设计说明书

本项目主要阐述如何使用 Scrapy 爬虫框架爬取淘宝、京东等电商网站。主要抓取京东、淘宝等网站上各商品种类销量前 20 名商品信息，包括商品名称、商品 URL、商品类型（三层分类）。

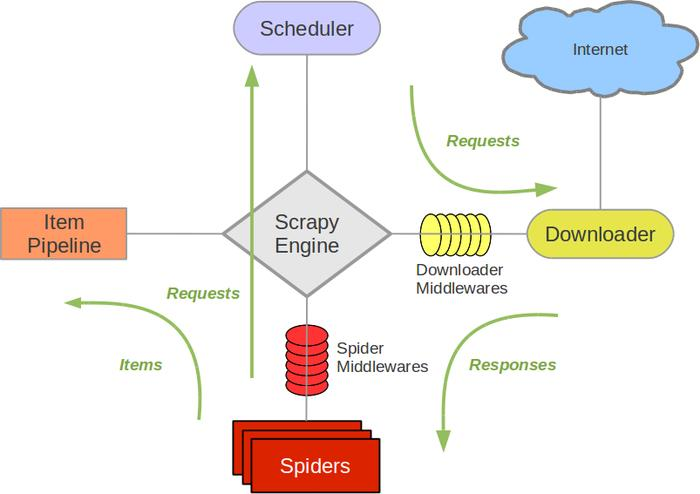
# 爬虫设计

## 设计思路

淘宝，京东等许多页面都用到了 AJAX，即由 JS 来异步加载数据，所以普通爬虫不能直接爬取，基于这个原因，除了 Scrapy 框架之外，我们还选择了 Splash 库。Splash 是一个轻量级、脚本化的服务器，并且提供了 HTTP API。使用 Scrapy-Splash，我们能够轻松的将 Scrapy 和 Splash 集成在一起，使得 Scrapy 能够爬取动态解析的页面。

## Scrapy 框架简要介绍

Scrapy爬虫框架的基本架构如图：



Scrapy 的数据流如下：

1. 每个爬虫需定义一个 start\_urls，Engine 开启爬虫后，从该爬虫的 start\_urls

获取初始 URLs 来爬取。

1. Engine 从 Spider 获取 URLs 后，就将其作为 Requests 放入 Scheduler 中调度。
2. 然后 Engine 向 Scheduler 索取下一部分的 URLs 来爬取。
3. Scheduler 就返回下一部分要爬取的 URLs 给 Engine，然后 Engine 就将其发送至

Downloader，期间通过 Downloader Middlewares（请求方向）。

1. 当页面下载完成，Downloader 就会生成该页面的 Response，并送至 Engine，期

间通过 Downloader Middlewares（回应方向）。

1. Engine 收到 Response 就会将其送至 Spider 处理，期间通过 Spider

Middleware(输入方向)。

1. Spider 会处理 Response，然后返回分析获得的 Items 和新的 Request 给

Engine。

1. Engine 就会将 Items 送至 Item Pipeline，将 Request 送至 Scheduler。
2. 处理过程就又会从(c)重复执行，直到没有新的 Request 送至 Scheduler，最后

Engine 就会关闭该 domain。

# 项目实现与分析

## 项目实现

项目整体如下：

.

|-- crawler

| |-- \_\_init\_\_.py

| |-- items.py

| |-- jd.sh

| |-- mongo.py

| |-- pipelines.py

| |-- settings.py

| |-- spiders

| | |-- \_\_init\_\_.py

| | |-- jd.py

| | |-- taobao.py

| | `-- utils.py

| `-- tb.sh

`-- scrapy.cfg

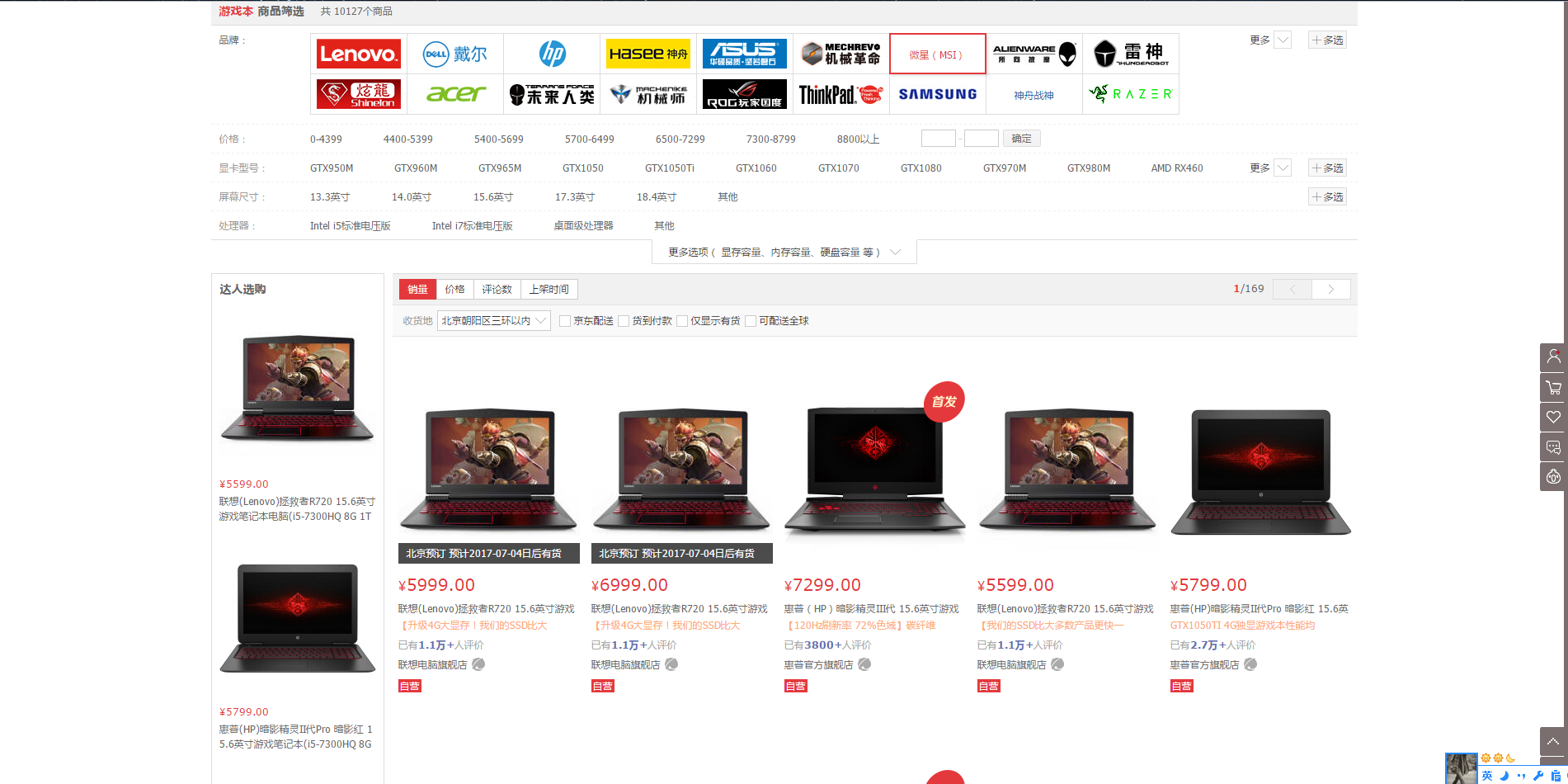
其中 crawler 是项目主目录，spiders 则存放京东和淘宝等爬虫，每个文件详细含义如下：

* scrapy.cfg Scrapy 项目的配置信息
* crawler.\_\_init\_\_.py 无特殊含义，仅用于标识一个包
* crawler.items.py 定义需要爬取的数据字段
* crawler.mongo.py 负责与 MongoDB 交互，包括创建连接，完成数据插入等
* crawler.pipelines.py 负责 item 的各项操作，包括字段检查、编码，插入 item 等
* crawler.settings.py 爬虫配置文件
* crawler.tb.sh 启动淘宝爬虫的脚本
* crawler.jd.sh 启动京东爬虫的脚本
* crawler.spiders.\_\_init\_\_.py 无特殊含义，仅用于标识一个包
* crawler.spiders.utils.py 工具类
* crawler.spiders.jd.py 京东爬虫
* crawler.spiders.tb.py 淘宝爬虫

## 项目难点

项目一个难点是如何找到销量前 20 的商品。对此，我们解决的方法是手工构造 URL。

如，对一个正常商品页面，如下：

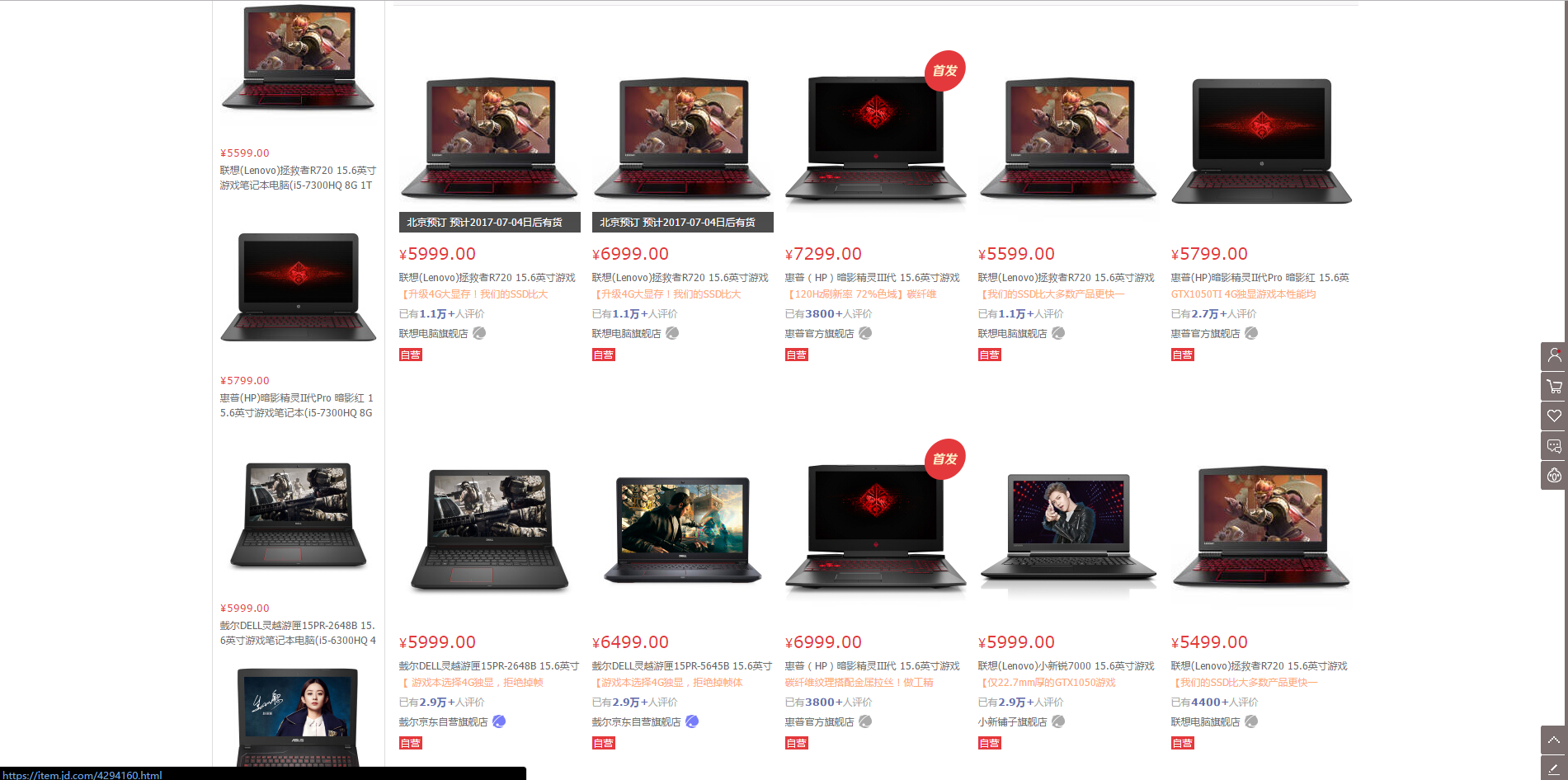


对应 URL 为：<https://list.jd.com/list.html?cat=670%2C671%2C1105&go=0>

如果我们尝试在该 URL 增加请求参数，构造如下 URL：

<https://list.jd.com/list.html?cat=670%2C671%2C1105&go=0&sort=sort_totalsales15_desc>

那么我们就得到按照销量排名的商品列表了：



对于淘宝爬虫，我们直接爬取淘宝排行榜，所以不需要这样处理。

## 爬虫爬取流程

以京东爬虫为例：

1. 首先在 start\_urls 中列出了所有的大类分类界面的 url。每个爬虫开启时都会调

用 start\_requests，处理 start\_urls 的 url。这里，我们覆盖实现了原来的

start\_requests。开启京东爬虫后，首先调用 start\_requests,用于从 start\_urls

来提取 url，封装成框架的 Request，指明回调函数为 parse()函数,然后交由爬虫

的 Scheduler 来处理

1. 第二步中，框架的 Scheduler 就会将上一步的 Request 发给 Downloader 处理，由

Downloader 来向服务器发送请求，下载页面，得到 response，然后调用京东的

parse()函数，来处理这个 response

1. 在 parse()中，我们得到了大类页面的 response，（大类分类页面图片见附录）解

析后我们从中提取中间页面的 url，即具有这样模式的

[r'.\*list\.jd\.com/list\.html']。继续向 Scheduler 发送 Request，这时指明回调

函数为 parse\_url

1. 类似第二步，Scheduler 将第三步的 Request 发送给 Downloader 来处理，由

Downloader 得到 response，然后调用京东爬虫的 parse\_url，来处理这个

response

1. 在 parse\_url()，我们得到了中间页面的 response（中间页面的图片见附录），解

析后我们从中提取最终分类页面的 url（最终页面的图片见附录）,对此 url 进行

处理，同样包装成 Request，指明回调函数 parse\_item

1. 类似地，得到 response 后，在 parse\_item 处理这个 response。这时，我们可以

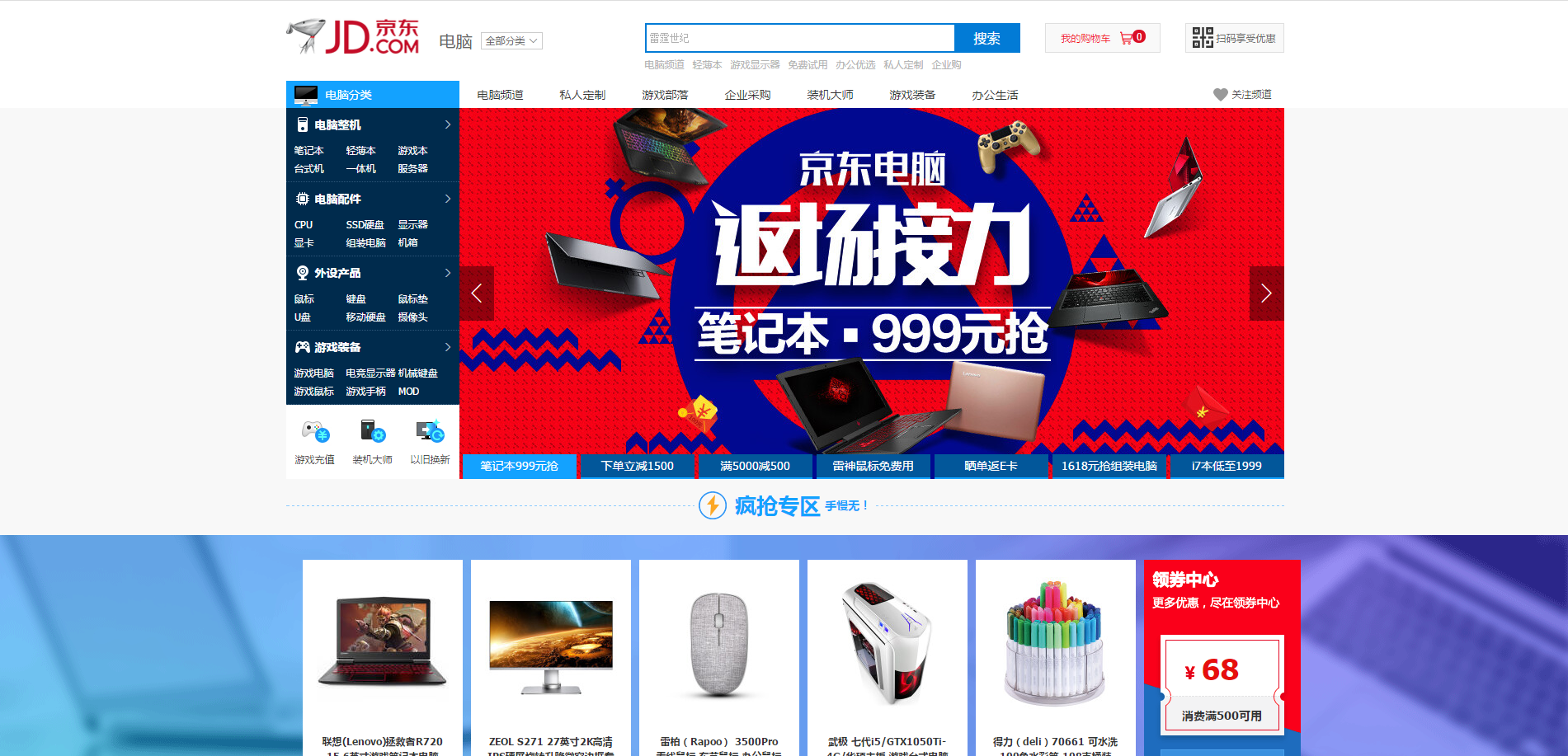
解析这个 response，提取排名前 20 的商品信息，并且将这些信息包装成 item 发

送给 Pipeline 处理。另外，我们在这个函数中继续提取 url，包装成 Request，让

Scheduler 处理

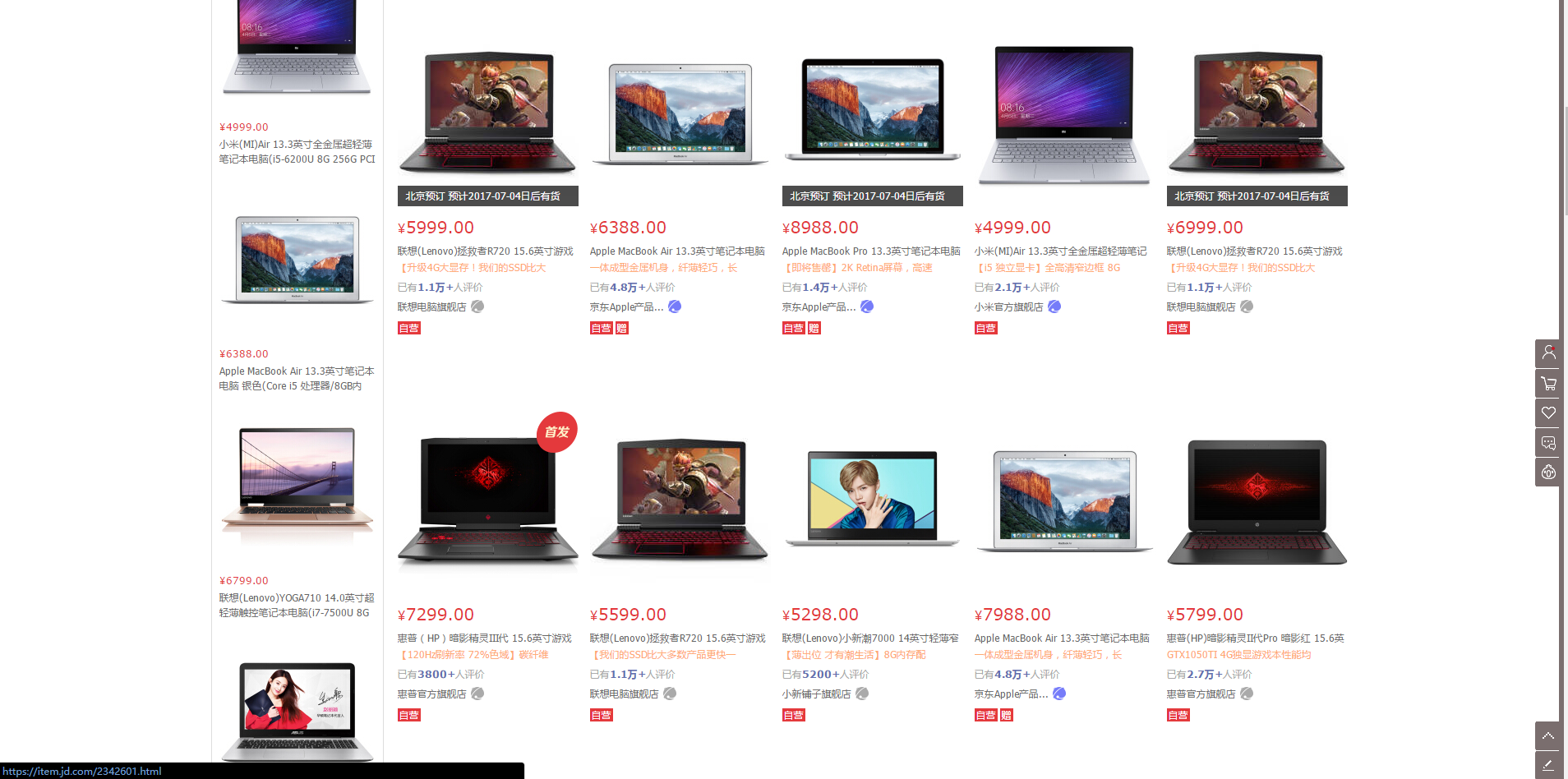
附页面说明：

1. 大类分类页面图片（以电子产品为例）：



1. 中间页面图片

在这里，我们需要对 URL 处理，使得商品按照销量排名：



得到上述页面，我们就可以基于该网页数据提取商品信息了。

## 网页种类和数据字段说明

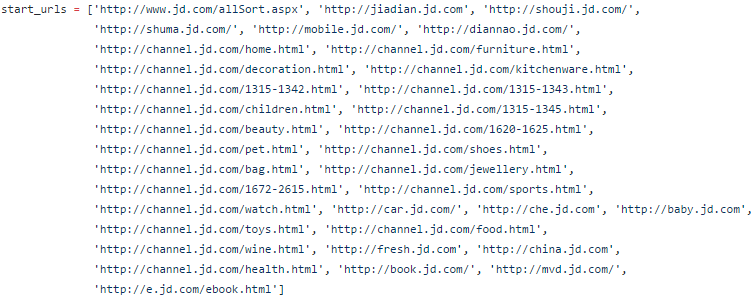
我们爬取了以下数据字段：

|  |  |
| --- | --- |
| 字段 | 含义 |
| title | 商品名称 |
| url | 商品 URL |
| rank | 销量排名 |
| top\_id | 商品所属第一层类别 |
| type\_id1 | 商品所属第二层类别 |
| type\_id2 | 商品所属第三层类别 |
| mall | 区分数据来源 |

一条示例数据如下：

{ "\_id" : ObjectId("5950cb3c92d4eb392e5b423b"), "title" : "探险者信封成人户外加厚超轻春秋冬野营装备用品旅行野外露营睡袋 2.0kg宝蓝色春亚纺", "url" : "http://list.jd.com/list.html?JL=4\_2\_0&cat=1318%2C1462%2C1474&page=1&trans=1", "top\_id" : "运动户外", "rank" : "19", "mall" : "2", "type\_id1" : "户外装备", "type\_id2" : "睡袋/吊床" }

对于京东爬虫，我们爬取了十多个商品大类，其中 start\_urls 定义如下：



对于淘宝爬虫，我们主要爬取淘宝排行榜，start\_urls 定义如下：



# 总结

我们在这个项目中利用 Scrapy 和 Splash 爬取了淘宝、京东等电商网站的商品信息。项目整体达到了预期效果，不过也存在一些问题，比如没有考虑反爬虫等措施。