2016-5-16

爬虫项目说明书

考试全队

队员：吴双 林伟强 江泽星 陈文源

何韦静

目录

1. 引言

1.1 编写目的

本文档介绍了本次参加此次比赛的作品。项目主要分为爬虫，前端和后台三个部分。通过阅读本文档，可以对此作品有一个清晰的了解。

2. 项目概述

2.1 背景

商家销售时，往往想知道目前行业内热销的货品是什么，针对热销的产品进行销售和补货。这里就希望获取到天猫、淘宝和京东等国内知名品牌电商的行业热销产品的排名，供市场进行参考。

2.2 项目的总述

本项目用 Python 实现的网络爬虫，对天猫、京东、淘宝国内三大知名电商网站进行了热门销售数据的采集，保存到后台，供前端网页查询，进行可视化的数据展示。

2.3 团队组成&分工

为了应对题目要求，我们团队由 4 个软件学院的学生和一个工业设计的学生组成。分工如下：

分工	负责
爬取数据	林伟强
设计数据存储方式，提供数据访问方式	陈文源
设计网页及展示效果	何韦静
根据网页设计图实现网页	江泽星
根据设计的展示效果图对爬取的数据进行可视化处理	吴双

2.4 开发环境

	操作系统	IDE	框架
前端	Windows	webstorm	bootstrap 3.55，jquery1.9，d3.js
爬虫	Ubuntu	pycharm	scrapy 1.0.5，splash0.6.1
后台	Ubuntu	IntelliJ IDEA	Spring MVC，Spring，Mybatis，Maven

3.项目详述

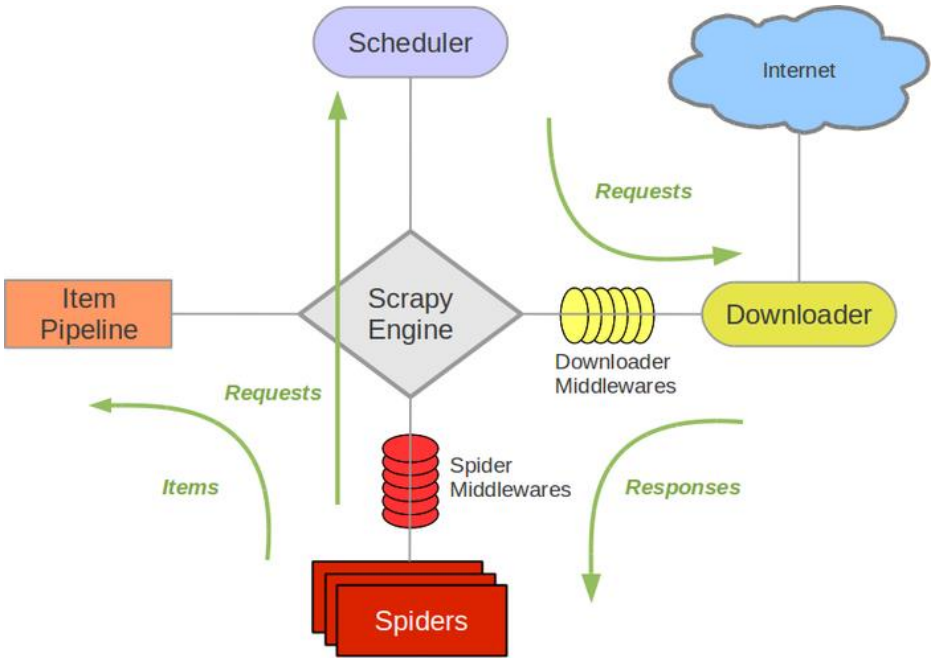
3.1 爬虫

3.1.1 整体思路：

淘宝，天猫，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，而在尝试查找 json 接口地址失败后，我选择了直接用浏览器内核引擎来加载 JS，从而获得原网页。在框架方面，我选了 scrapy 这个流行的 python 爬虫框架，而为了让 scrapy 爬虫能正确的解析出原网页，需要用到 Splash 库，这个库能让 scrapy 使用 Splash 的 HTTP API，从而能做到解析 JS。

3.1.2 scrapy 框架介绍：

本爬虫项目基于 scrapy 1.0.5
1.Scrapy 框架的基本架构如下图：



a) Scrapy 的数据流如下：
b) 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。
c) Engine 从 Spider 获取 URLs 后，就将其作为 Requests 放入 Scheduler 中调度。
d) 然后 Engine 向 Scheduler 索取下一部分的 URLs 来爬取。
e) 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）。
f) 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）。
g) Engine 收到 Response 就会将其送至 Spider 处理，期间通过 Spider Middleware(输入方向)。
h) Spider 会处理 Response，然后返回分析获得的 Items 和新的 Request 给 Engine。
i) Engine 就会将 Items 送至 Item Pipeline，将 Request 送至 Scheduler。
j) 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。

3.1.3 爬虫项目介绍：

crawler/

i. scrapy.cfg

ii. crawler/

1. __init__.py

2. items.py

3. pipelines.py
4. settings.py
5. db.py
6. spiders/
    i. tm.py
    ii. taobao.py
    iii. jd.py

items.py 定义了爬虫爬取数据存放数据的字段。pipelines.py 定义了对 item 的操作，包括对 item 各个字段的编码，是否为空，以及将 item 写入数据库等。settings.py 定义了这个项目的全局设置，包括了爬虫下载网页的最大时间，下载延迟，以及使用到的 Splash 中间件。db.py 定义了数据库的操作，包括创建表，插入数据等。

spiders 是个文件夹,用来存放爬虫。在这个文件夹。定义三个爬虫，分别为 taobao.py、tm.py、jd.py。对应为淘宝的爬虫，天猫的爬虫，京东的爬虫。

3.1.4 爬虫爬取流程

以京东的爬虫为例：

一. 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，用于处理 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

二. 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

三. 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

四. 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

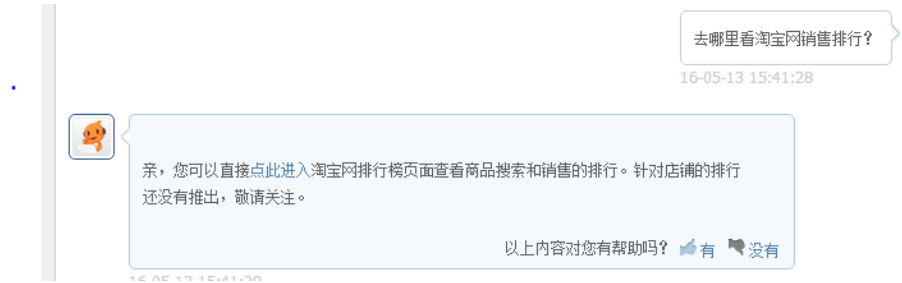
五. 在 parse_url()，我们得到了中间页面的 response（中间页面的图片见附录），解析后我们从中提取最终分类页面的 url（最终页面的图片见附录）,对此 url 进行

处理，同样包装成 Request，指明回调函数 parse_item

六. 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

京东爬虫的流程大致如上，其他爬虫可以类似参考。

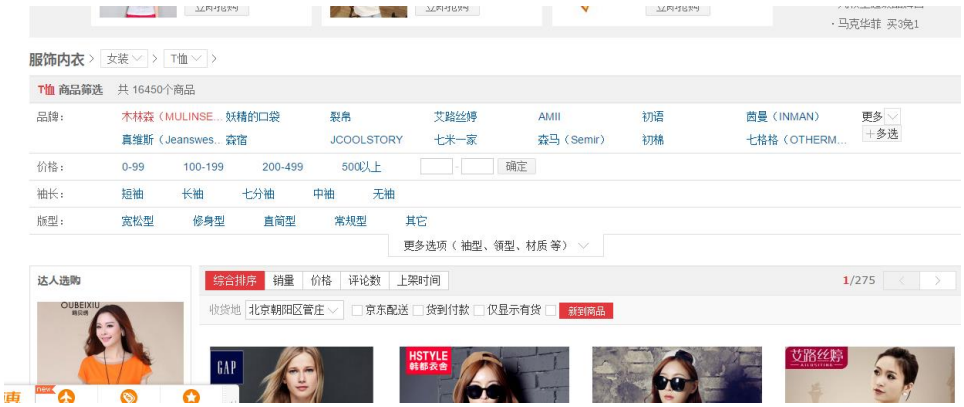
京东天猫爬取的网站是官网，淘宝我们选择的是淘宝销量排行榜，更加贴合题目对于销量的要求



附页面说明：

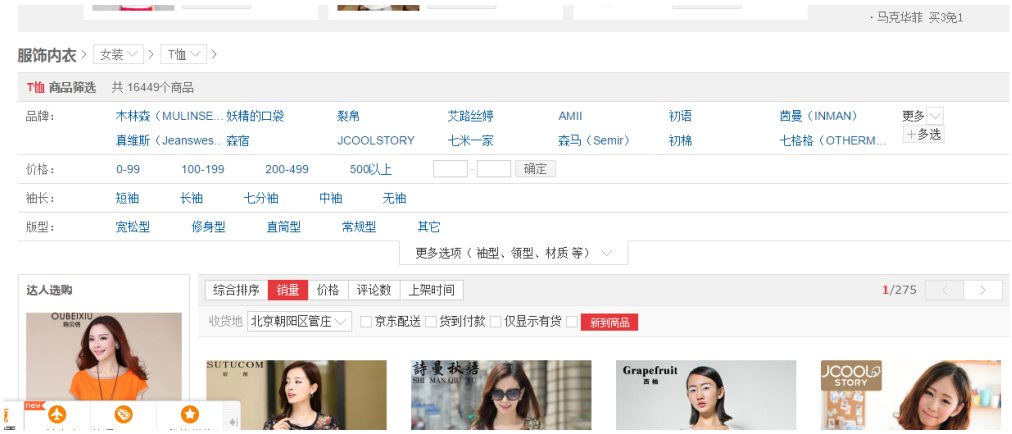
1. 大类分类页面图片（以女装类为例）

2. 中间页面图片



注意，这里展示商品时不是按销量来排名的。

3. 最终分类页面图片



3.1.5 爬虫数据库介绍

名字	Field	说明
id	int(11),NOT NULL,PRIMARY KEY,AUTO_INCREMENT	
mall	int(11)	用来区分商城，0 为淘宝，1 为天猫，2 为京东
rank	int(11)	每个商品在它分类下的排名
title	varchar(250)	商品名称
price	varchar(10)	商品价格
turnover_index	double	淘宝排行榜的成交指数
top_id	varchar(200)	第一层分类
type_id1	varchar(200)	第二层分类
type_id2	varchar(200)	第三层分类

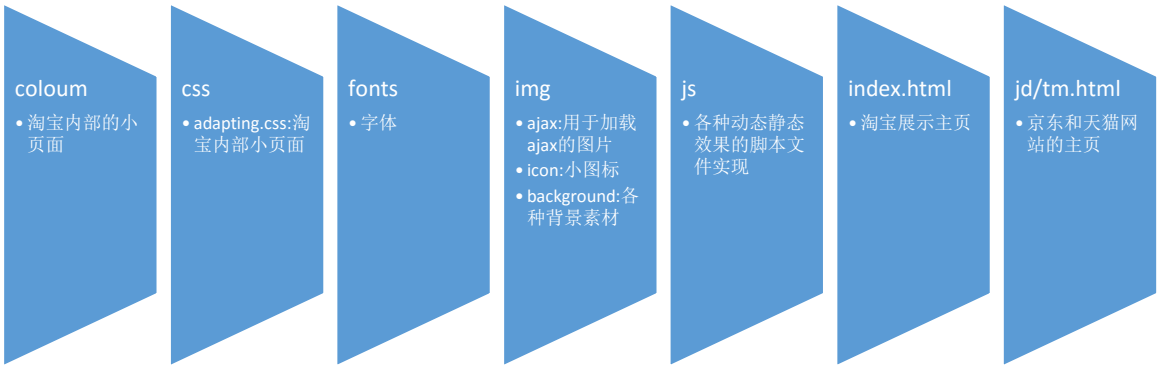
url	varchar(200)	商品分类排行页面的url

3.2 网页端

3.2.1 整体介绍

前端主要是将爬虫爬到的数据进行可视化。由于爬虫爬取了三个热门电商平台，导致数据量在百万级别，不可能在短时间内一一排版设计，故我们主要实现了淘宝的网页设计，京东和天猫的展示选择了用数据驱动的文档（Data-Driven Documents，简称 d3）的形式来展示，即以爬取的数据生成对应的展示图，实现大数据的可视化处理。在静态网页(html,css)的基础上使用 d3 的数据绑定的特性将排名展示 ,具有更大的灵活性和可扩展性。

3.2.2 代码目录说明



3.2.3 网页设计思路

这是一个展示销量排名数据的网站。由于要将淘宝、京东和天猫的数据分开展示，我们将网站分为了几个部分。同时，这个网站同时具备导航及数据展示的功能。在设计过程中我们思考将两部分功能完整而清晰的为用户展现，并将其和谐的连接起来。

由于用户需要根据其自身需要，经过几次的分类选择才能看到最终的数据，网页的标准流的设计是我们首先考虑的重点。我们设计了浮动标题栏来为用户做出有效引导。用户通过点击文字选区可简单快捷的实现各页面的跳转，并保证各分类的有效进行，避免用户不知道自己到底在哪个界面应该干什么。因所占位置较小，使用文字使指示清晰明了，易于用户辨认。

我们先介绍淘宝数据排名展示部分的设计思路。我们将淘宝内产品行业分为八个大类，分别为数码家电、服饰、化妆品、母婴、食品、文体、家居、车/玩具/宠物。在每个大类之下，我们根据每个行业的特点和情况，进而分出了许多小类。我们为每个大类

设计了图标，方便用户理解。我们还为每个大类选择了不同的主题色。在首页，我们将八个大类的图标和文字于同一水平排列。光标放在每一大类上，相应区域会显示对应的主题色，形成模块间的对比，可以很好地帮助用户知道可以转到哪里，也是整个配色方案显得更加有活力。

每一大类有独立的页面组成。从首页跳转到每一大类的页面时，首先显示每一大类的封面图，帮助用户确认所选分类。Icon 下设计了箭头，给用户有效的指引。点击箭头，页面将平滑下滑至小分类页面。考虑到使用的任何一个下拉框都会对用户造成信息的隐藏而需要额外的操作才能显示，因此我们将每个小分类下更细的分类全部展示出来，减少用户所需的操作，使信息更加显而易见。光标扫到的每个细分类都会变颜色，提醒用户知道自己可以选到那些页面。一旦用户点击一个细分类，页面将下滑至最终的数据展示页面。

数据展示页面由柱状图组成。柱状图的中轴为 1 到 20 的数字，左边为前 20 的商品名，右边为每件商品对应的销量。这样直观清晰而全面的展示了用户所需要的数据，有效的帮助用户进行分析。另外我们为柱状图设计的动态效果，有效的为用户视线做出引导。

在京东和天猫的网页设计上，由于其分类较繁多，为了使用户有更清晰的判断，我们在淘宝销售排名展示网页的基础上，进一步简化了整体的设计。整体配色采用原网站（天猫或京东）logo 的主配色，给用户更熟悉的感觉。首先，封面图指明了用户所在位置。接着，箭头给予用户明确有效的指引。点击箭头后，页面同样向下滑动至分类页面。为了使分类更有条理，我们采用了树状图的设计。用户点击所选中的分类后，树状图进行动态展开。展开后，使用红线清晰指出用户的路径，也保持了整体布局的稳定。在用户点击了最细的分类时，页面产生该类的前二十排名。此处数据展示采用直接展示产品名的方式，左边直接列出产品排名，简单明了，右边直接列出排名对应的产品。

3.2.4 页面说明

Index.html

Figure 1 index.html

- 该页面是淘宝的一级分类页面,将淘宝的大类分为八个主题,用户可以通过点击图标或者导航栏进入一级分类。
- 用户可以点击右上角其他商城展示页面跳转链接

一级分类.html

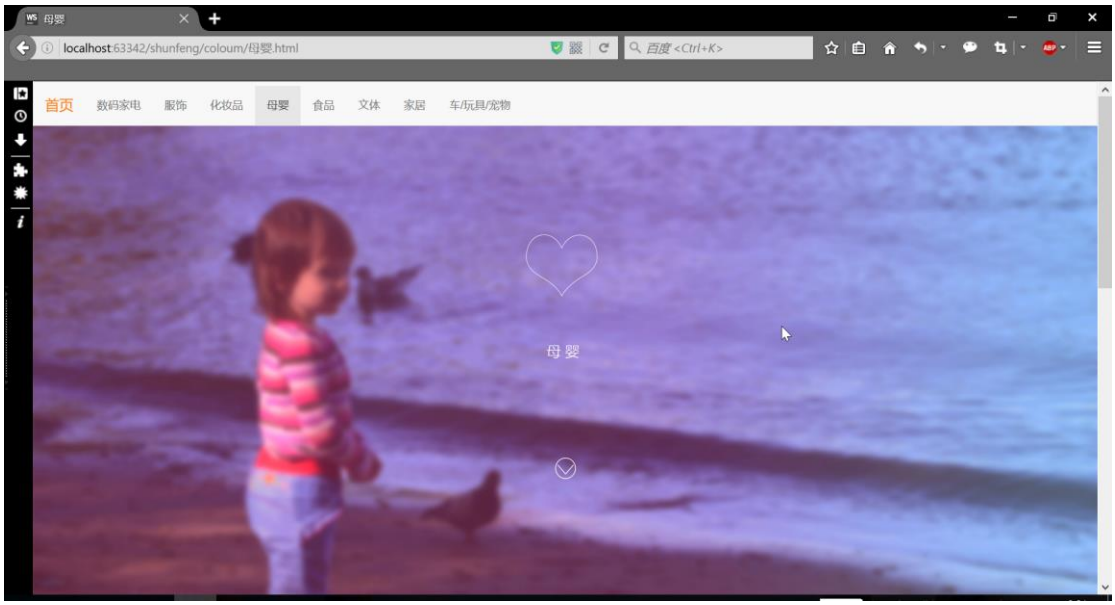


Figure 2 **淘宝一级分类** 1

- 点击箭头页面平缓花香一级分类 **2**

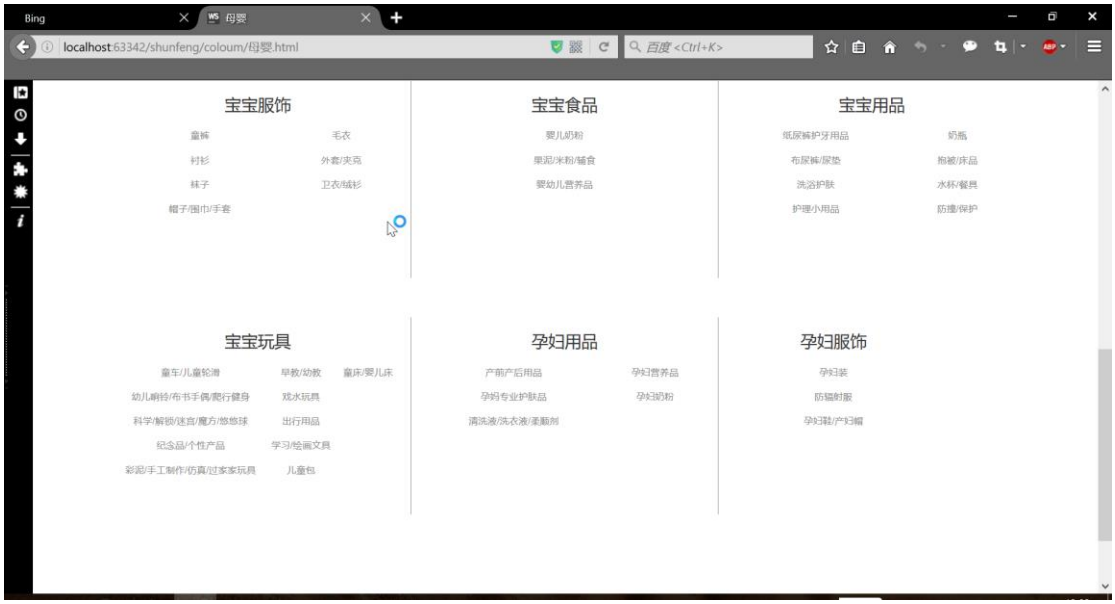


Figure 3 **淘宝一级分类** 2

- 点击小类显示排行页面滑动到小项的前二十名柱状图。

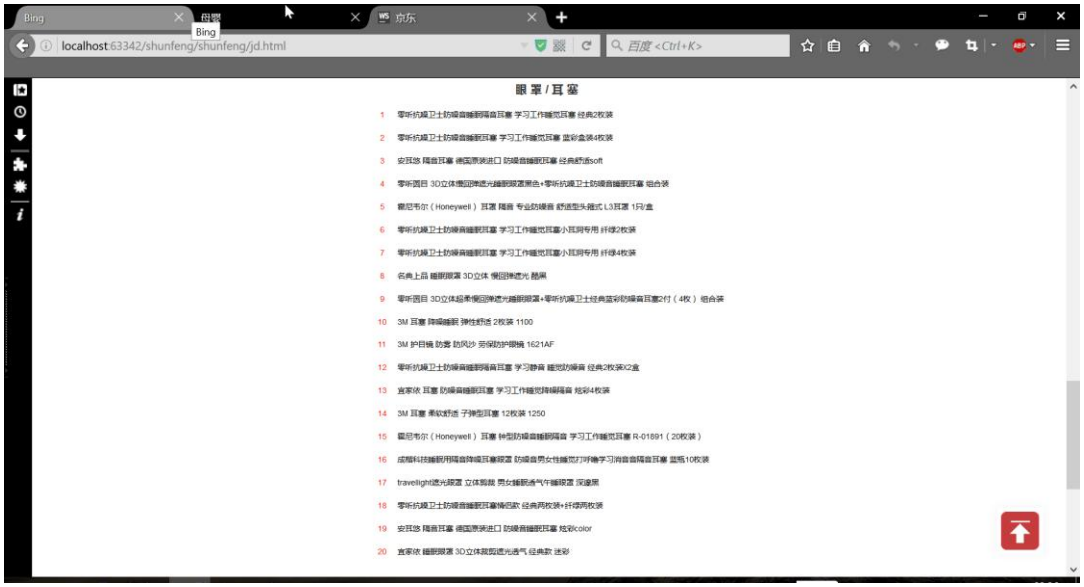
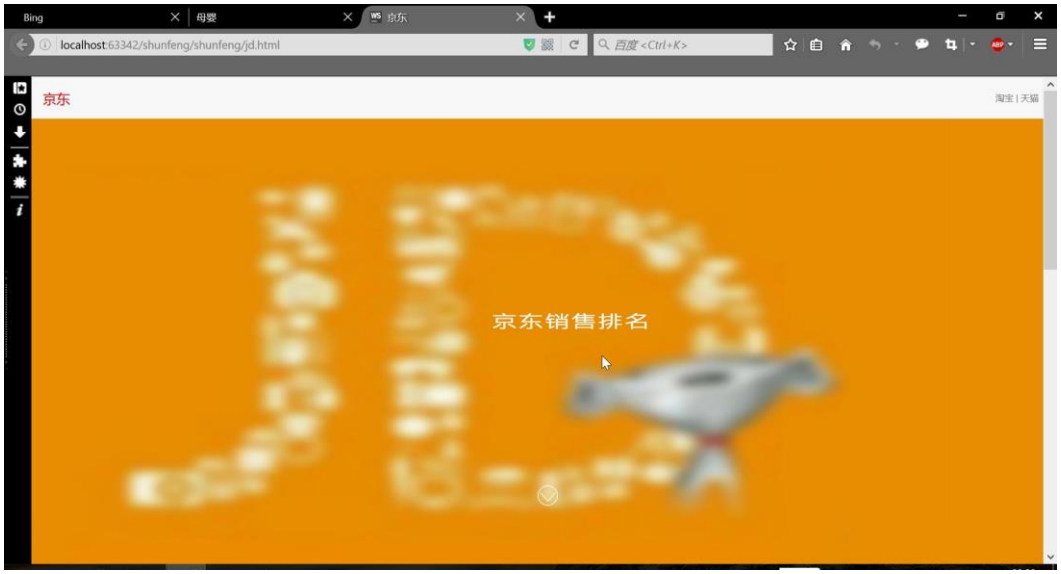
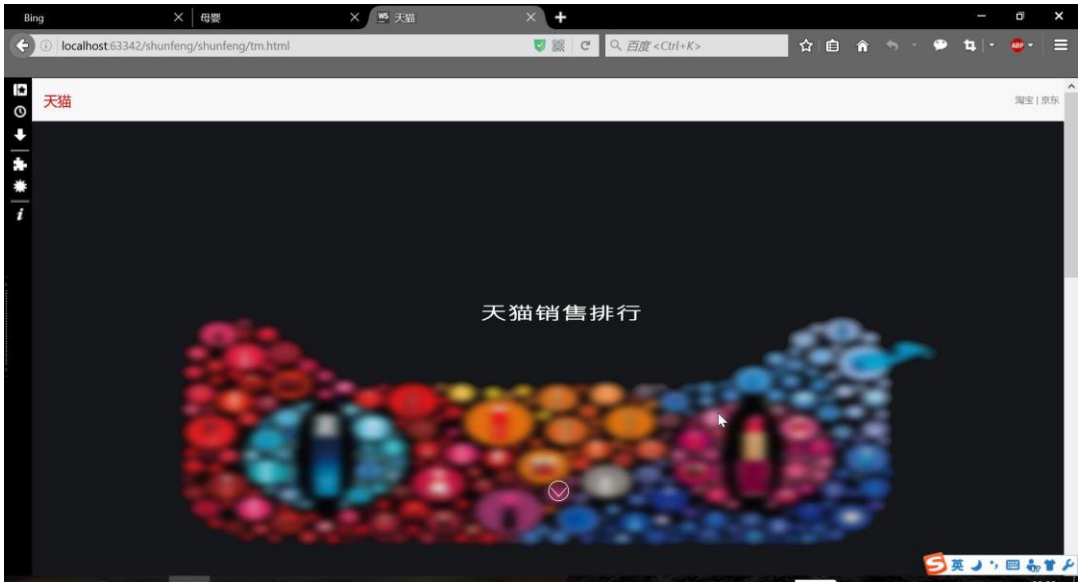


Figure 4 **淘宝数据展示部分**

- 该部分是请求后台数据,获得后展示的前二十名排名的商品和销量.
- 点击商品名页面跳转至爬虫爬取的源页面
- 点击右边按钮页面回到商品树顶端

Jd.html&tm.html





- 该页面是京东&天猫的主要页面
- 点击箭头滑至京东商品分类树状图

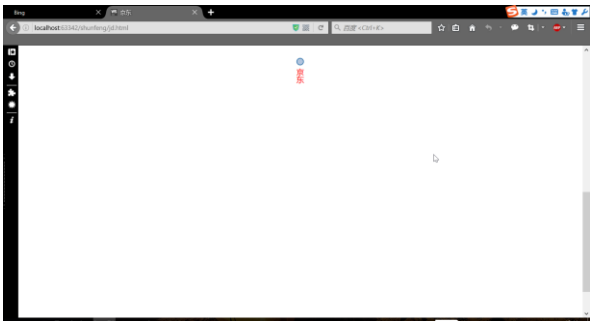


Figure 5 商品分类树状图 1

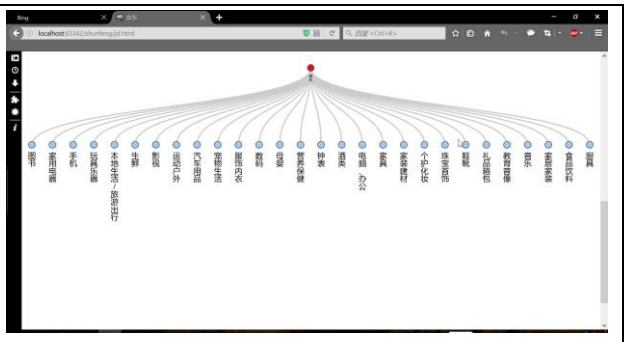


Figure 6 商品分类树状图 2

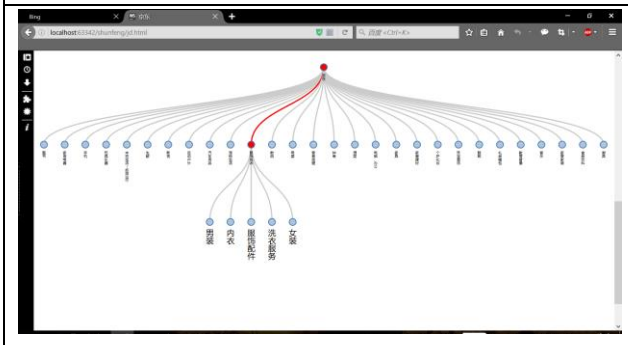


Figure 7 商品分类树状图 3

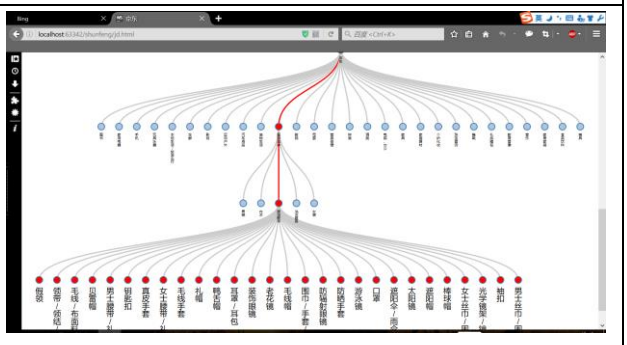


Figure 8 商品分类树状图 4

- 我们通过 d3 绑定后台分类数据产生三级分类的树状图，以此来展示具体的商品分类
- 难点：树状图的交互 。每次点击都通过改变对应的数据来改变树状图的结构，达到交

互的效果。每次只展示一项是为了避免过多项目照成画面拥挤

- 点击最后一级的小分类，产生指定商品前二十名的排名

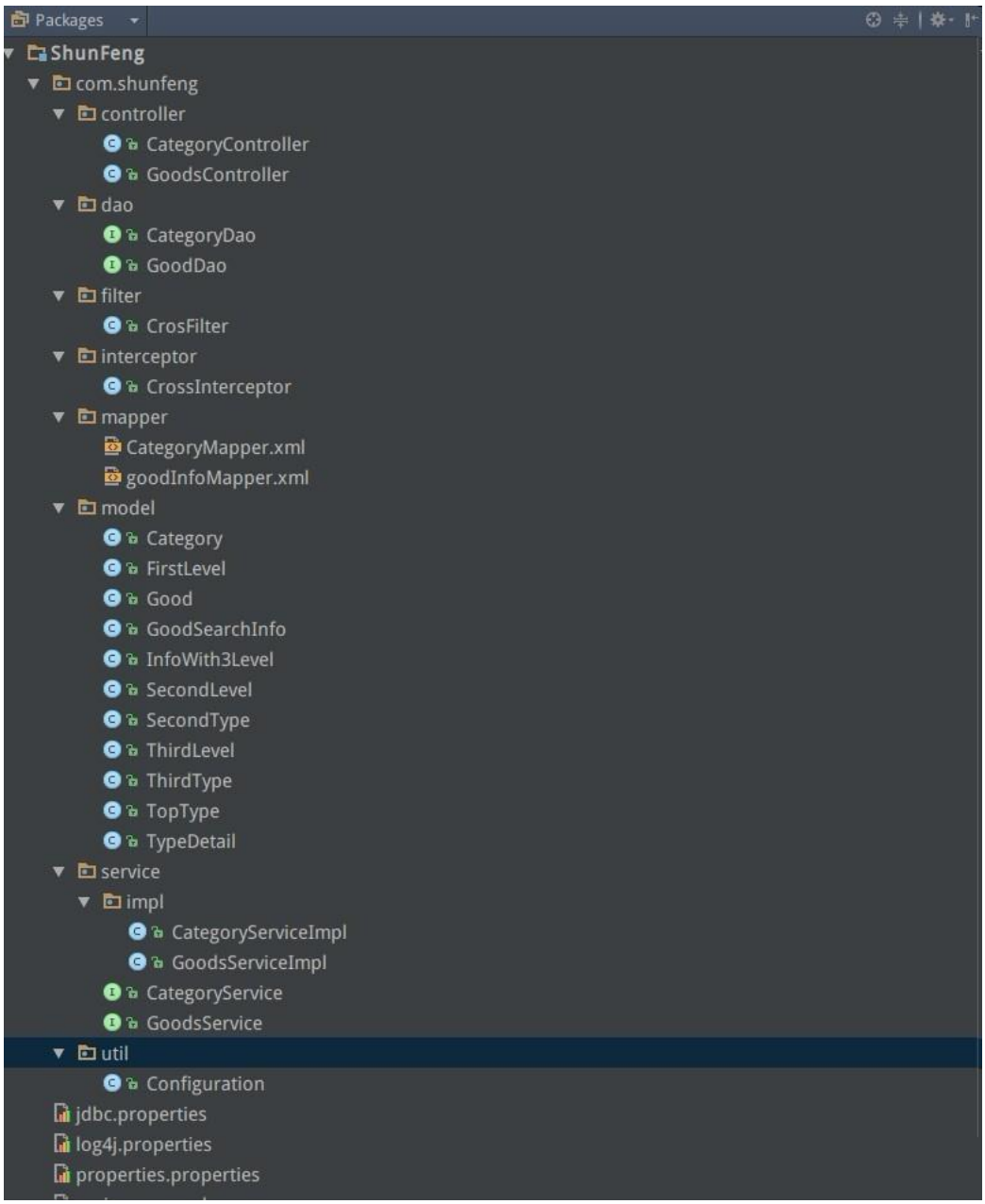
- 商品的前二十名展示,左边是销量排名,右边是对应的商品名,点击商品名即可获得商品相关的链接。

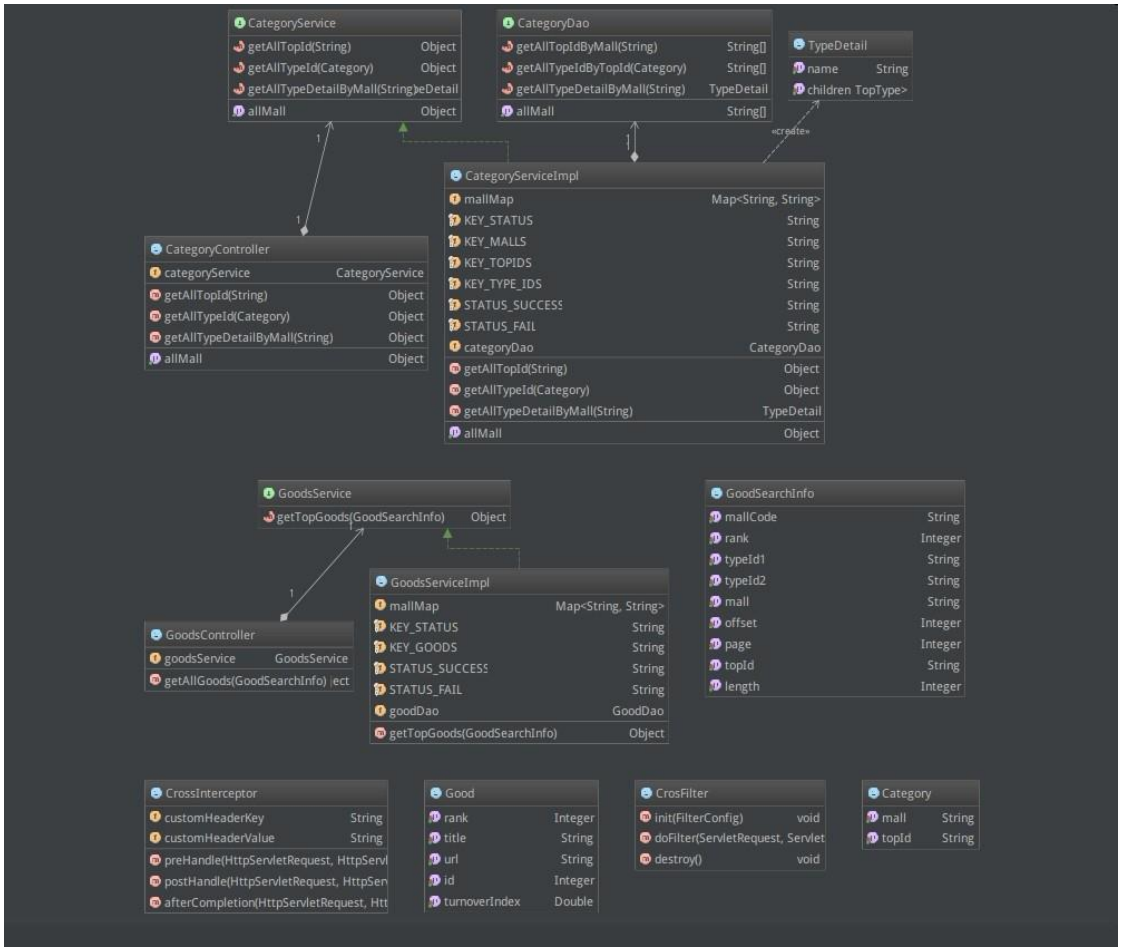
3.3 后台

3.3.1 整体介绍

前端向后台发送相应的请求，后台服务从请求中获取相应的参数，并且根据请求跳转到对应的控制器，控制器启动对应的服务向数据库发起连接，并且取出数据，再对数据进行相应的业务逻辑处理，封装产生的结果，最终以 JSON 格式返回数据。

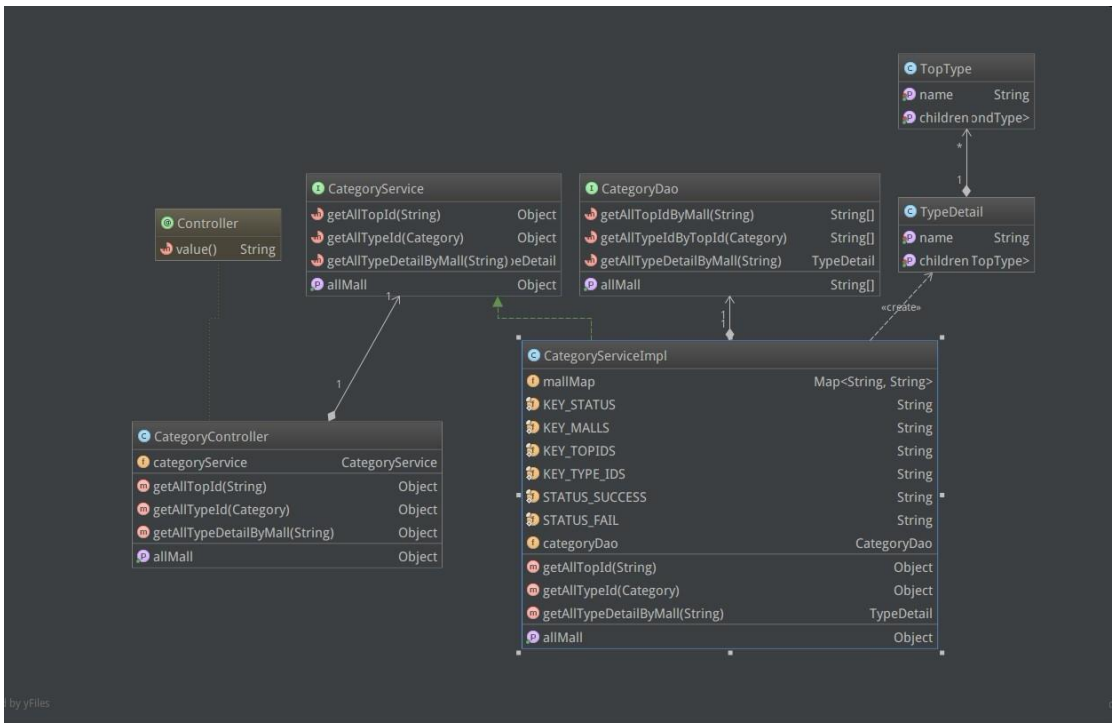
3.3.2 具体类图以及类依赖关系




CategoryService
getAllTopId(String) Object
getAllTypeId(Category) Object
getAllTypeDetailByMall(String) TypeDetail
allMall Object
CategoryDao
getAllTopIdByMall(String) String[]
getAllTypeIdByTopId(Category) String[]
getAllTypeDetailByMall(String) TypeDetail
allMall String[]
TypeDetail
name String
children TopType>
«create»
CategoryServiceImpl
mallMap Map<String, String>
KEY_STATUS String
KEY_MALLS String
KEY_TOPIDS String
KEY_TYPE_IDS String
STATUS_SUCCESS String
STATUS_FAIL String
categoryDao CategoryDao
getAllTopId(String) Object
getAllTypeId(Category) Object
getAllTypeDetailByMall(String) TypeDetail
allMall Object
CategoryController
categoryService CategoryService
getAllTopId(String) Object
getAllTypeId(Category) Object
getAllTypeDetailByMall(String) Object
allMall Object
GoodsService
getTopGoods(GoodSearchInfo) Object
GoodsServiceImpl
mallMap Map<String, String>
KEY_STATUS String
KEY_GOODS String
STATUS_SUCCESS String
STATUS_FAIL String
goodDao GoodDao
getTopGoods(GoodSearchInfo) Object
GoodSearchInfo
mallCode String
rank Integer
typeId1 String
typeId2 String
mall String
offset Integer
page Integer
topId String
length Integer
GoodsController
goodsService GoodsService
getAllGoods(GoodSearchInfo) ect
CrossInterceptor
customHeaderKey String
customHeaderValue String
preHandle(HttpServletRequest, HttpServl
postHandle(HttpServletRequest, HttpSer
afterCompletion(HttpServletRequest, Htt
Good
rank Integer
title String
url String
id Integer
turnoverIndex Double
CrosFilter
init(FilterConfig) void
doFilter(ServletRequest, Servlet
destroy() void
Category
mall String
topId String

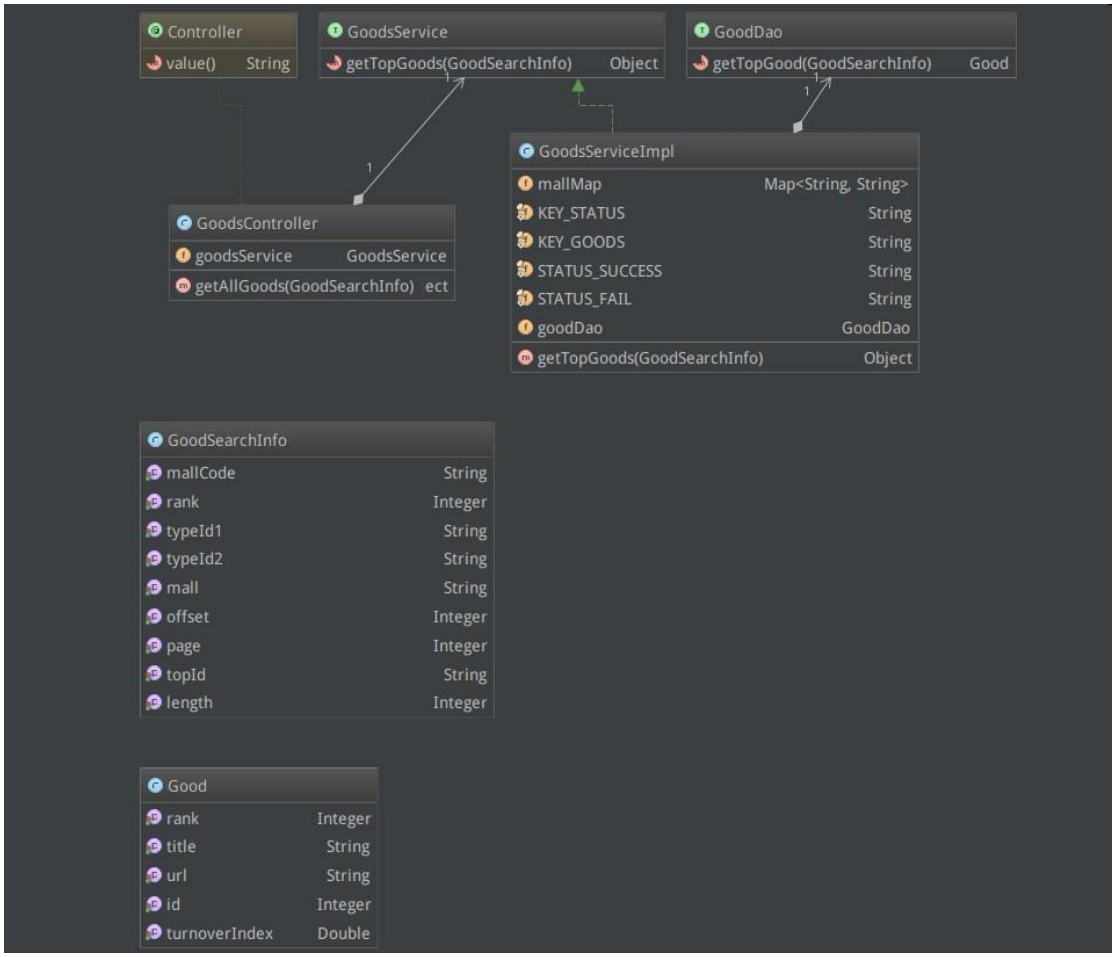
3.3.3 **具体处理流程**:

获取类别的处理流程



前端发送请求到后台，经过 Filter 管道过滤后由 DispatcherServlet 分发到 CategroyController，CategroyController 调用 CategoryService 进行服务，在 CategoryService 中，将建立起与数据库的连接，交互并且获取数据，并且将原始数据进行处理，进一步封装成对应的种类信息，其中种类信息一共对应 3 个级别，分别对应 TopType,SecondType,ThirdType,最后将其封装成一个 TypeDetail 对象返回到 CategoryController,并且经过封装处理形成遵循 JSON 格式的字符串返回给前端。

获取具体商品信息的处理流程



首先 GoodsController 接收前端发来的 Request，并且将其传递进来的参数封装成一个 GoodSearchInfo 对象，接下来 GoodsController 把 GoodSearchInfo 对象传递给 GoodsService 接口的实现类 GoodsServiceImpl 对象，由 GoodsService 通过对 GoodSearchInfo 进行处理加工再传递给 GoodDao 对象，让其与数据库连接并且提取原始数据，并且将原始数据映射到 Good 类的各个属性成员，封装完后再返回给 GoodsService，最后 GoodsService 将已经封装好的 Good 对象返回给调用它的 GoodsController，GoodsController 把 Good 返回，最后 Good 经过 JSON 封装处理形成遵循 JSON 格式的字符串返回给前端。

4. 总结与展望

作为短时间内做完的作品，我们深知还有许多能完善的地方。只要给我们时间，我们还能做得更好！

4.1 爬虫部分

总结

1. 初期没有完全考虑清楚数据规模，把三个商城的数据都存在了一个表，导致后期一个表具有了百万的数据，不得不把淘宝、京东和天猫的数据分开。
2. 爬虫爬取数据时，第一个问题是，要注意检查当前页面的商品排名是不是按照销量来排名的，因为京东、天猫等还有综合排名、价格排名等，而这些排名都不是我们需要的；第二个问题，即使当前商品是按照销量进行排名，还需要检查当前页面是不是这个分类排名的第一页，因为我们需要提取的是每个小类销量前 20 的商品信息，如果当前页面的排名是 20 名后的商品，那么这个页面对我们来说就是没用的；第三个问题是，京东、天猫等还有按照是否有货，是否包邮的选项，不能让这些干扰到销量的排名。而上述的问题都可以构造 url 参数来实现我们的要求：当前页面按照销量来排名，是当前分类的第一页，当时没有货到付款、包邮、有货等选项干扰。
3. 经检查，京东和天猫的所有分类并不能完全爬下来，还是有一些小的分类并没有爬到。
4 .另外，检查数据库发现，京东和天猫的数据存在许多冗余，即同一小类的数据，多次出现在了数据库了，而多出来的这部分数据完全无用，还会对后台查询负担。

展望

1. 进一步修改爬虫的爬取规则，增加爬虫的爬取深度，这里包括提取不同的 url，让爬虫能爬取到所有的分类
2. 个人猜测：因为使用了 Splash 库，导致原来 scrapy 框架的 url 去重机制失效。所以，我们需要自己实现一个爬虫中间件，在这个中间件实现 url 去重机制，这样就不会爬取到重复的 url 了

4.2 前端部分

总结

1. 数据展示方法是前端的亮点,我们从设计到实现上在此花费的功夫比较多,我们采用了目前比较流行用于数据可视化的 js 框架：d3.js,通过利用其数据绑定的特性,将数据和展示合二为一。
2. 在构建淘宝一级分类的专题页面的时候，由于页面结构大致相同，仅有背景图，专属颜色，二级分类等有差别，我们用通过不同的 title 来标识本页面，然后用 javascript 根据 title 的不同进行不同的页面渲染，减少了重复代码，同时也便于前端的扩充和修改。
3. 京东和天猫采用树状图目录展示减轻了网页编写的负担，只要爬到数据就能自动生成对应的目录树，有强大的适用性。该图编写的核心在于理解 d3 的绑定数据机制，在此花费的精力比较多此外，文本竖直排版也是一个难点，最后用 tspan 解决。

展望

1. 增强复用性：淘宝一级分页可以模板化，比如使用 python 的模板语言，可以进一步减少重复代码。
2. 界面优化：天猫和京东的页面可以进一步美化。
3. 访问效率优化：压缩图片，js，css 等代码加快访问速度，前端代码进一步重构，减少请求次数等。
4. 根据后台信息，扩展更多的交互元素。

4.2 后台部分

总结

1 .后台采取统一返回 JSON 格式数据的接口降低前端与后台的耦合性，有利于前端与后台专注各自的开发与工作。

2. 后台通过 MyBatis 实现查询结果与对象相映射，降低直接查询的次数，避免 N+1 查询，提高处理效率。

3. 科学地为数据库建立索引，在查询数据与增改数据中尽量保持平衡点，提高查询性能，加快响应速度。展望

展望

1. 增强参数验证，提高接口安全性与稳定性。

2.将数据库中的表重新合理地划分，避免过大的数据表拖慢查询效率。

3.研究更多相关的分析算法并且与实际存储的数据相结合，从中挖掘分析并提取出有用的信息。

5.项目展示

线上展示网址：http://119.29.158.92/sf/index.html

注：1.用火狐打开效果最好，IE 对 bootstrap 的兼容性效果不是很理想

2.我们会继续完善，也许会有和文档描述不一样的地方，那说明我们更新了:)