**19. 数据清洗-清洗电商评论数据**

**一、数据清洗概述**

数据清洗是对数据进行重新审查和校验的过程，目的在于删除重复信息、纠正存在的错误，并提供数据一致性。

数据清洗从名字上也看的出就是把“脏”的“洗掉”，指发现并纠正数据文件中可识别的错误的最后一道程序，包括检查数据一致性，处理无效值和缺失值等。因为数据仓库中的数据是面向某一主题的数据的集合，这些数据从多个业务系统中抽取而来而且包含历史数据，这样就避免不了有的数据是错误数据、有的数据相互之间有冲突，这些错误的或有冲突的数据显然是我们不想要的，称为“脏数据”。我们要按照一定的规则把“脏数据”“洗掉”，这就是数据清洗。而数据清洗的任务是过滤那些不符合要求的数据，将过滤的结果交给业务主管部门，确认是否过滤掉还是由业务单位修正之后再进行抽取。不符合要求的数据主要是有不完整的数据、错误的数据、重复的数据三大类。数据清洗是与问卷审核不同，录入后的数据清洗一般是由计算机而不是人工完成。

[](http://192.168.1.241/doc/exper/126/img/01.png)

**二、解析Json**

通过爬虫，我们可以得到一些Json形式的数据文件。

[](http://192.168.1.241/doc/exper/126/img/02.png)

但是Json文件，内容格式比较混乱。所以还需要进一步清洗，提取出有用的信息。

[](http://192.168.1.241/doc/exper/126/img/03.png)

上一节，演示了使用代码格式化工具http://tool.oschina.net/codeformat/json，对Json评论进行格式化的过程。

格式化后的代码，显得结构更加明显清晰

例如，comments中详细内容：

id": 10432588299,

guid": "6c1d83b1-ac45-4189-a041-774eaff87df9",

content": "割手，相当的割手，无语了", //评论内容 √

creationTime": "2017-05-22 23:37:24", //写评论的时间 √

isTop": false, //是否置顶

referenceTime": "2017-05-20 18:35:11", //收货时间 √

firstCategory": 9987, //第一分类 √

secondCategory": 653, //第二分类 √

thirdCategory": 655, //第三分类 √

replyCount": 0,

score": 3, //打分 √

nickname": "j\*\*\*柜", //昵称 √

userClient": 2,

productColor": "碳黑色",

productSize": "32GB",

userLevelName": "金牌会员", //会员级别 √

plusAvailable": 0,

productSales": [

{

"dim": 3,

"saleName": "选择套装",

"saleValue": "官方标配"

}

,

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

isMobile": true, //是否移动端

days": 2, //评论时间距【收货/下单】时间多长时间

afterDays": 0

在这段Json中，对我们有用的是，如下一些字段

"id": 10432588299,

"guid": "6c1d83b1-ac45-4189-a041-774eaff87df9",

"content": "割手，相当的割手，无语了", //评论内容 √

"creationTime": "2017-05-22 23:37:24", //写评论的时间 √

"isTop": false, //是否置顶

"referenceTime": "2017-05-20 18:35:11", //收货时间 √

"score": 3, //打分 √

"nickname": "j\*\*\*柜", //昵称 √

"userLevelName": "金牌会员", //会员级别 √

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

"isMobile": true, //是否移动端

"days": 2, //评论时间距【收货/下单】时间多长时间

当然，判断字段是否对我们有用，判断依据是根据需求来定的。后续做的一些需求，会用到哪些字段，此处就会采集哪些字段。

这一节我们会使用MapReduce，对大量的Json文件，进行清洗，以得到结构化的文本文件。

**三、搭建解析框架**

1. 准备hadoop2lib.tar.gz和fastjson-1.2.31.jar软件包

[hfut@master ~]$ mkdir ~/qingxi-data

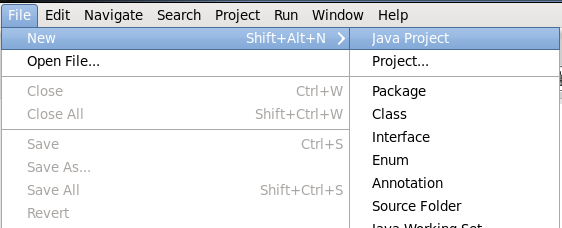
[hfut@master ~]$ cp ~/resources/software/fastjson-1.2.31.jar ~/qingxi-data/

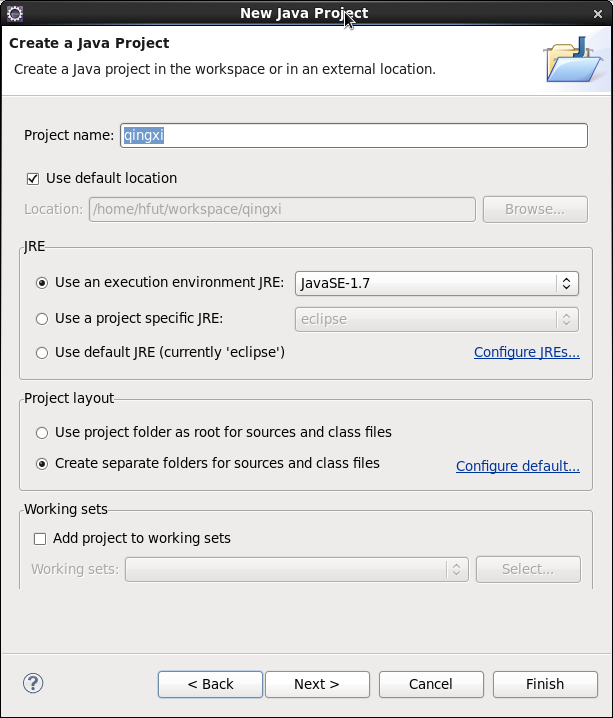
[hfut@master ~]$ cp ~/resources/software/hadoop2lib.tar.gz ~/qingxi-data/

[hfut@master ~]$ cd qingxi-data/

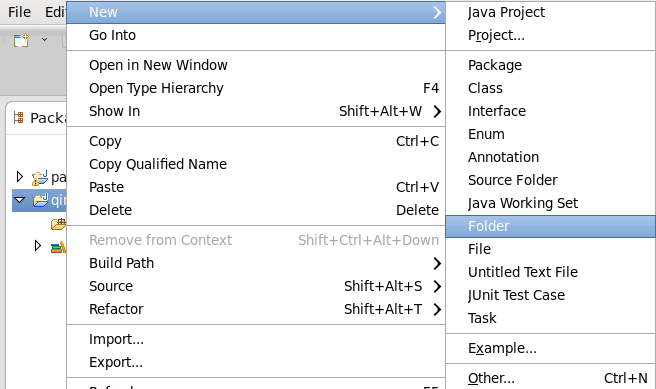
[hfut@master qingxi-data]$ tar -zxvf hadoop2lib.tar.gz

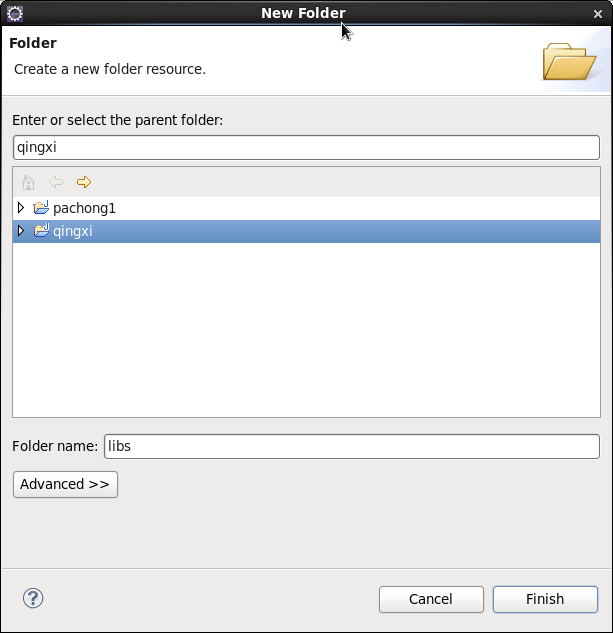
2. 新建Java Project，将项目命名为qingxi





3. 右键项目名，新建一个目录，命名为libs用于存储项目依赖的jar包

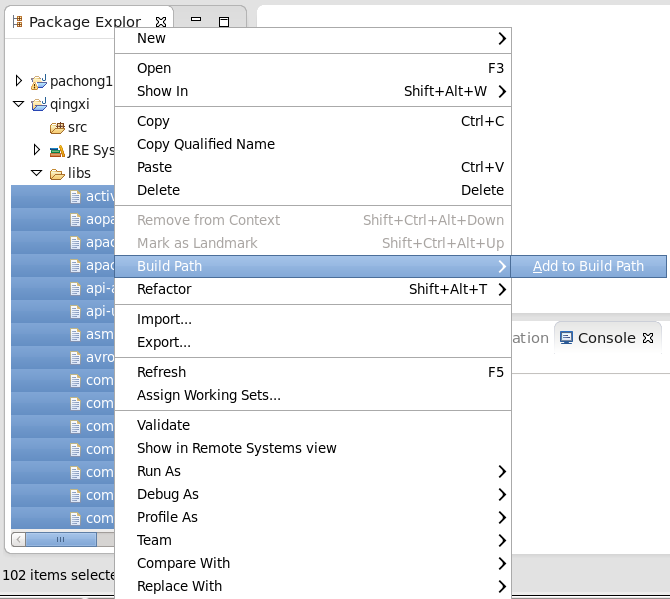




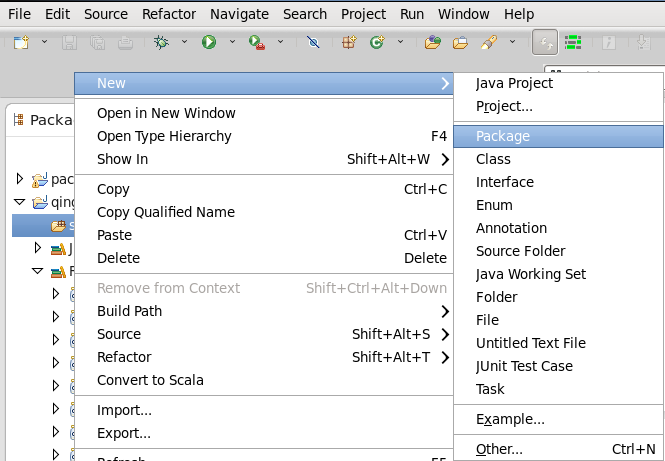
将/qingxi-data/hadoop2lib目录下，所有的jar包，拷贝到项目下的libs目录下。

将/qingxi-data/目录下，fastjson-1.2.31.jar，拷贝到项目下的libs目录下。

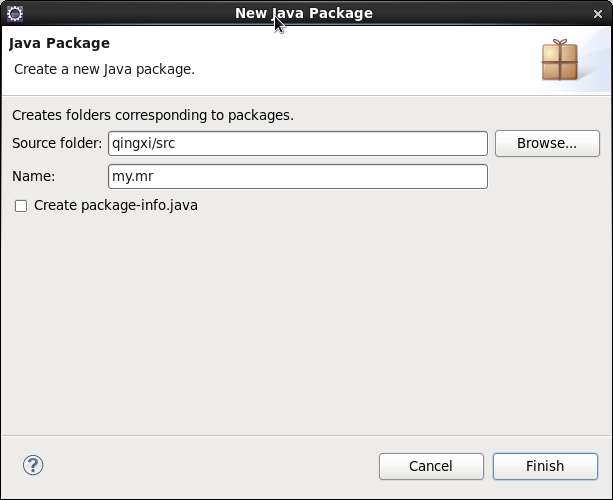
选中libs下，所有的jar文件，依次点击“Build Path” => "Add to Build Path"



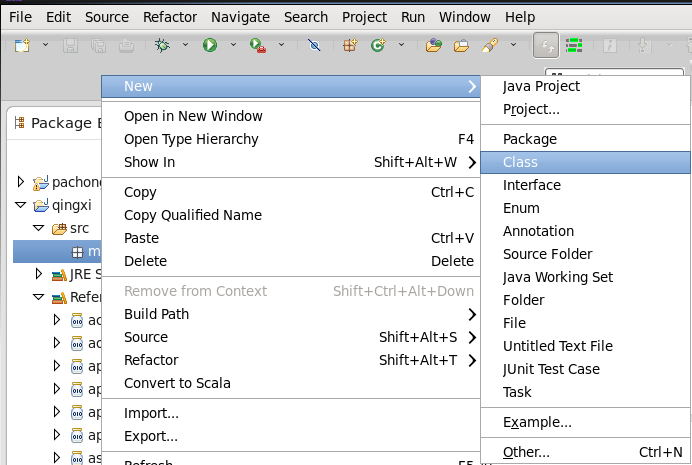
4. 右键src，点击 "New"-> "Package"，新建一个包



将包命名为my.mr

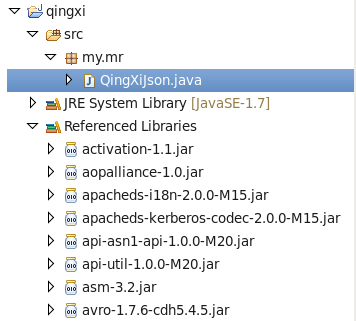


右键包名，依次点击“New”->“Class”



并将类命名为QingXiJson

这样清洗过程的框架搭建完毕，下面开始编写代码实现功能。



**四、编写MapReduce代码**

1. 启动hadoop

2. 准备数据，将爬虫爬取的数据上传到hdfs的/qingxi/in目录中

[hfut@master ~]$ hadoop fs -mkdir -p /qingxi/in

[hfut@master ~]$ hadoop fs -put ~/tmp/sclub.jd.com/\* /qingxi/in

3. 代码所实现的需求，是使用MapReduce解析Json文件，最终输出格式化的文本文件。

首先来看MapReduce通用的框架结构样式。

public class QingXiJson {

public static void main(String[] args) throws IOException,

ClassNotFoundException, InterruptedException {

}

public static class doMapper extends Mapper<Object, Text, Text, Text> {

@Override

protected void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

}

}

public static class doReducer extends Reducer<Text, Text, Text, Text>{

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

}

}

}

通过分析可以知道，此处只用Map任务即可实现具体功能，所以可以省去Reduce任务。

4. Main主函数。这里的main函数也是通用的结构

public static void main(String[] args) throws IOException,

ClassNotFoundException, InterruptedException {

Job job = Job.getInstance();

job.setJobName("QingXiJson");

job.setJarByClass(QingXiJson.class);

job.setMapperClass(doMapper.class);

//job.setReducerClass(doReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

Path in = new Path("hdfs://master:9000/qingxi/in");

Path out = new Path("hdfs://master:9000/qingxi/out/1");

FileInputFormat.addInputPath(job, in);

FileOutputFormat.setOutputPath(job, out);

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

①定义Job

②设置Job参数

③设置Map任务

④设置Reduce任务

⑤定义任务的输出类型

⑥设置任务的输入输出目录

⑦提交执行

5. 再来看Map任务，实现Map任务，必须继承org.apache.hadoop.mapreduce.Mapper类，并重写类里的map方法。

每个json文件，包含一条Json文本数据。通过map任务，取得文件里的数据，并通过fastjson类，对json文件进行解析，获取json中的字段。

最终使用StringBuilder类，将相关字段以‘\t’分隔拼接成一行，进行输出。

public static class doMapper extends Mapper<Object, Text, Text, Text> {

@Override

protected void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

String initJsonString = value.toString();

JSONObject initJson = JSONObject.parseObject(initJsonString );

if (!initJsonString.contains("productCommentSummary") && !initJsonString.contains("comments")) {

return;

}

JSONObject myjson = initJson.getJSONObject("ten");

JSONObject productCommentSummary = myjson.getJSONObject("productCommentSummary");

String productId = productCommentSummary.get("productId").toString();

String commentCount = productCommentSummary.get("commentCount").toString();

String goodCount = productCommentSummary.get("goodCount").toString();

String generalCount = productCommentSummary.get("generalCount").toString();

String poorCount = productCommentSummary.get("poorCount").toString();

String goodRateShow = productCommentSummary.get("goodRateShow").toString();

String generalRateShow = productCommentSummary.get("generalRateShow").toString();

String poorRateShow = productCommentSummary.get("poorRateShow").toString();

/\* comments 包括十条评论 \*/

JSONArray comments = myjson.getJSONArray("comments");

for (int i = 0; i < comments.size(); i++) {

JSONObject comment = comments.getJSONObject(i);

String guid = comment.getString("guid");

String content = comment.getString("content").replace('\n', ' ');

String creationTime = comment.getString("creationTime");

String score = comment.getString("score");

String nickname = comment.getString("nickname");

String userLevelName = comment.getString("userLevelName");

String userClientShow = comment.getString("userClientShow");

String isMobile = comment.getString("isMobile");

String days = comment.getString("days");

StringBuilder sb = new StringBuilder();

sb.append(productId); sb.append("\t");

sb.append(commentCount); sb.append("\t");

sb.append(goodCount); sb.append("\t");

sb.append(generalCount); sb.append("\t");

sb.append( poorCount ); sb.append("\t");

sb.append( goodRateShow ); sb.append("\t");

sb.append( generalRateShow ); sb.append("\t");

sb.append( poorRateShow ); sb.append("\t");

sb.append( guid ); sb.append("\t");

sb.append( content ); sb.append("\t");

sb.append( creationTime ); sb.append("\t");

sb.append( score ); sb.append("\t");

sb.append( nickname ); sb.append("\t");

sb.append( userLevelName ); sb.append("\t");

sb.append( userClientShow ); sb.append("\t");

sb.append( isMobile ); sb.append("\t");

sb.append( days );

String result = sb.toString();

context.write(new Text(result), new Text(""));

}

}

}

6. 此外实验代码需要导入的依赖包如下：

import java.io.IOException;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

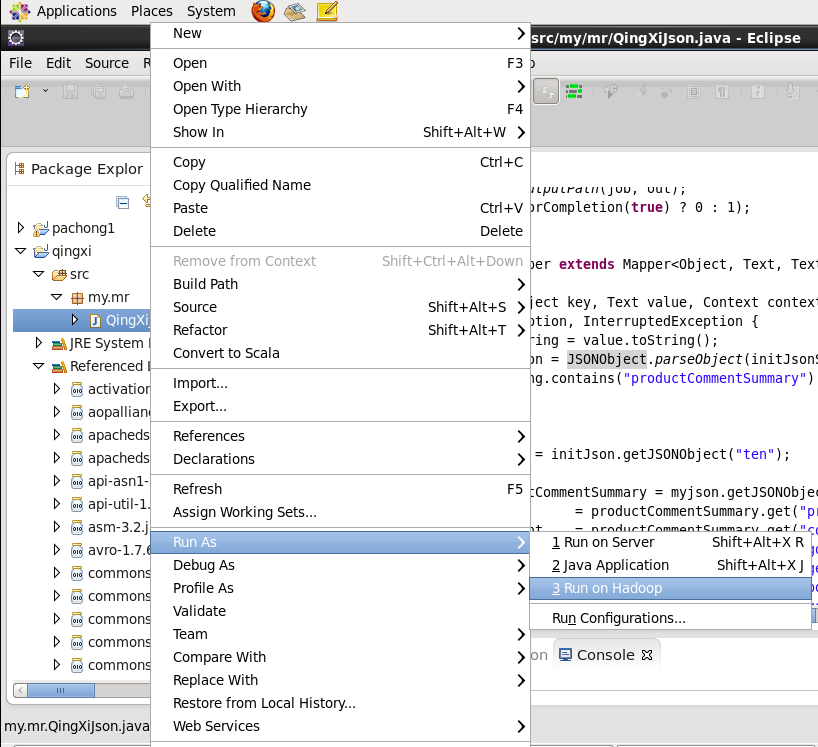
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

**五、执行测试**

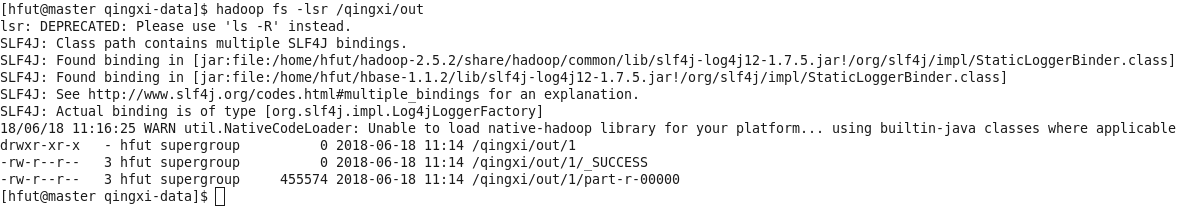
1. 在mapreduce类中，右键，Run As => Run on Hadoop，将任务提交到hadoop中执行



2. 等待任务执行完毕。切换目录到~/qingxi-data/下，并在命令行界面，输入脚本，查看hdfs上/qingxi/out是否有内容输出

[hfut@master ~]$ cd ~/qingxi-data/

[hfut@master qingxi-data]$ hadoop fs -lsr /qingxi/out

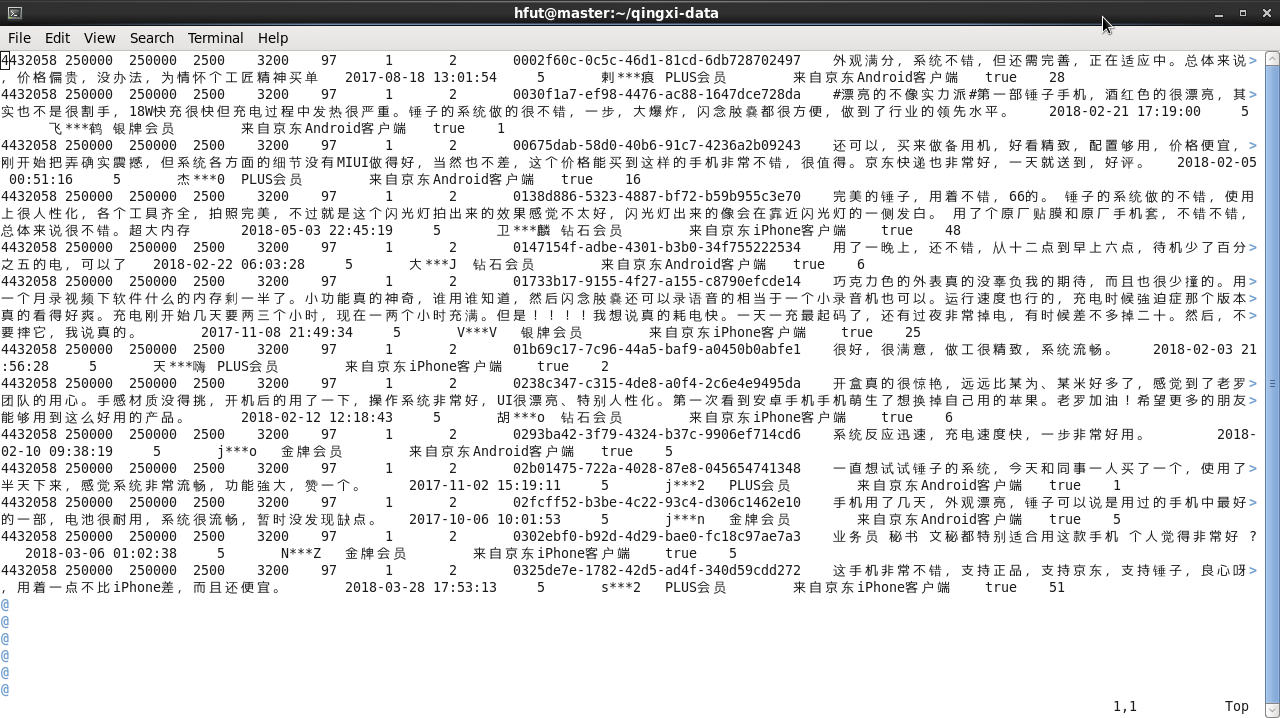


若有输出，则将hdfs输出内容，下载到linux本地

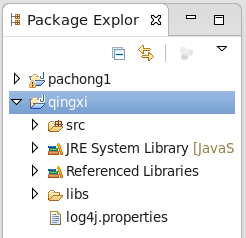
[hfut@master qingxi-data]$ hadoop fs -get /qingxi/out/1/\*

使用vim或cat查看下载到的文件内容，可以看到结构比较清晰

[hfut@master qingxi-data]$ vi part-r-00000



3. 若未在hdfs上，查看到输出结果，可以通过log日志排错。将~/hadoop-2.5.2/etc/hadoop/log4j.properties文件，拷贝到qingxi项目的src目录下。



可以看到在eclipse的console界面有执行过程的输出。

