的得到我们想要的获取价格中所必须的两个url链接中的参数，打开Chrome的Network进行抓包即可。如图所示：**

****

**这样就完成了抓取整个数据的流程。**

**当然除此之外，还有一个难点就是第二步中的下一页。**

### 第六步

访问第二步中的下一页。这个说简单也简单，说难也有点难。

**在第二步中得到了小分类的编号，这里就要利用这个编号了，那么还需要一个下一页的url地址，在响应中的并没有下一页的url地址，那么它会在哪儿呢？一般数据只会出现在两个地方，一个是访问地址的响应中，另一个地方就是js,或者动态请求。**

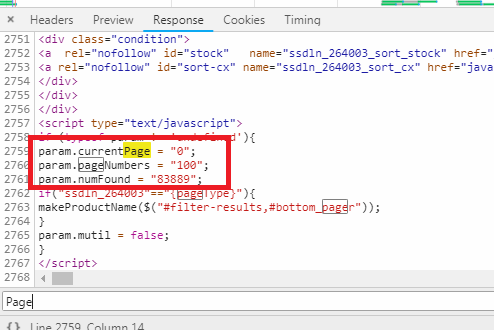
只要细心很耐心，会发现它在动态请求那部分中，也就是ajax中。

这也是一种方法。

**当然还有一个方法**，你鼠标移入第二页的地方，浏览器会显示这个url如图所示：

****

注意看右下角，发现了什么，哈哈，这就是细节，我尝试了以下它需要两个元素，一个是小分类的编号，另一个是当前页码，那么这个当前页码在哪儿，很显然不是在响应中就是在动态请求中，会发现它居然在响应中。搜索响应Page如图所示



**是不是出来了，这就好办了，只要构造下一页的url地址即可。其他的交给scrapy框架的引擎来处理。**