**18. 数据采集-爬取电商评论数据**

**一、什么是爬虫**

网络爬虫，也叫网络蜘蛛（spider），是一种用来自动浏览万维网的网络机器人。其目的一般为编纂网络索引。

网络搜索引擎等站点通过爬虫软件更新自身的网站内容或其对其他网站的索引。网络爬虫可以将自己所访问的页面保存下来，以便搜索引擎事后生成索引供用户搜索。

爬虫访问网站的过程会消耗目标系统资源。不少网络系统并不默许爬虫工作。因此在访问大量页面时，爬虫需要考虑到规划、负载，还需要讲“礼貌”。 不愿意被爬虫访问、被爬虫主人知晓的公开站点可以使用robots.txt文件之类的方法避免访问。这个文件可以要求机器人只对网站的一部分进行索引，或完全不作处理。

互联网上的页面极多，即使是最大的爬虫系统也无法做出完整的索引。因此在公元2000年之前的万维网出现初期，搜索引擎经常找不到多少相关结果。现在的搜索引擎在这方面已经进步很多，能够即刻给出高质量结果。

爬虫还可以验证超链接和HTML代码，用于网络抓取。

**二、利用爬虫可以做哪些有趣的事**

**1. 爬取古诗文http://so.gushiwen.org/mingju/ju\_4652.aspx**

《小松》 唐代 杜荀鹤

自小刺头深草里，而今渐觉出蓬蒿。

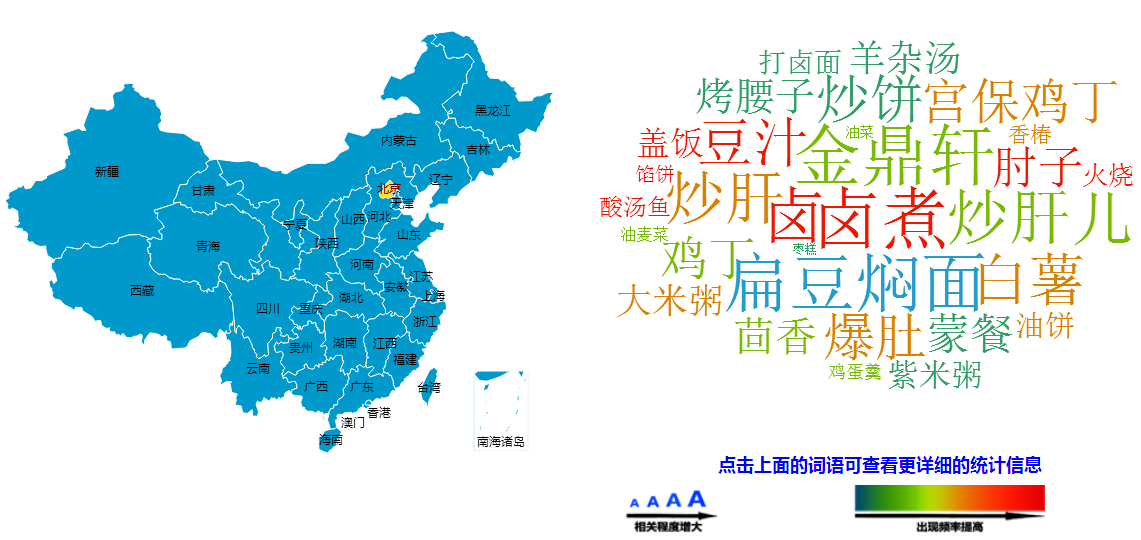
时人不识凌云木，直待凌云始道高。

**2. 爬取电商数据： 如意淘、惠惠购物助手、西贴、购物党http://www.xitie.com/**





**3. 爬取社会化媒体 http://ys.8wss.com/**



4. 爬取金融数据，量化分析

5. 爬取新闻数据，舆情、文章聚合等

**三、常见的爬虫框架**

1. Java语言开发的分布式爬虫：Nutch

分布式爬虫，主要是解决两个方面的问题：①海量URL管理 ②网速。

Nutch一般应用于搜索引擎

Nutch依赖于Hadoop运行

2. JAVA语言开发的单机爬虫：Crawler4j、WebMagic、WebCollector

3. Python语言开发的单机爬虫：scrapy、PySpider

[](http://192.168.1.241/doc/exper/125/img/04.png)

4. 自己动手，编写爬虫

**四、关于webMagic**

WebMagic是一个简单灵活的Java爬虫框架。基于WebMagic，你可以快速开发出一个高效、易维护的爬虫。

WebMagic的设计参考了业界最优秀的爬虫Scrapy，而实现则应用了HttpClient、Jsoup等Java世界最成熟的工具，目标就是做一个Java语言Web爬虫的教科书般的实现。

网站：http://webmagic.io/

中文文档：http://webmagic.io/docs/zh/

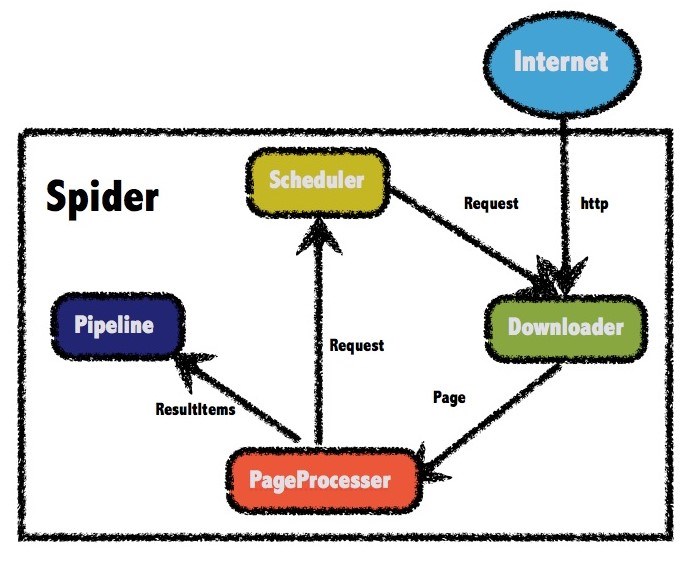
**五、爬虫架构**

WebMagic的结构分为Downloader、PageProcessor、Scheduler、Pipeline四大组件，并由Spider将它们彼此组织起来。这四大组件对应爬虫生命周期中的下载、处理、管理和持久化等功能。

WebMagic的设计参考了Scrapy，但是实现方式更Java化一些。

而Spider则将这几个组件组织起来，让它们可以互相交互，流程化的执行，可以认为Spider是一个大的容器，它也是WebMagic逻辑的核心。

WebMagic总体架构图如下：

[](http://192.168.1.241/doc/exper/125/img/05.jpg)

**组件讲解**

1. Downloader

Downloader负责从互联网上下载页面，以便后续处理。WebMagic默认使用了Apache HttpClient作为下载工具。

2. PageProcessor

PageProcessor负责解析页面，抽取有用信息，以及发现新的链接。WebMagic使用Jsoup作为HTML解析工具，并基于其开发了解析XPath的工具Xsoup。

在这四个组件中，PageProcessor对于每个站点每个页面都不一样，是需要使用者定制的部分。

3. Scheduler

Scheduler负责管理待抓取的URL，以及一些去重的工作。WebMagic默认提供了JDK的内存队列来管理URL，并用集合来进行去重。也支持使用Redis进行分布式管理。

除非项目有一些特殊的分布式需求，否则无需自己定制Scheduler。

4. Pipeline

Pipeline负责抽取结果的处理，包括计算、持久化到文件、数据库等。WebMagic默认提供了“输出到控制台”和“保存到文件”两种结果处理方案。

Pipeline定义了结果保存的方式，如果你要保存到指定数据库，则需要编写对应的Pipeline。对于一类需求一般只需编写一个Pipeline。

**六、爬取策略**

1. 并行策略

一个并行爬虫是并行运行多个进程的爬虫。它的目标是最大化下载的速度，同时尽量减少并行的开销和下载重复的页面。为了避免下载一个页面两次，爬虫系统需要策略来处理爬虫运行时新发现的URL，因为同一个URL地址，可能被不同的爬虫进程抓到。

2. 重新访问策略

网站的属性之一就是经常动态变化，而爬取网站的一小部分往往需要花费几个星期或者几个月。等到网站爬虫完成它的爬取，很多事件也已经发生了，包括增加、更新和删除。 在搜索引擎的角度，因为没有检测这些变化，会导致存储了过期资源的代价。

3. 平衡礼貌策略

爬虫相比于人，可以有更快的检索速度和更深的层次，所以，他们可能使一个站点瘫痪。不需要说一个单独的爬虫一秒钟要执行多条请求，下载大的文件。一个服务器也会很难响应多线程爬虫的请求。 就像Koster所注意的那样，爬虫的使用对很多任务作都是很有用的，但是对一般的社区，也需要付出代价。使用爬虫的代价包括：

网络资源：在很长一段时间，爬虫使用相当的带宽高度并行地工作。

服务器超载：尤其是对给定服务器的访问过高时。

质量糟糕的爬虫，可能导致服务器或者路由器瘫痪，或者会尝试下载自己无法处理的页面。

个人爬虫，如果过多的人使用，可能导致网络或者服务器阻塞。

对这些问题的局部解决方法是漫游器排除协议（Robots exclusion protocol），也被称为robots.txt议定书。

4. 选择策略

链接跟随限制

URL规范化

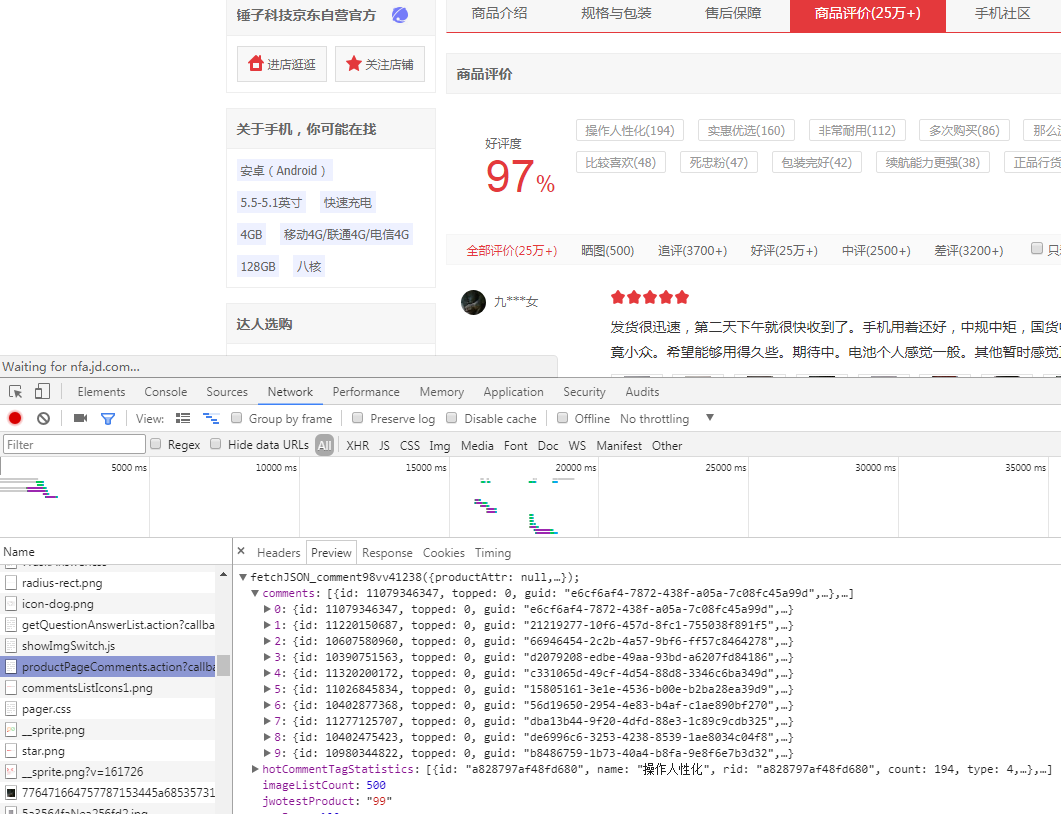
路径上移爬取

主题爬取

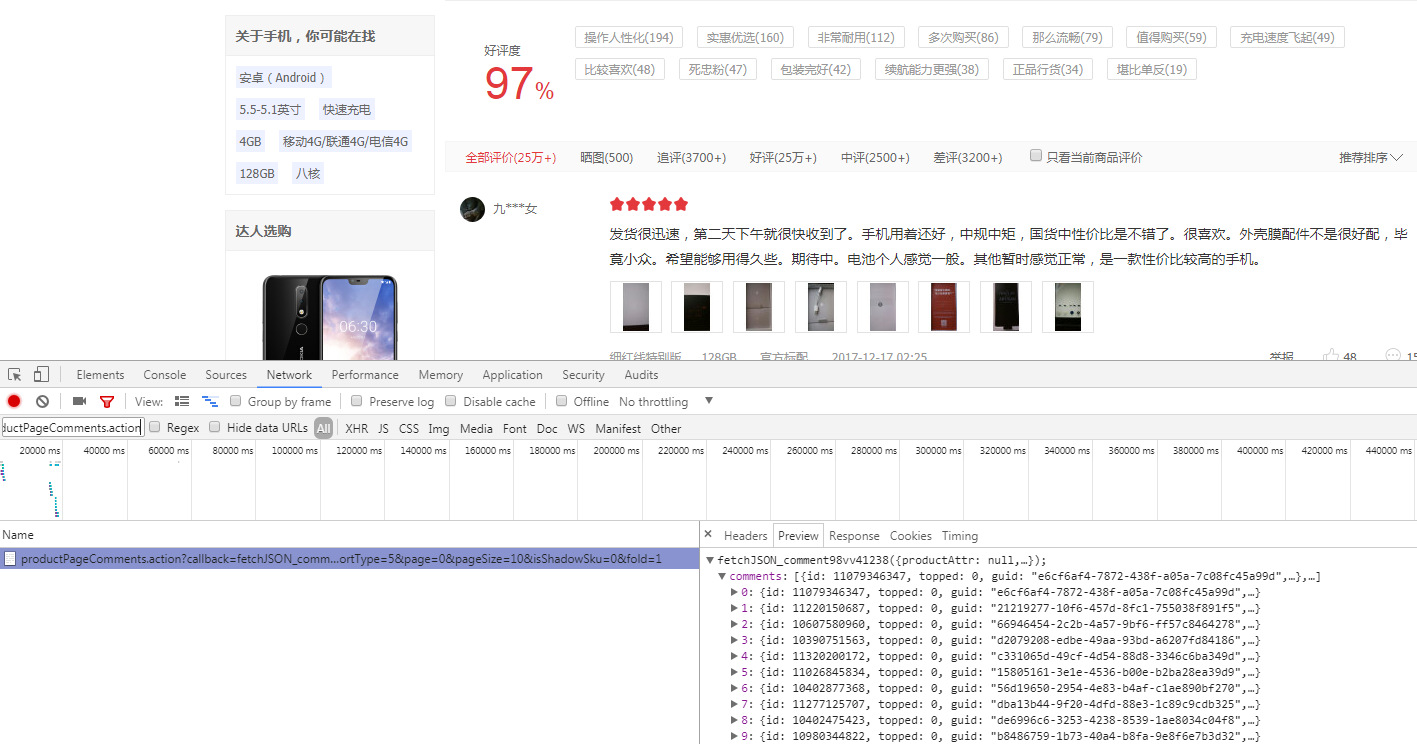
**七、分析电商网站源码结构**

1. 使用Chrome浏览器，打开京东官网，随便选择一款商品https://item.jd.com/4432058.html

按键盘上的F12键，进入开发者工具页面。点击开发者工具里的“NetWork”，再点击商品详情页里的“商品评论”，就可以看到浏览器请求的数据文件。



利用Filter过滤网址，输入productPageComments.action，查找相关的网址请求，此请求就是获取商品评论内容。我们可以看到右侧，显示出了10条评论。



2. 查看productPageComments.action这个链接，后面的参数

完整链接：

https://sclub.jd.com/comment/productPageComments.action?callback=fetchJSON\_comment98vv41238&productId=4432058&score=0&sortType=5&page=0&pageSize=10&isShadowSku=0&fold=1

请求url为：https://sclub.jd.com/comment/productPageComments.action

请求时参数：callback=fetchJSON\_comment98vv41238&productId=4432058&score=0&sortType=5&page=0&pageSize=10&isShadowSku=0&fold=1

callback=fetchJSON\_comment98vv28421

productId=4432058

score=0

sortType=5

page=0

pageSize=10

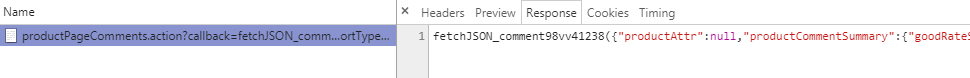
isShadowSku=0

fold=1

上面参数中，最主要的是page页码，pagesize每页多少条评论。通过调整page参数的数值，可以发现，即使当前商品有上万条的评论数据，商品评论信息，最多也只显示100页的数据，也就是page最大为100，评论条数最多为1000。

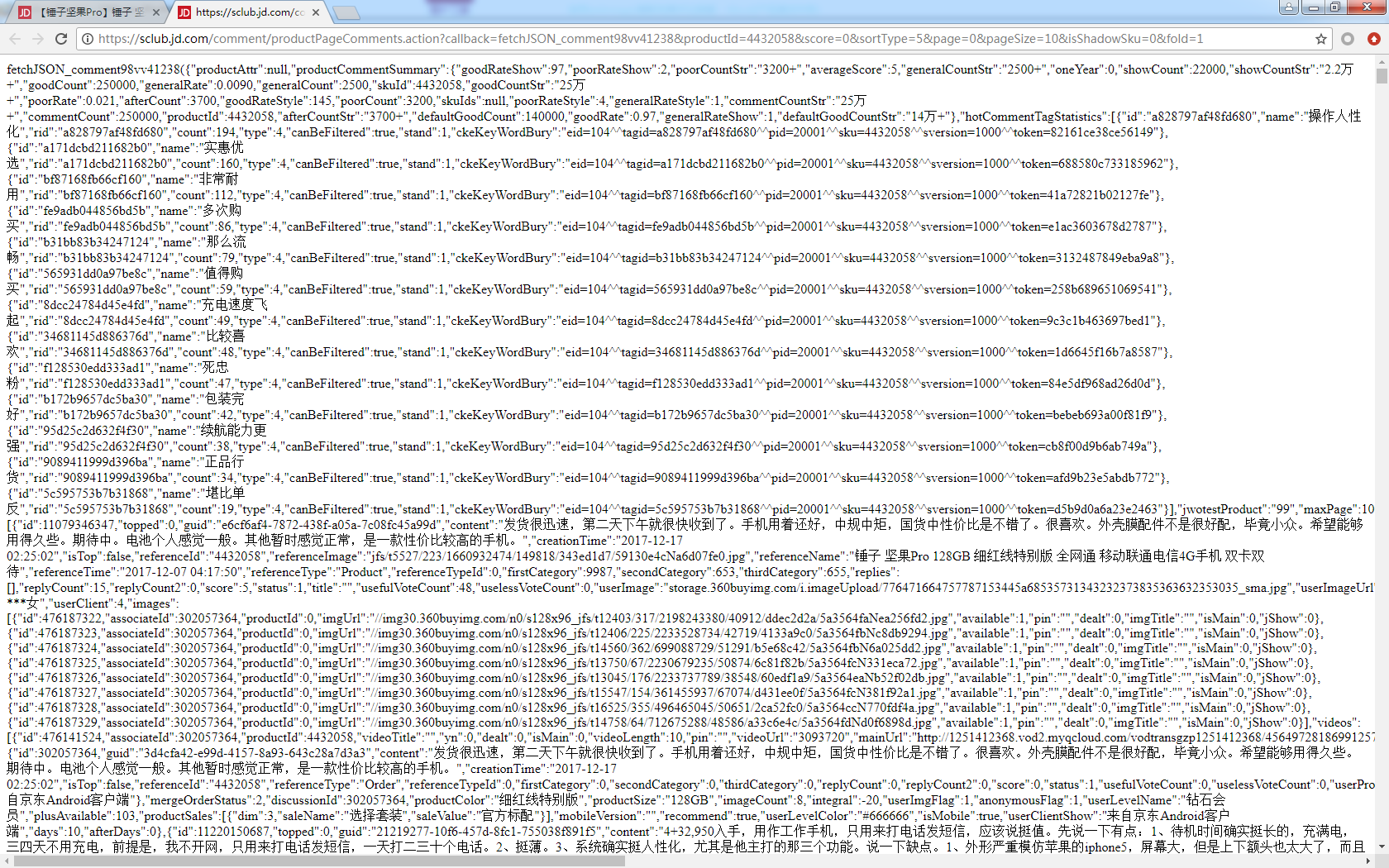
另外，这里有一项参数：

callback=fetchJSON\_comment98vv28421，可以看看到参数后面的数据fetchJSON\_comment98vv28421与http响应Response中的头部一致，所以可以尝试去掉此参数。于是可得返回的Json形式的评论信息。



**八、分析电商评论部分字段含义**

首先看下返回的Json形式的评论，可以看到格式比较混乱，可读性较差



想办法格式化一下，访问代码格式化工具http://tool.oschina.net/codeformat/json。使用此工具，对Json评论进行格式化。

可以看到如下格式比较清晰的效果：



在结构化的Json中，查找有用的信息。

productCommentSummary 存储了对该商品全部评论的汇总信息；

comments 存储了当前页面的10条评论数据

productCommentSummary中详细内容：

"productCommentSummary": {

"goodRateShow": 97, //好评率

"poorRateShow": 2, //差评率

"poorCountStr": "3200+",

"averageScore": 5, //平均分

"generalCountStr": "2500+",

"oneYear": 0,

"showCount": 22000,

"showCountStr": "2.2万+",

"goodCount": 250000, //好评数

"generalRate": 0.009,

"generalCount": 2500, //中评数

"skuId": 4432058,

"goodCountStr": "25万+",

"poorRate": 0.021,

"afterCount": 3700,

"goodRateStyle": 145,

"poorCount": 3200, //差评数

"skuIds": null,

"poorRateStyle": 4,

"generalRateStyle": 1,

"commentCountStr": "25万+",

"commentCount": 250000,

"productId": 4432058, //商品id

"afterCountStr": "3700+",

"defaultGoodCount": 140000,

"goodRate": 0.97,

"generalRateShow": 1, //中评率

"defaultGoodCountStr": "14万+"

},

comments中一条评论的结构：

"id": 11079346347,

"topped": 0,

"guid": "e6cf6af4-7872-438f-a05a-7c08fc45a99d",

"content": "发货很迅速，第二天下午就很快收到了。手机用着还好，中规中矩，国货中性价比是不错了。很喜欢。外壳膜配件不是很好配，毕竟小众。希望能够用得久些。期待中。电池个人感觉一般。其他暂时感觉正常，是一款性价比较高的手机。", //评论内容 √

"creationTime": "2017-12-17 02:25:02", //写评论的时间 √

"isTop": false, //是否置顶

"referenceId": "4432058",

"referenceImage": "jfs/t5527/223/1660932474/149818/343ed1d7/59130e4cNa6d07fe0.jpg",

"referenceName": "锤子 坚果Pro 128GB 细红线特别版 全网通 移动联通电信4G手机 双卡双待",

"referenceTime": "2017-12-07 04:17:50", //收货时间 √

"referenceType": "Product",

"referenceTypeId": 0,

"firstCategory": 9987, //第一分类 √

"secondCategory": 653, //第二分类 √

"thirdCategory": 655, //第三分类 √

"replies": [ ],

"replyCount": 15,

"replyCount2": 0,

"score": 5, //打分 √

"status": 1,

"title": "",

"usefulVoteCount": 48,

"uselessVoteCount": 0,

"userImage": "storage.360buyimg.com/i.imageUpload/776471664757787153445a68535731343232373835363632353035\_sma.jpg",

"userImageUrl": "storage.360buyimg.com/i.imageUpload/776471664757787153445a68535731343232373835363632353035\_sma.jpg",

"userLevelId": "105",

"userProvince": "",

"viewCount": 0,

"orderId": 0,

"isReplyGrade": false,

"nickname": "九\*\*\*女", //昵称  √

"userClient": 4,

"images": [],

"videos": [],

"showOrderComment": {},

"mergeOrderStatus": 2,

"discussionId": 302057364,

"productColor": "细红线特别版",

"productSize": "128GB",

"imageCount": 8,

"integral": -20,

"userImgFlag": 1,

"anonymousFlag": 1,

"userLevelName": "钻石会员", //会员级别 √

"plusAvailable": 103,

"productSales": [],

"mobileVersion": "",

"userLevelColor": "#666666",

"recommend": true,

"userClientShow": "来自京东Android客户端", //评论设备

"isMobile": true, //是否移动端

"days": 10, //评论时间距【收货/下单】时间多长时间

"afterDays": 0

了解了相关字段含义，下面开始使用webmagic，编写爬虫，爬取京东评论。

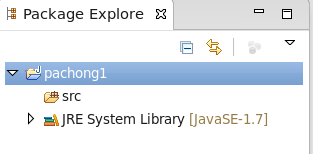
**九、搭建爬虫代码框架**

1. 获取webmagic爬虫项目所依赖的lib包

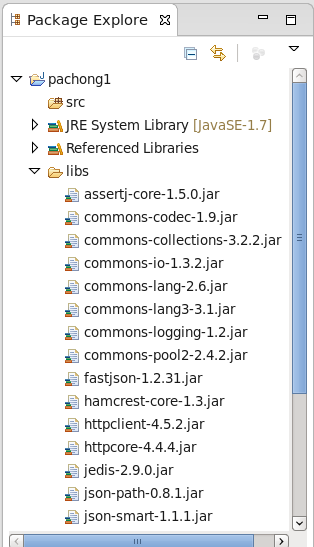
[hfut@master ~]$ cp ~/resources/software/webmagic-0.7-libs.tar.gz ~/

[hfut@master ~]$ tar -zxvf webmagic-0.7-libs.tar.gz

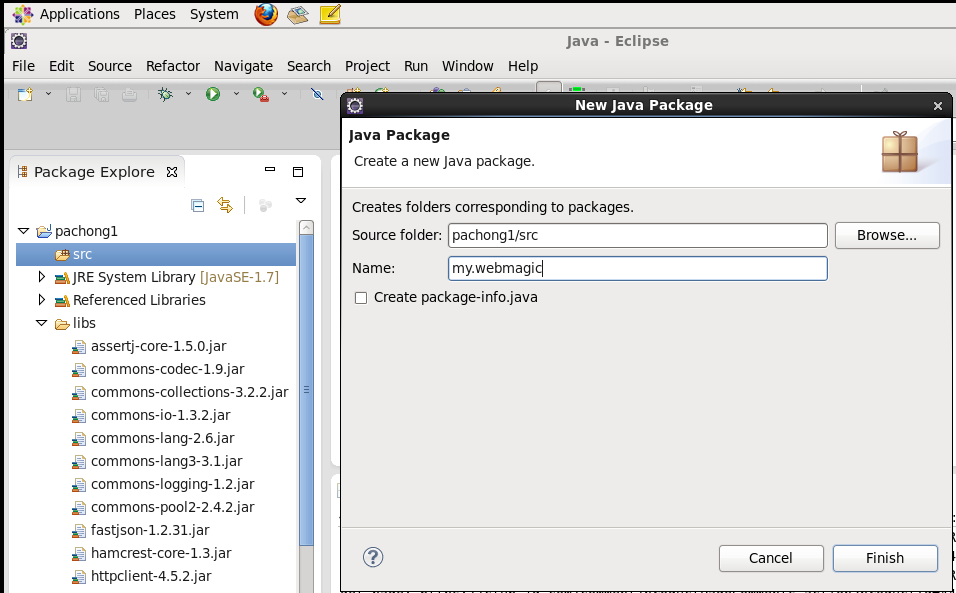
2. 打开eclipse开发工具，新建Java Project，命名为pachong1



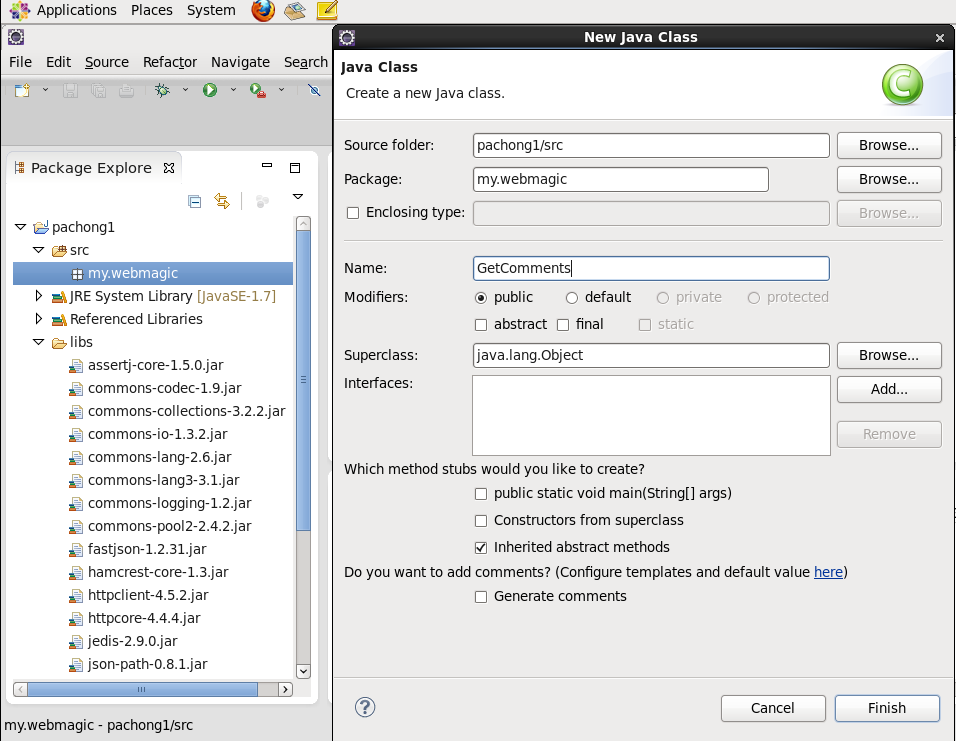
3. 点击项目名pachong1，新建一个目录，命名为libs，并将~/webmagic-0.7-libs下的所有的jar包，拷贝到libs目录下。并选中所有jar包，右键点击“Build Path” => “Add to Build Path”



4. 在pachong1项目下，点击src => New => Package新建一个包my.webmagic



5. 在my.webmagic包下，新建GetComments类



6. 代码编写

在WebMagic里，实现一个基本的爬虫只需要编写一个类，实现 us.codecraft.webmagic.processor.PageProcessor接口即可。这个类基本上包含了抓取一个网站，需要写的所有代码。实现接口后，重写里面的getSite和process方法

import us.codecraft.webmagic.Page;

import us.codecraft.webmagic.Site;

import us.codecraft.webmagic.processor.PageProcessor;

public class GetComments implements PageProcessor {

@Override

public Site getSite() {

return null;

}

@Override

public void process(Page arg0) {

}

}

**十、模拟爬虫为普通用户**

getSite()方法，返回的数据类型为us.codecraft.webmagic.Site类型。也就是对当前爬虫，进行设置，包括编码、抓取间隔、超时时间、重试次数等，也包括一些模拟的参数，例如User Agent、cookie，以及代理的设置。

private Site site = Site.me().setDomain("sclub.jd.com").setSleepTime(2000);

如果想设置更详细内容，可以这样做：

private Site site = Site.me()

.setDomain("sclub.jd.com")

.setSleepTime(2000)

.addCookie("TrackID", "15yCkQ1-EBrUjdqR00DQ5QscyENaUom43JpYSvjMJpq6anA50TrPtspNLl6U8XaLsrVJcwNmEdABTf0HNRnM0w2bDvqpTE7RZbUMrf46d7Y0")

.addCookie("\_\_jda", "122270672.15289023667021425393993.1528902367.1528938475.1528969012.4")

.addCookie("\_\_jdb", "122270672.7.15289023667021425393993|4.1528969012")

.addCookie("\_\_jdc ", "122270672")

.addCookie("\_\_jdu", "15289023667021425393993")

.addCookie("\_\_jdv", "122270672|cps.youmai.com|t\_1000049399\_39857496|tuiguang|aed1c4304f694e599245a430daeddb2f|1528905344545")

.addHeader("User-Agent", "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.3")

.addHeader("Accept", "\*/\*")

.addHeader("Accept-Encoding", "gzip, deflate, br")

.addHeader("Accept-Language", "en-US,en;q=0.8")

.addHeader("Connection", "keep-alive")

.addHeader("Referer", "https://www.jd.com/")

.setUserAgent("Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.101 Safari/537.3");

这里一些项目内容，例如setDomain的值、setUserAgent的值，可以从这里获取



Cookie:

shshshfpa=babfeb75-31d9-07dc-a0f6-be26b618d2aa-1528902363; unpl=V2\_ZzNtbRIDFhUnDURVfk5bDWJXFlVKVEYRJQ1EVS5JCFAyUBEKclRCFXwUR1NnGV4UZAEZWUZcRhxFCEdkexhdBGYGGl9KX3MXfQFCUn4RWzVXABJtSmdDEXYBTlV4G1sCYQIRXUdURxd9Ck5ceSlsAlczQggeDxx7dglGU0sZXQ1gBRZYRlFCJXQ4R2Qwd11IZwcRVEpWQBdyD0BVeBlZBmMBGl9KX0EldDhF; \_\_jdv=122270672|cps.youmai.com|t\_1000049399\_39857496|tuiguang|aed1c4304f694e599245a430daeddb2f|1528905344545; PCSYCityID=1116; TrackID=15yCkQ1-EBrUjdqR00DQ5QscyENaUom43JpYSvjMJpq6anA50TrPtspNLl6U8XaLsrVJcwNmEdABTf0HNRnM0w2bDvqpTE7RZbUMrf46d7Y0; pinId=Qt4n3Dx5KWPR40iONEjiKA; \_tp=IhBEuolsSUuizEP%2BkIxQlw%3D%3D; ceshi3.com=000; wlfstk\_smdl=du1p9xyo4r42zng2rhmipevhhlhnb9bo; 3AB9D23F7A4B3C9B=4HEWSNHHWIGGCQE7PEH6JLUXCALOGGGAT6N2EAQVG2SKCJIFW4R63EHVR5GFOR53GZOCQQBQCCBZP7WZC2OE3MPCQI; shshshfp=84790e35436f53e25f89e95a63c3e9e9; shshshfpb=09fec561a60942ea443b56478d0c84fa1bc688805b83efe5c5b2132e43; shshshsID=156711d3a1ebc4856080373b008c34b1\_6\_1528969272965; \_\_jda=122270672.15289023667021425393993.1528902367.1528938475.1528969012.4; \_\_jdb=122270672.7.15289023667021425393993|4.1528969012; \_\_jdc=122270672; \_\_jdu=15289023667021425393993

这里的setDomain是请求的Host地址

最终，将site对象，在getSite方法作为结果返回。

[](http://192.168.1.241/doc/exper/125/img/24.png)

**十一、处理页面内容**

process方法，是webmagic爬虫的核心部分，可以从下载到的Html页面，抽取到想要的信息。WebMagic里主要使用了三种抽取技术：XPath、正则表达式和CSS选择器。此处使用page.getJson()，得到当前页码Json形式的全部评论，并且存储到ten这个变量中。

public void process(Page page) {

page.putField("ten", page.getJson().toString());

}

**十二、设置Scheduler**

Scheduler是WebMagic中进行URL管理的组件。一般来说，Scheduler包括两个作用：①对待抓取的URL队列进行管理； ②对已抓取的URL进行去重。

WebMagic内置了几个常用的Scheduler：

[](http://192.168.1.241/doc/exper/125/img/18.png)

通过观察url可以发现规律，page表示页码，并从0开始依次增加，便可以抓取到每个页面的10条评论

此处我们使用QueueScheduler，使用for循环，得到100个url，并将待抓取的url放到内存队列中存储。

创建main函数，并在main函数中编写代码：

String url\_init = "https://sclub.jd.com/comment/productPageComments.action?" + "productId=4432058&score=0&sortType=5&page=0&pageSize=10&isShadowSku=0&fold=1";

String url\_pattern = "https://sclub.jd.com/comment/productPageComments.action?" + "productId=4432058&score=0&sortType=5&pageSize=10&isShadowSku=0&fold=1&page=";

String output = "/home/hfut/tmp/";

QueueScheduler scheduler = new QueueScheduler();

Spider spider = Spider.create(new GetComments())

.addUrl(url\_init)

.setScheduler(scheduler)

.addPipeline(new JsonFilePipeline(output))

.addPipeline(new ConsolePipeline());

for (int i = 0; i < 100; i++) {

Request request = new Request();

request.setUrl(url\_pattern + i);

scheduler.push(request, spider);

}

spider.thread(50).run();

在main函数中，Spider是爬虫启动的入口。在启动爬虫之前，我们需要使用一个PageProcessor创建一个Spider对象，然后使用run()进行启动。同时Spider的其他组件（Downloader、Scheduler、Pipeline）都可以通过set方法来进行设置。

.addUrl(url\_init)

.setScheduler(scheduler)

.addPipeline(new JsonFilePipeline(output))

.addPipeline(new ConsolePipeline());

在此段代码中

.addUrl 爬虫初始化时，需要爬取的页面

.setScheduler 设置爬虫计划，将待爬取的url告诉爬虫

.addPipeline(new JsonFilePipeline(output)) 设置爬取结果存储形式和位置，这里以Json形式存储

.addPipeline(new ConsolePipeline()) 设置爬取结果存储形式和位置，这里将结果同时输出到console页面。

**十三、执行测试**

完整代码：

**package** my.webmagic;

**import** us.codecraft.webmagic.Page;

**import** us.codecraft.webmagic.Request;

**import** us.codecraft.webmagic.Site;

**import** us.codecraft.webmagic.Spider;

**import** us.codecraft.webmagic.pipeline.ConsolePipeline;

**import** us.codecraft.webmagic.pipeline.JsonFilePipeline;

**import** us.codecraft.webmagic.processor.PageProcessor;

**import** us.codecraft.webmagic.scheduler.QueueScheduler;

**public** **class** GetComments **implements** PageProcessor {

// 对爬取站点的一些属性进行设置，例如：设置域名，设置代理等；

**private** Site site = Site.*me*()

.setDomain("sclub.jd.com")

.setSleepTime(2000)

.addHeader("Referer", "https://item.jd.com/100006947212.html/");

**public** Site getSite() {

**return** site;

}

**public** **void** process(Page page) {

page.putField("ten", page.getJson().toString());

}

**public** **static** **void** main(String[] args) {

String url\_init = "https://club.jd.com/comment/productPageComments.action?"

+ "productId=100006947212&score=0&sortType=5&page=0&pageSize=10&isShadowSku=0&fold=1";

String url\_pattern = "https://sclub.jd.com/comment/productPageComments.action?"

+ "productId=100006947212&score=0&sortType=5&pageSize=10&isShadowSku=0&fold=1&page=";

String output = "/home/hfut/tmp/";

QueueScheduler scheduler = **new** QueueScheduler();

Spider spider = Spider.*create*(**new** GetComments()).addUrl(url\_init)

.setScheduler(scheduler)

.addPipeline(**new** JsonFilePipeline(output))

.addPipeline(**new** ConsolePipeline());

**for** (**int** i = 1; i < 100; i++) {

Request request = **new** Request();

request.setUrl(url\_pattern + i);

scheduler.push(request, spider);

}

spider.thread(50).run();

}

}

在项目上，右键，点击Run As => Java Application。

运行结束后查看/home/hfut/tmp/sclub.jd.com目录：

