

合肥工业大学

专业课程

(计算机与信息学院)

大数据处理技术实验报告

专 业 班 级	计算机科学与技术 16-3 班
学 生 姓 名 及 学 号	任恒 2016212063
课 程 教 学 班 号	001 班
任 课 教 师	吴共庆
实 验 指 导 教 师	吴共庆
实 验 地 点	科教楼 D502

2018~2019 学年第 二 学期

说 明

实验报告是关于实验教学内容、过程及效果的记录和总结，因此，应注意以下事项和要求：

1. 每个实验单元在 50 页的篇幅内完成一份报告。“实验单元”指按照实验指导书规定的实验内容。若篇幅不够，可另附纸。

2、各实验的预习部分的内容是进入实验室做实验的必要条件，请按要求做好预习。

3. 实验报告要求：书写工整规范，语言表达清楚，数据和程序真实。理论联系实际，认真分析实验中出现问题与现象，总结经验。

4. 参加实验的每位同学应独立完成实验报告的撰写，其中程序或相关的设计图纸也可以采用打印等方式粘贴到报告中。严禁抄袭或拷贝，否则，一经查实，按作弊论取，并取消理论课考试资格。

5. 实验报告作为评定实验成绩的依据。

目录

实验一 Hadoop 平台上部署 WordCount 程序.....4

一、 实验目的和要求.....4

二、 实验任务4

三、 实验准备方案4

四、 实验用仪器、设备4

五、 实验内容与步骤（过程及数据记录）4

六、 感想、体会、建议9

七、 实验成绩9

实验二 统计某电商网站买家收藏商品数量10

一、 实验目的和要求.....10

二、 实验任务10

三、 实验准备方案11

四、 实验用仪器、设备11

五、 实验内容与步骤（过程及数据记录）：11

六、 感想、体会、建议14

七、 实验成绩15

实验三 评论数据采集处理与可视化.....16

一、 实验目的和要求.....16

二、 实验任务16

三、 实验准备方案16

四、 实验用仪器、设备16

五、 实验内容与步骤（过程及数据记录）16

1. 数据采集16

2. 数据清洗搭建解析框架20

3. 分词项目搭建24

4. 可视化26

六、 感想、体会、建议38

七、 实验成绩38

实验序号及名称：实验一 Hadoop 平台上部署 WordCount 程序

实验时间： 2019 年 4 月 2 日

一、 实验目的和要求

1. 学会使用基本的命令行编译运行 mapreduce 程序。
2. 对 mapreduce 代码编写有基本的熟悉，对 mapreduce 执行过程有更深层次的了解。
3. 在实验过程中学会自己查找解决问题的方法，培养处理独自处理问题的能力。

二、 实验任务

1. 找一个文本文件
2. 使用 mapreduce 统计每一个词出现的次数

三、 实验准备方案

包括以下内容：（硬件类实验：实验原理、实验线路、设计方案等）

（软件类实验：所采用的系统、组件、工具、核心方法、框架或流程图、程序清单等）

实验采用的系统：centos 6.5、hadoop2.5.2

工具：VMware WorkStation

四、 实验用仪器、设备

实验仪器：计算机一台

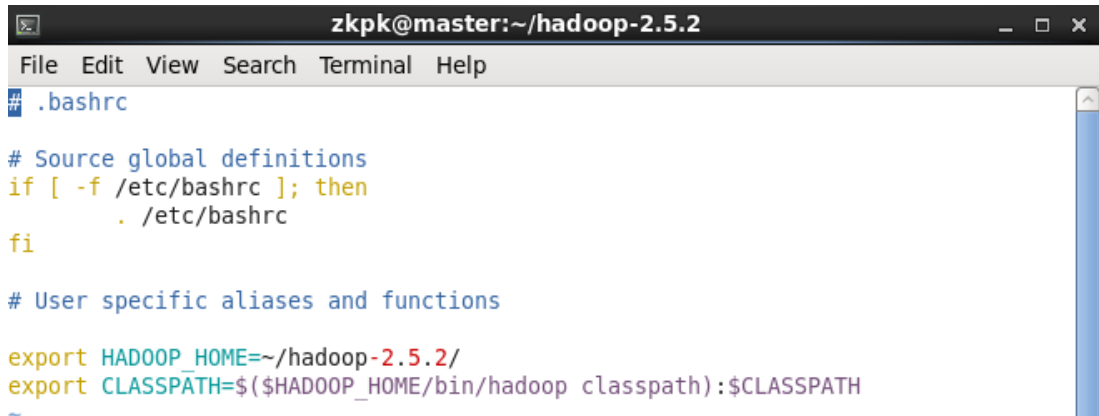
五、 实验内容与步骤（过程及数据记录）

1. 我们将 Hadoop 的 classpath 信息添加到 CLASSPATH 变量中，在 ~/.bashrc 中添加最后两行命令。

```
>>vi ~/.bashrc
```

编辑输入：

```
export HADOOP_HOME=~/.hadoop-2.5.2/  
export CLASSPATH=$(HADOOP_HOME/bin/hadoop classpath):$CLASSPATH
```



```
zkpk@master:~/hadoop-2.5.2
File Edit View Search Terminal Help
#.bashrc

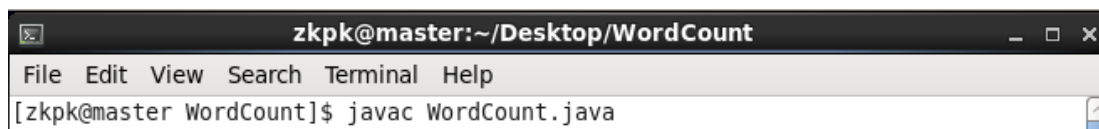
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions

export HADOOP_HOME~/hadoop-2.5.2/
export CLASSPATH=(${HADOOP_HOME/bin/hadoop classpath}):$CLASSPATH
~
```

图 1 配置 CLASSPATH 变量

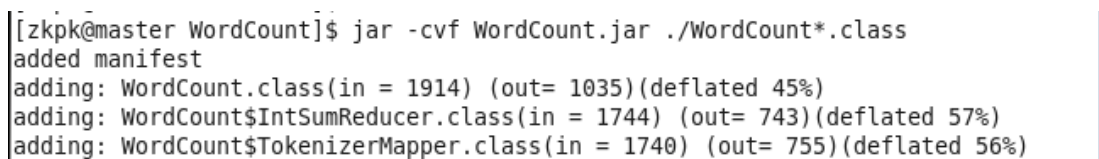
2. 编译 java 文件



```
zkpk@master:~/Desktop/WordCount
File Edit View Search Terminal Help
[zkpk@master WordCount]$ javac WordCount.java
```

图 2 编译 java 文件

3. 将 class 文件打包成 jar 包



```
[zkpk@master WordCount]$ jar -cvf WordCount.jar ./WordCount*.class
added manifest
adding: WordCount.class(in = 1914) (out= 1035)(deflated 45%)
adding: WordCount$IntSumReducer.class(in = 1744) (out= 743)(deflated 57%)
adding: WordCount$TokenizerMapper.class(in = 1740) (out= 755)(deflated 56%)
```

图 3 将 class 文件打包

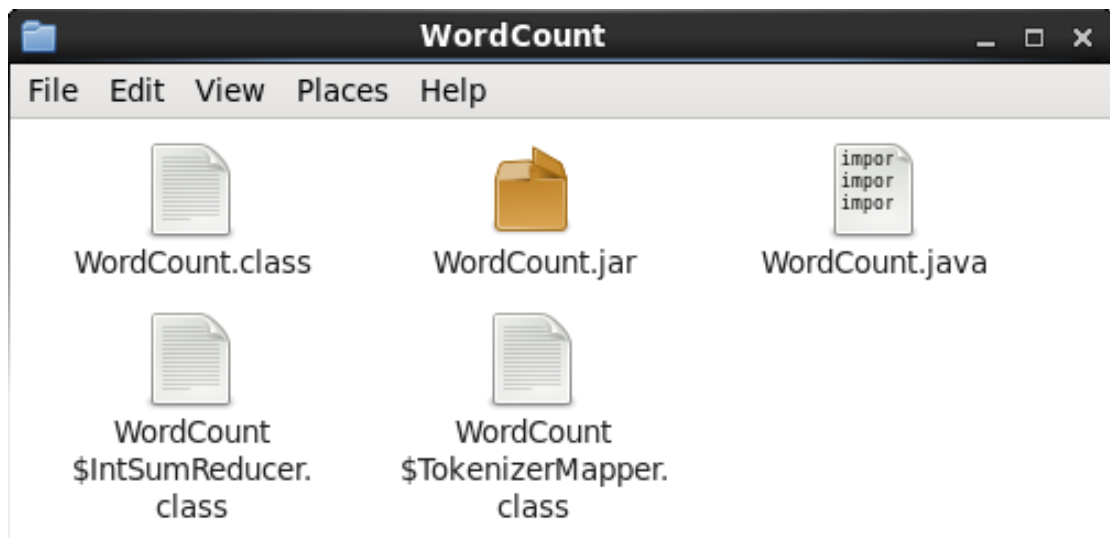
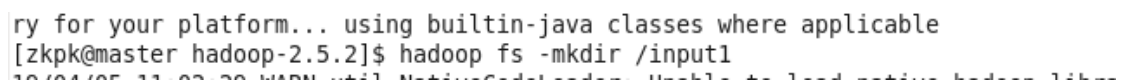


图 4 生成的 jar 包

4. 新建 input1 文件夹:



```
ry for your platform... using builtin-java classes where applicable
[zkpk@master hadoop-2.5.2]$ hadoop fs -mkdir /input1
10/04/05 11:00:00 WARN util.NativeCodeLoader: Unable to load native hadoop library
```

图 5 新建文件夹存储数据

5. 上传待统计文本到输入文件夹

```
adding: WordCount$TokenizerMapper.class(in = 1740) (out= 755)(deflated 56%)  
[zkpk@master WordCount]$ hadoop fs -put zgx /input1
```

图 6 上传文件到 HDFS

6. 运行 WordCount.java 编译生成的 jar 包。

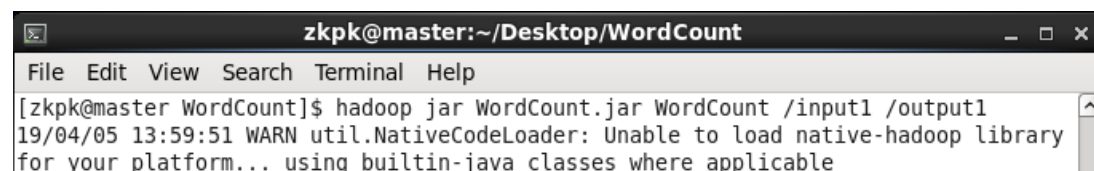


图 7 使用 Hadoop 运行 jar 包

7. 查看统计结果（此处为了方便直接在 eclipse 中访问 HDFS）

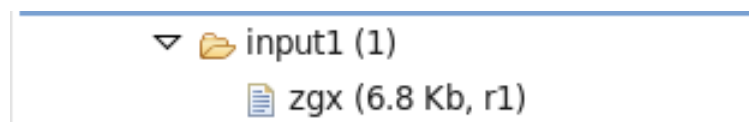


图 7 待统计文件夹

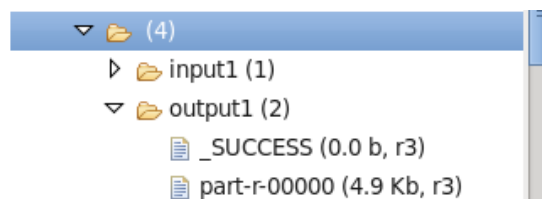


图 8 生成结果文件

8. 查看 WordCount 统计结果

```
72 Jack? 1  
73 John 4  
74 John." 1  
75 John?" 1  
76 Marion 4  
77 Marion!" 1  
78 Marion. 3  
79 Martin 1  
80 Massah 2  
81 Now 1  
82 Old 2  
83 Ole 1  
84 One 1  
85 Ruthven 3  
86 Ruthven, 2  
87 Ruthven. 1  
88 Ruthven?" 1  
89 Ruthvens." 1  
90 She 1  
91 St. 4  
92 Tell 1  
93 That 1  
94 The 5  
95 Want 1
```

图 9 词频统计结果。

```
1 "He said I didn't belong to the Ruthven family?" said Jack slowly, when he felt able  
2  
3 "He did, and I told him I didn't believe him."  
4  
5 "But-but-I don't understand you, Darcy. Am I not Jack Ruthven, the son of the late C  
6  
7 "He says not."  
8  
9 "What! Does he mean to say that my mother isn't my mother at all?" ejaculated Jack, v  
10  
11 "That's it exactly, and he added that Marion wasn't your sister."  
12  
13 "I'll-I'll punch his head for that!" was the quick return.  
14  
15 "I felt like doing that, too, Jack, even though he is so much older than either of u  
16
```

图 10 待统计原文

在图 10 中是随意找的一篇英文小说，可以看到，在图 9 中由于使用的是空格作为分隔符，因此有些单词是连着标点符号的，这显然不是我们想要的，因此对其进一步进行处理。处理之后的结果如图 11 所示。可以看到多余的标点符号等都已经得到了有效的处理，得到的词频统计结果更加的准确，没有导致因为标点符号的不同导致算作两个词的情况发生。

```
1 A 2  
2 Although 1  
3 Am 1  
4 And 1  
5 Andand 1  
6 As 2  
7 Atlantic 1  
8 Ben 2  
9 Bens 2  
10 Big 1  
11 Bill 1  
12 But 7  
13 ButbutI 1  
14 Colonel 1  
15 Come 1  
16 Could 1  
17 Darcy 1  
18 Darcythat 1
```

图 11 进一步处理的统计结果

9. 部分代码解释

Map 函数:

```
private static final IntWritable one = new IntWritable(1);

private Text word = new Text();

public TokenizerMapper() {}

public void map(Object key, Text value, Mapper<Object, Text, Text,
IntWritable>.Context context) throws IOException, InterruptedException {
    /* 设置输入 key、value 对以及输出 key、value 对的格式 */

    StringTokenizer itr = new StringTokenizer(value.toString());

    /* 以空格为标识符进行分词 */

    while(itr.hasMoreTokens()){

        String str2=itr.nextToken().replaceAll("\\pP", "");

        /* 对标点符号进行处理，使用正则去掉标点符号 */

        this.word.set(str2);

        context.write(this.word, one);

    /* 发送 key-Value 对 */

    }
}
```

Redeuce 函数:

```
private IntWritable result = new IntWritable();

public IntSumReducer() {}

public void reduce(Text key, Iterable<IntWritable> values,
    Reducer<Text, IntWritable, Text, IntWritable>.Context context)
    throws IOException,
    InterruptedException {

    int sum = 0;

    IntWritable val;

    for(Iterator itr = values.iterator(); itr.hasNext(); sum += val.get()){

        val = (IntWritable)itr.next();

    }

    /* 对相同 key 的 value 列表进行出现次数的相加 */
}
```



```
        this.result.set(sum);  
        context.write(key, this.result);  
    /* 发送 key、value 对, 分别为单词以及出现的次数 */
```

六、感想、体会、建议

通过本次实验，我对 mapreduce 代码的编写有了基本的掌握，虽然在上学期已经在云计算课程中编写过相关的代码，现在都相当于是重新熟悉一下 mapreduce 代码编写的过程。此外，通过 mapreduce 代码的编写可以在一定程度上加深对 map 以及 reduce 过程的了解与掌握，通过本次实验，为下面对 mapreduce 的进一步应用做了准备。

七、实验成绩

指导教师签名：

年 月 日

实验序号及名称：实验二 统计某电商网站买家收藏商品数量

实验时间：2019 年 4 月 15 日

一、实验目的和要求

1. 编写 mapreduce 代码统计每位用户收藏商品的数量。
2. 掌握 eclipse 如何运行 mapreduce 程序。
3. 进一步巩固对 mapreduce 的了解

二、实验任务

现有某电商网站用户对商品的收藏数据，记录了用户收藏的商品 id 以及收藏日期，名为 buyer_favorite1。buyer_favorite1 包含：买家 id，商品 id，收藏日期这三个字段，数据以“\t”分割，样本数据及格式如下：

1. 买家 id	商品 id	收藏日期
2. 10181	1000481	2010-04-04 16:54:31
3. 20001	1001597	2010-04-07 15:07:52
4. 20001	1001560	2010-04-07 15:08:27
5. 20042	1001368	2010-04-08 08:20:30
6. 20067	1002061	2010-04-08 16:45:33
7. 20056	1003289	2010-04-12 10:50:55
8. 20056	1003290	2010-04-12 11:57:35
9. 20056	1003292	2010-04-12 12:05:29
10. 20054	1002420	2010-04-14 15:24:12
11. 20055	1001679	2010-04-14 19:46:04
12. 20054	1010675	2010-04-14 15:23:53
13. 20054	1002429	2010-04-14 17:52:45
14. 20076	1002427	2010-04-14 19:35:39
15. 20054	1003326	2010-04-20 12:54:44
16. 20056	1002420	2010-04-15 11:24:49
17. 20064	1002422	2010-04-15 11:35:54
18. 20056	1003066	2010-04-15 11:43:01
19. 20056	1003055	2010-04-15 11:43:06
20. 20056	1010183	2010-04-15 11:45:24
21. 20056	1002422	2010-04-15 11:45:49

22.	20056	1003100	2010-04-15	11:45:54
23.	20056	1003094	2010-04-15	11:45:57
24.	20056	1003064	2010-04-15	11:46:04
25.	20056	1010178	2010-04-15	16:15:20
26.	20076	1003101	2010-04-15	16:37:27
27.	20076	1003103	2010-04-15	16:37:05
28.	20076	1003100	2010-04-15	16:37:18
29.	20076	1003066	2010-04-15	16:37:31
30.	20054	1003103	2010-04-15	16:40:14
31.	20054	1003100	2010-04-15	16:40:16

要求编写 MapReduce 程序，统计每个买家收藏商品数量

三、 实验准备方案

包括以下内容：（硬件类实验：实验原理、实验线路、设计方案等）

（软件类实验：所采用的系统、组件、工具、核心方法、框架或流程图、程序清单等）

实验采用的系统：centos 6.5、hadoop2.5.2

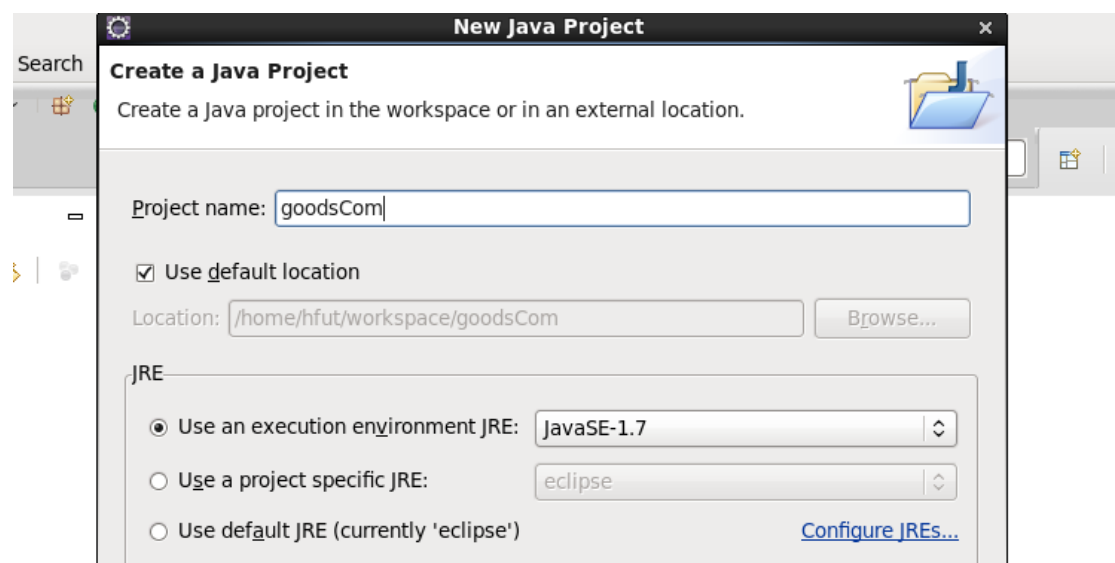
工具：VMware WorkStation

四、 实验用仪器、设备

实验仪器：计算机一台

五、 实验内容与步骤（过程及数据记录）：

1. 新建工程



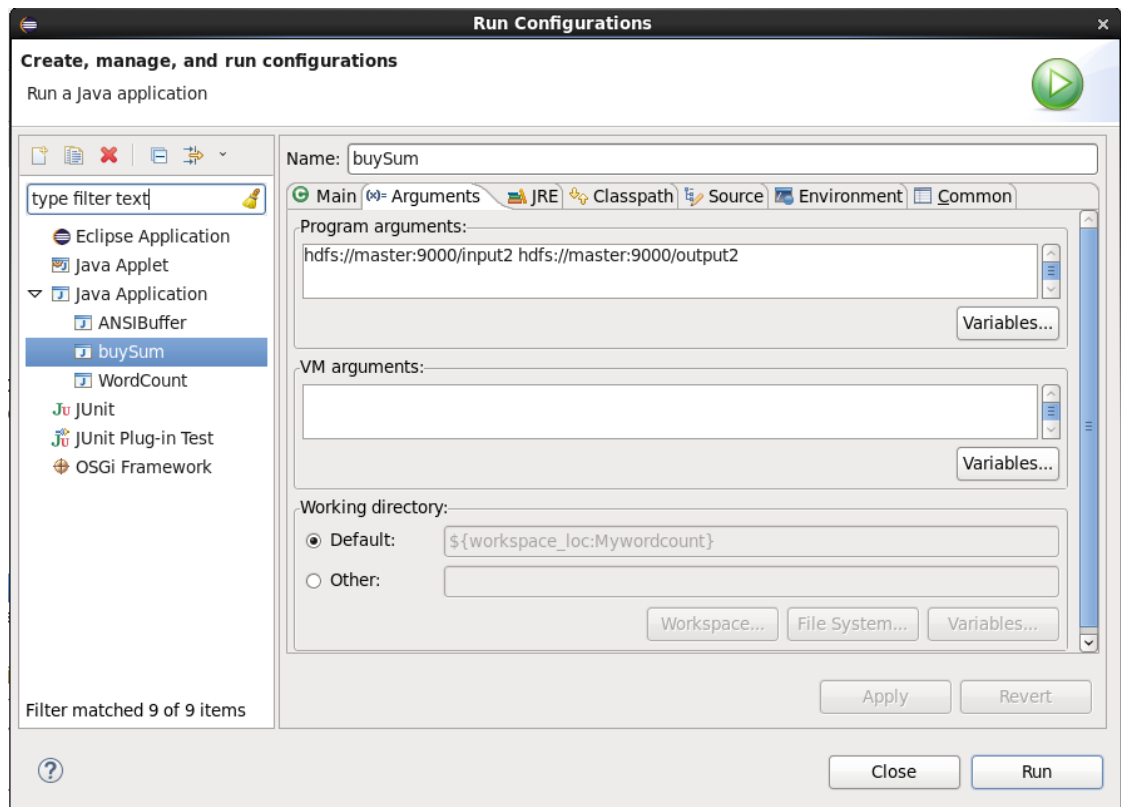
图一

2. 待处理数据结构

表一 原始数据结构

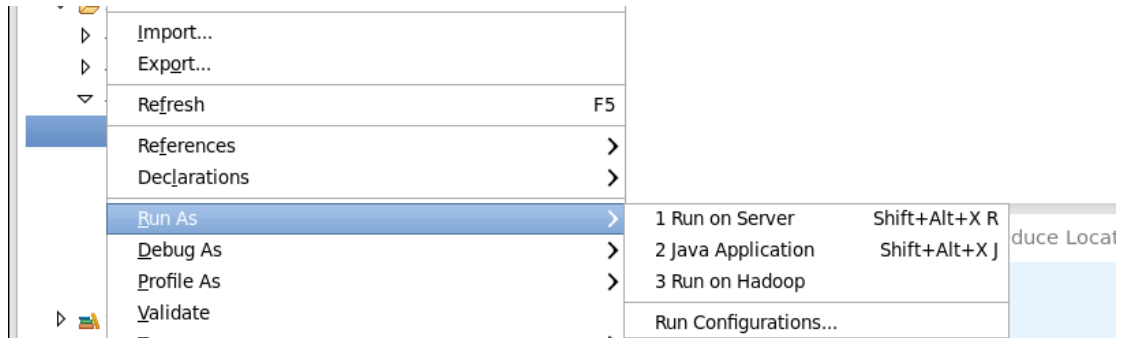
买家 ID	商品 ID	收藏时间
10181	1000481	2010-04-04 16:54:31
20001	1001597	2010-04-07 15:07:52
20001	1001560	2010-04-07 15:08:27
20042	1001368	2010-04-08 08:20:30
20067	1002061	2010-04-08 16:45:33
20056	1003289	2010-04-12 10:50:55
20056	1003290	2010-04-12 11:57:35
20056	1003292	2010-04-12 12:05:29

3. 设置输入输出路径



图二 设置输入输出路径

4. Run->run on Hadoop:



5. 实验结果

1	10181	1
2	120054	4
3	120055	1
4	120056	3
5	120064	1
6	120076	1
7	20001	2
8	20042	1
9	20056	3
10	20067	1
11	220056	6
12	220076	4
13	320054	2
14		

图三 统计结果

6. 处理数据流程

在 map 函数中将每一个 key、value 对的 value 进行分割,分隔符为' \t' , 然后提取提取分割后的列表中的第一项 (即用户 ID), 然后发送一个 key、value 对。在 reduce 函数中将相同 key 值的 value 相加, 即可得到用户收藏的商品总数。

7. 部分代码

Map 函数

```
String[] words = value.toString().split("\t");
/* 使用' \t' 作为分隔符。得到切分列表 */
word.set(words[0]);
context.write(word, one);
/* 发送一个 key, value 对, 表示用户收藏了一项商品。 */
```

Reduce 函数

```

int sum = 0;
for (IntWritable val : values) {
    sum += val.get();
}
/* 对于 key 相同，即同一个用户，统计他收藏的商品数量 */
result.set(sum);
设置将要发出的 value 值。
context.write(key, result);
/* 发送一个 key、value 对，key 和 value
分别为用户 ID 以及用户收藏的商品数量 */
主函数：
job.setJarByClass(buSum.class);
/* 通过 class 名字获取到 class 所在的 jar 包 */
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
/* 设置 map、reduce 和 combiner 的所在类 */
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
/* 设置输入输出参数格式 */
FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
/* 设置输入输出路径 */

```

六、感想、体会、建议

通过本次实验是在上一次 WordCount 的基础上进一步的掌握 mapreduce 编程，通过自己动手写代码可以对 mapreduce 的过程有更深层次的认识。同时也通过 eclipse 更方便的运行 mapreduce 程序。此外为了能够使用 eclipse 也需要续费一定的时间，在此并未详细给出安装过程，通过前面的实验教程可以相对容易的配置成功，避免了很多坑，但是在以往自己配置的过程中还是出现过很多问题，

当时耗费了挺久的时间，归因于网上的部分教程很不详细以及自己对 linux 系统的不熟悉。但是在本次的过程就相对的方便了很多，减少了很多麻烦，过程比较清晰、详细，在此感谢一下老师花费时间给出的详细教程以及上课时的演示。

七、 实验成绩

指导教师签名：

年 月 日

实验序号及名称：实验 三 评论数据采集处理与可视化

实验时间： 2019 年 4 月 28 日

一、 实验目的和要求

1. 爬取京东上的 1000 条评论
2. 将爬取的信息格式化
3. 将评论存储到 mysql 中
4. 使用 jieba 等分词工具分词
5. 制作词云进行可视化

二、 实验任务

爬取京东或淘宝某一商品的评论 1000 条,使用 jieba 分词等分词工具分词,统计词频(使用 MapReduce 或 HBase 或 Hive),并以词云、统计表的方式可视化呈现。

三、 实验准备方案

包括以下内容:(硬件类实验:实验原理、实验线路、设计方案等)

(软件类实验:所采用的系统、组件、工具、核心方法、框架或流程图、程序清单等)

实验采用的系统: centos 6.5、hadoop2.5.2

工具: VMware WorkStation

其他工具: jieba 分词工具, echart, Tomcat server 等

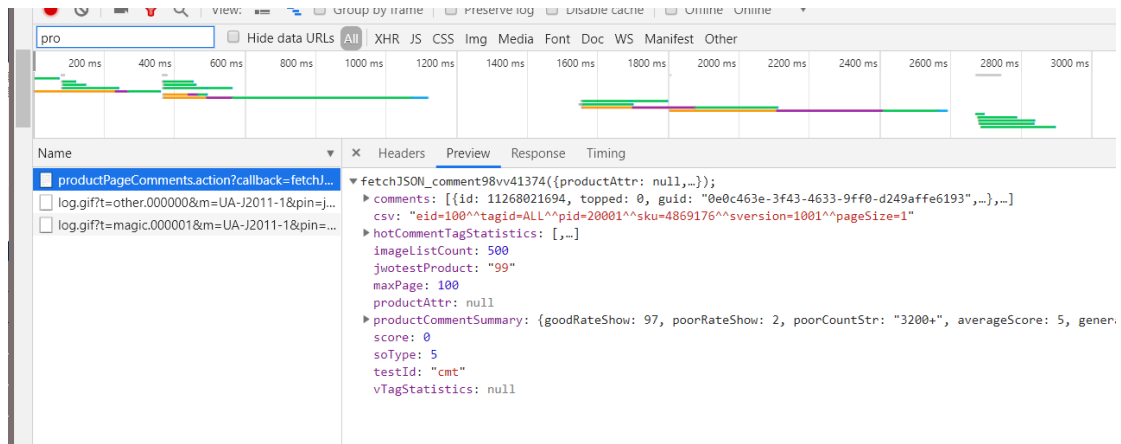
四、 实验用仪器、设备

笔记本一台

五、 实验内容与步骤(过程及数据记录)

1. 数据采集

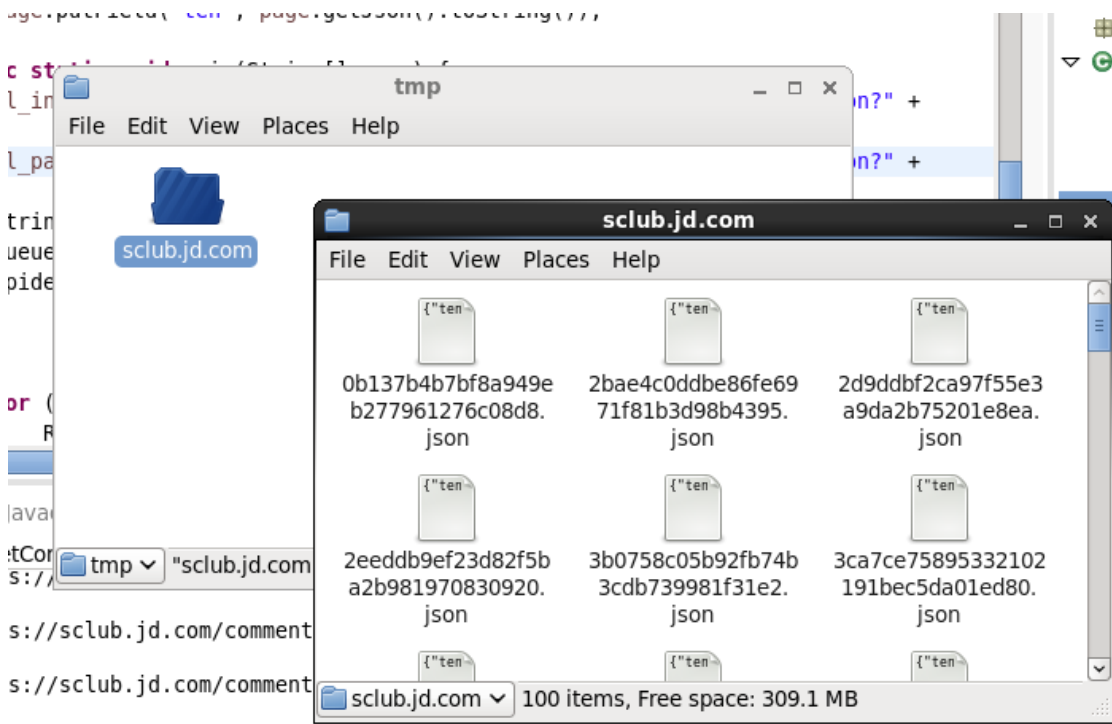
1.1 进入网址 <https://item.jd.com/4432058.html>, 查看浏览器请求的数据文件, 在 network->filter 中输入 pro 选择 productPageComments... 文件, 可以看到文件内容如下图所示:



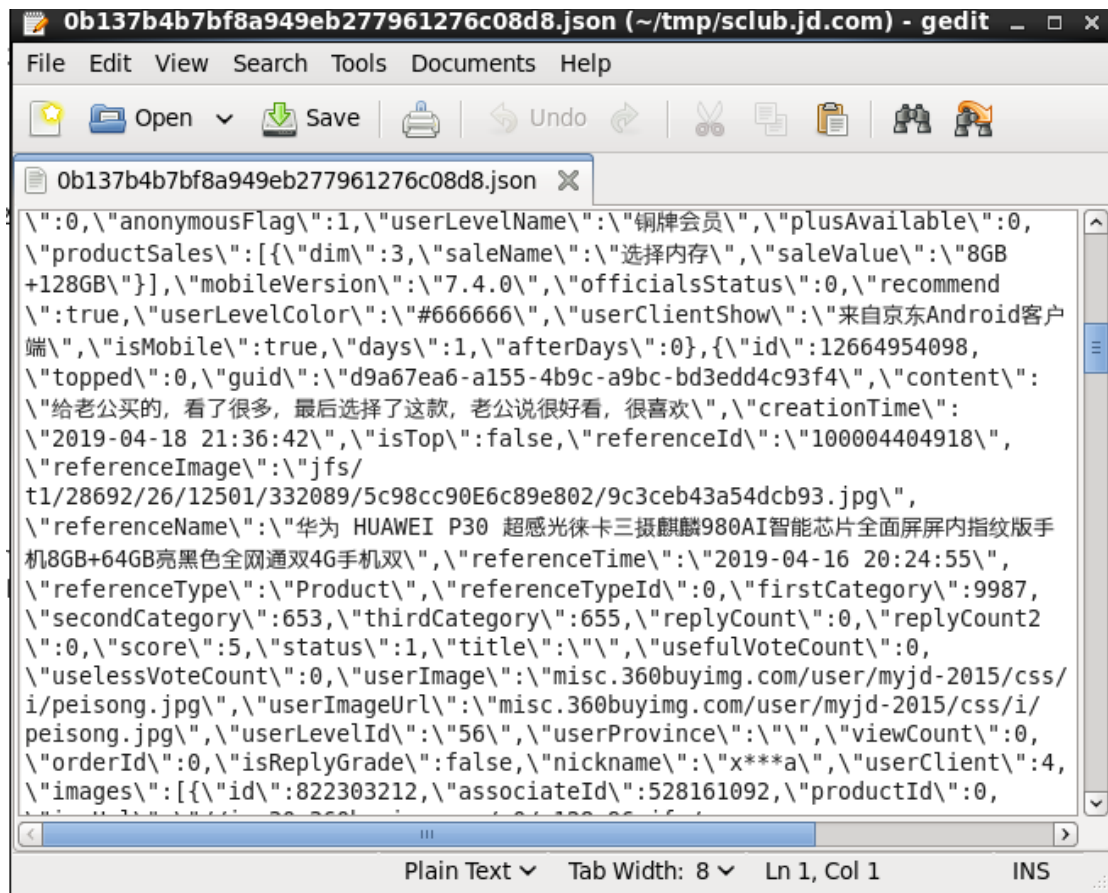
1.2 选中文件，右键复制 call back 在浏览器中打开，查看返回的 Json 形式的评论信息：



1.3 新建 java 工程，以 java project 的形式运行爬虫代码，查看使用 java 爬虫下载的数据是否成功、正确：



从上图可以看到确实是下载了 100 个文件，随即打开一个文件查看内容：



1.4 本次评论所使用的爬虫代码，更改了代理等设置：

```
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)
        .addCookie("TrackID", "15yCkQ1-EBruJdqR00DQ5QscyENaUom43JpYSvjMJpq6anA50TrPtspNL16U8XaLsrVJcwNmEdABTfOHNrnMOW2bDvqpTE7RZbUMrf46d7Y0")
        .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|aed1c4304f694e599245a43
0daeddb2f|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");

    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://sclub.jd.com/comment/productPageComments.action?"
+
"productId=100004404934&score=0&sortType=5&page=0&pageSize=10&isShadowSku=0&fold
=1";

String url_pattern = "https://sclub.jd.com/comment/productPageComments.action?"
+
"productId=100004404934&score=0&sortType=5&pageSize=10&isShadowSku=0&fold=1&page
=";

    String output = "/home/zkpk/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);
    }

```

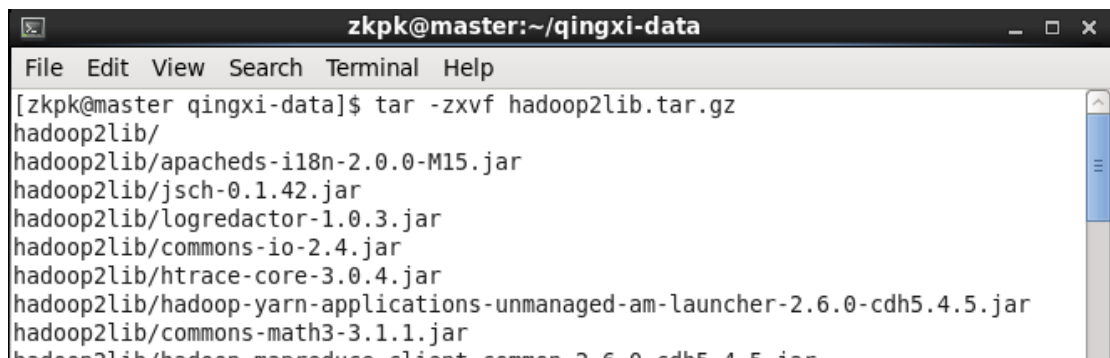
```

        spider.thread(50).run();
    }
}

```

2. 数据清洗搭建解析框架

2.1 准备 hadoop2lib.tar.gz 和 fastjson-1.2.31.jar 软件包并解压到当前文件夹：

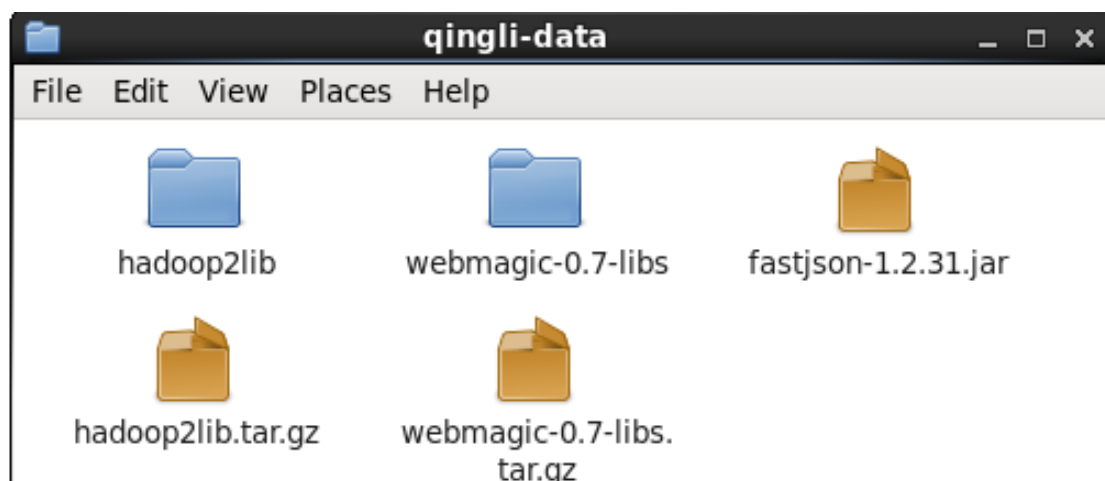


```

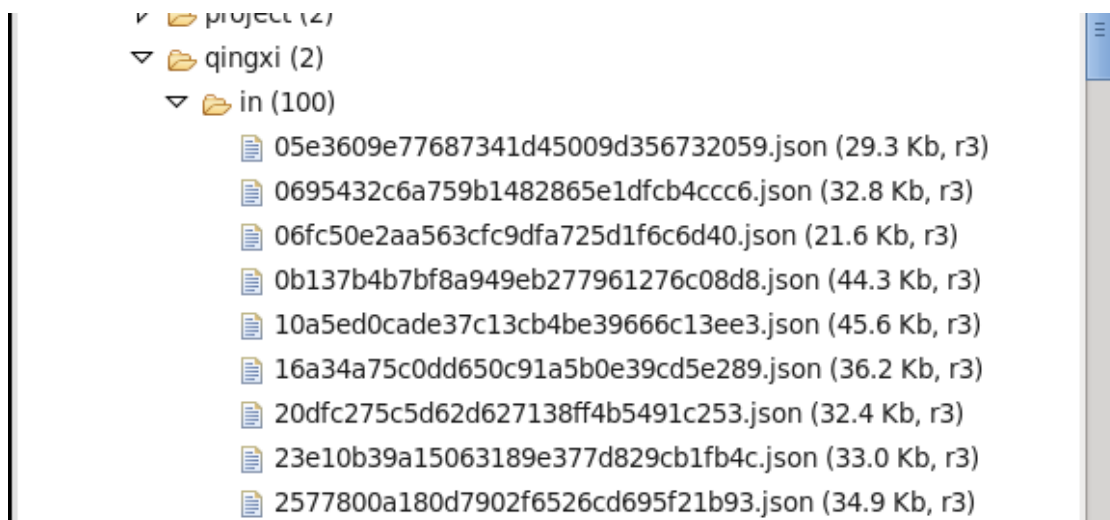
zkpk@master:~/qingxi-data
File Edit View Search Terminal Help
[zkpk@master qingxi-data]$ tar -zxvf hadoop2lib.tar.gz
hadoop2lib/
hadoop2lib/apacheds-i18n-2.0.0-M15.jar
hadoop2lib/jsch-0.1.42.jar
hadoop2lib/logredactor-1.0.3.jar
hadoop2lib/commons-io-2.4.jar
hadoop2lib/htrace-core-3.0.4.jar
hadoop2lib/hadoop-yarn-applications-unmanaged-am-launcher-2.6.0-cdh5.4.5.jar
hadoop2lib/commons-math3-3.1.1.jar
hadoop2lib/hadoop-mapreduce-client-common-2.6.0-cdh5.4.5.jar

```

2.2 解压以及复制结果如下：



2.2 将所有评论上传到 HDFS，并查看是否上传成功，此处为在 eclipse 中查看：



GetComments.jav	QingXijson.java	hdfs://master:9	hdfs://master:9	
12 100004404934	45000	11000	100 100 97 2 1	03c363e1-309e-4209-9492-d49b799364d9 特别
13 100004404934	45000	11000	100 100 97 2 1	04371f57-a1cb-4d0e-a8c6-5f25ed746488 很
14 100004404934	45000	11000	100 100 97 2 1	044f8151-d717-4221-95dd-5701ef039e05 非
15 100004404934	45000	11000	100 100 97 2 1	045b040c-b552-4259-bddb-bffbdd6bffaf 发
16 100004404934	45000	11000	100 100 97 2 1	049f7e6a-9c71-49dd-ba99-7dd23534c00c 首
17 100004404934	45000	11000	100 100 97 2 1	04dcf2d1-6249-40d4-8f20-604602dd7f55 p3
18 100004404934	45000	11000	100 100 97 2 1	04f22b49-0a2c-4733-a855-e80c19245e18 物
19 100004404934	45000	11000	100 100 97 2 1	05021d60-f721-4276-af45-8d115dd4f0c3 手
20 100004404934	45000	11000	100 100 97 2 1	05077591-c267-4c16-89e0-f0a975a333c2 手
21 100004404934	45000	11000	100 100 97 2 1	05539c34-782a-45bb-b637-6f0ad8ebfa98 手
22 100004404934	45000	11000	100 100 97 2 1	05980dc7-636f-4f6d-b135-e9df9edf2084 分
23 100004404934	45000	11000	100 100 97 2 1	05a937aa-1978-49ac-b835-7fa955cbca82 天
24 100004404934	45000	11000	100 100 97 2 1	05e3ee04-2d31-48aa-845f-c14fe02bc40c 艇
25 100004404934	45000	11000	100 100 97 2 1	06215fca-0cec-4a60-be52-c41c8ad546da 快
26 100004404934	45000	11000	100 100 97 2 1	06424229-60f6-445d-a8f9-87e4cd0d90de 欧

[illegible]

2.5.2/etc/hadoop/log4j.properties 文件,拷贝到 qingxi 项目的 src 目录下。

```
<terminated> QingXijson [Java Application] /usr/java/jdk1.7.0_71/bin/java (Apr 26, 2019, 8:25:27 PM)
HDFS: Number of large read operations=0
HDFS: Number of write operations=103
Map-Reduce Framework
  Map input records=100
  Map output records=990
  Map output bytes=306151
  Map output materialized bytes=310368
  Input split bytes=13100
  Combine input records=0
  Combine output records=0
  Reduce input groups=990
  Reduce shuffle bytes=310368
  Reduce input records=990
  Reduce output records=990
  Spilled Records=1980
  Shuffled Maps =100
  Failed Shuffles=0
  Merged Map outputs=100
  GC time elapsed (ms)=9009
  CPU time spent (ms)=0
  Physical memory (bytes) snapshot=0
  Virtual memory (bytes) snapshot=0
  Total committed heap usage (bytes)=15514587136
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
```

2.6 数据清洗代码

```
package my.mr;

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;

public class QingXiJson {
    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);
    }

    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");

```

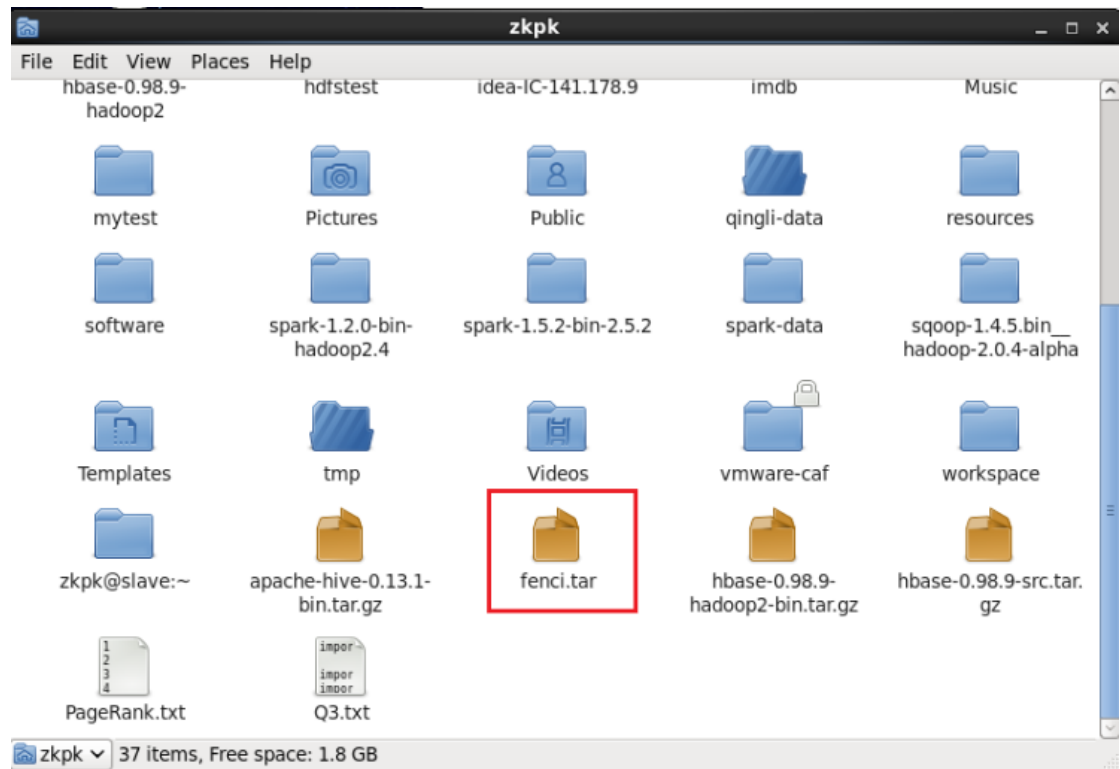
```

        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(""));
    }
}
}
}

```

3. 分词项目搭建

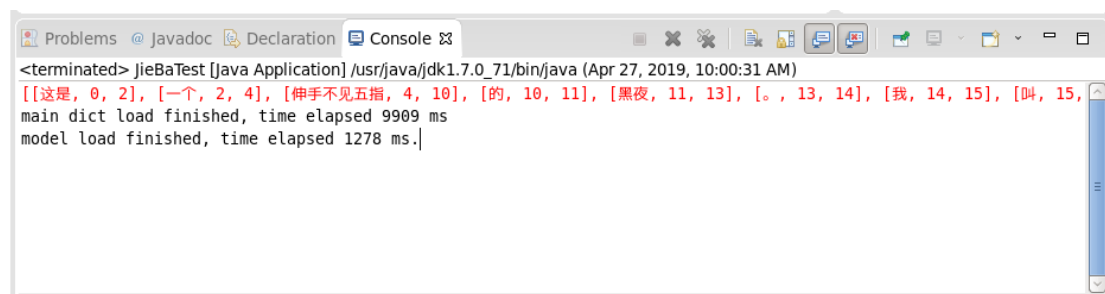
3.1 获取分词项目 fenci.tar:



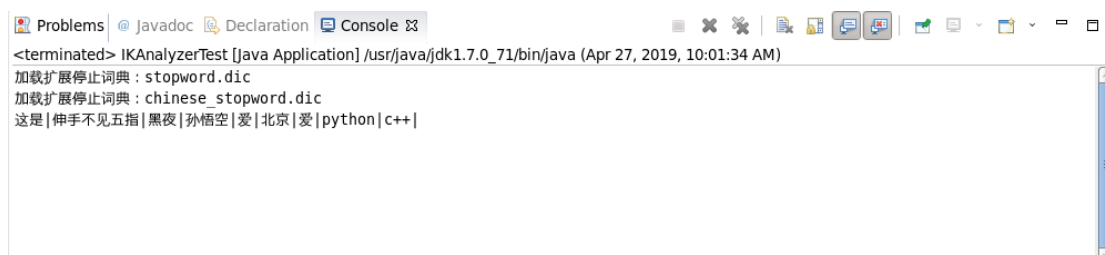
3.2 将 fenci.tar 导入到当前工程当中:



3.3 选择其中的 jieBaTest.java 文件并运行，查看结巴分词效果：



3.4 选择其中的 IKAnalyzerTest.java 文件并运行，查看 IKAnalyzer 分析效果：

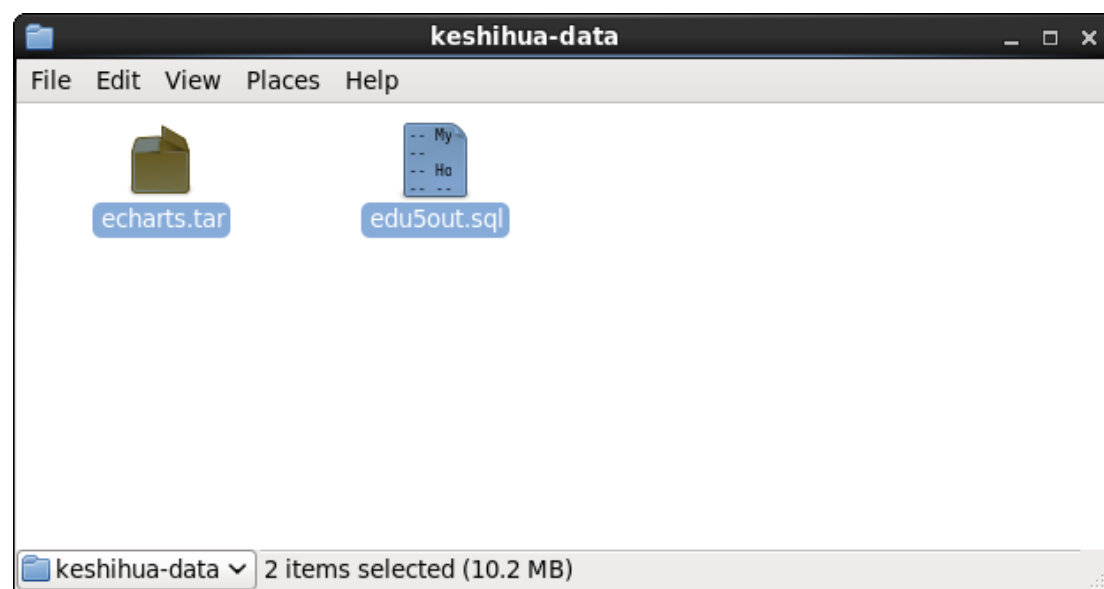


3.5 选择 JcsegTest.java 问价并运行，查看 JcsegTest 分词方法：



从以上结果中可以清楚的看到使用 jieba 和 IKAnalyzer 的分词效果要明显的优于第三个的分词效果，是否可以将人名、地名提取出来在一定程度上可以反应分词工具的好坏。

3.6 加载可视化文件，将 echarts.tar 加载到 eclipse 中：



3.7 使用 service mysqld start 命令打开 mysql 并创建新的数据库：

注意：此处使用的 root 登录。

```
zkpk@master:/home/zkpk/Desktop
File Edit View Search Terminal Help
Starting mysqld: [ OK ]
[root@master Desktop]# service mysqld start
Starting mysqld: [ OK ]
[root@master Desktop]# mysqld -u root
bash: mysqld: command not found
[root@master Desktop]# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database edu5out;
Query OK, 1 row affected (0.12 sec)

mysql> use edu5out;
Database changed
mysql>
```

4. 可视化

4.1 将/home/zkpk/keshihua-data 目录下的 edu5out.sql 脚本导入到 mysql 的 edu5out 库中：

```
mysql> source /home/zkpk/keshihua-data/edu5out.sql;
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

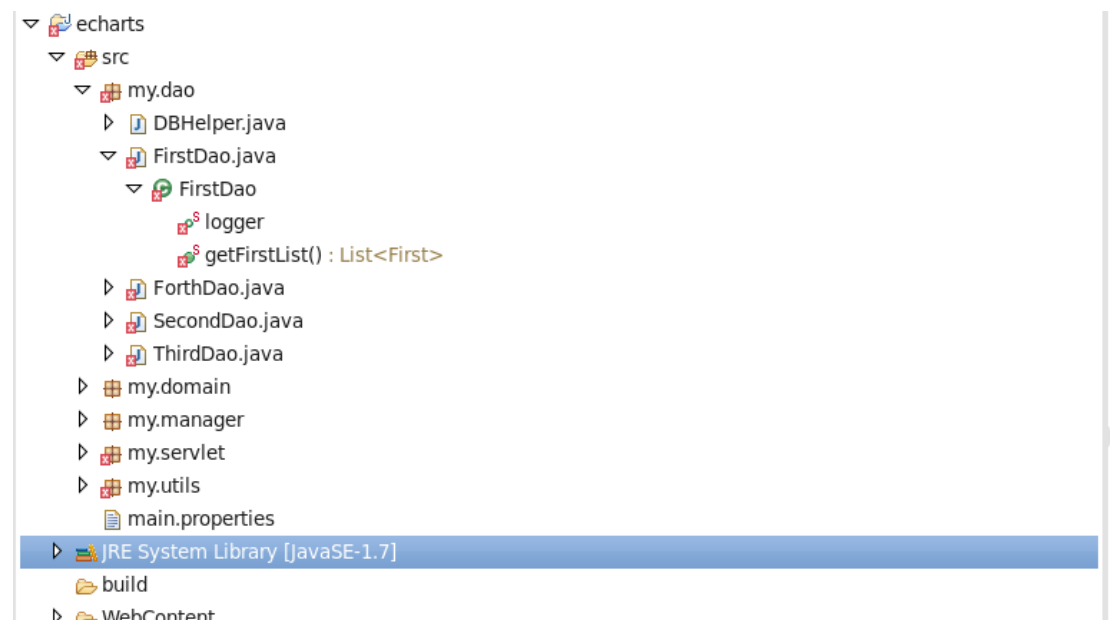
Query OK, 0 rows affected (0.00 sec)
```

4.2 查看当前数据库（edu_5out）中存在有哪些表（show tables）:

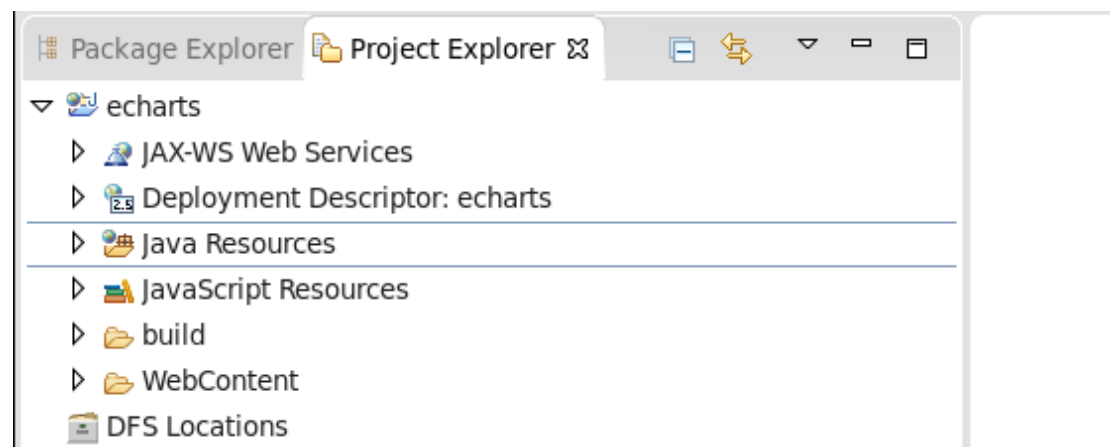
```
mysql> show tables;
+-----+
| Tables_in_edu5out |
+-----+
| creationtimesql   |
| dayssql           |
| ismobilesql       |
| userlevelnamesql  |
+-----+
4 rows in set (0.00 sec)

mysql> █
```

4.3 导入 echarts 包后的项目情况:



4.4 修改 jre system Library，重新配置之后的工程错误得到改正:



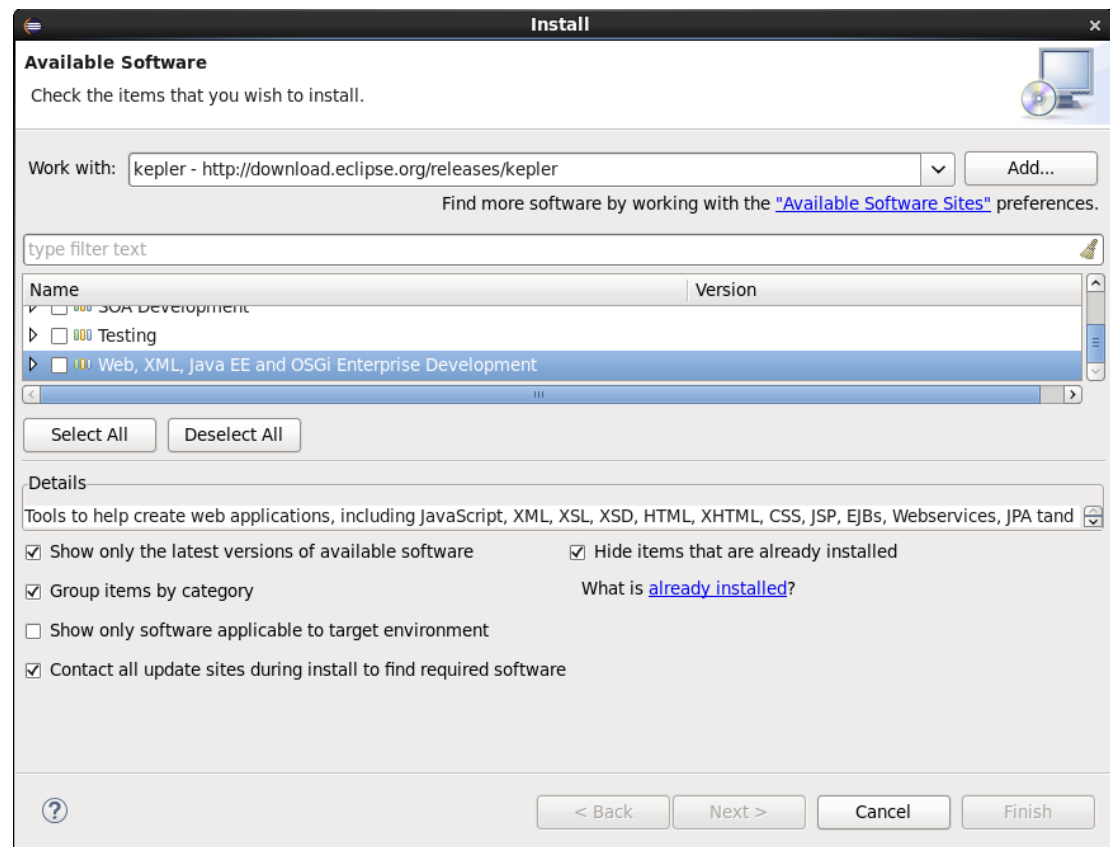
4.5 配置 server (Tomcat)

4.5.1 解决找不到 server 选项的问题:

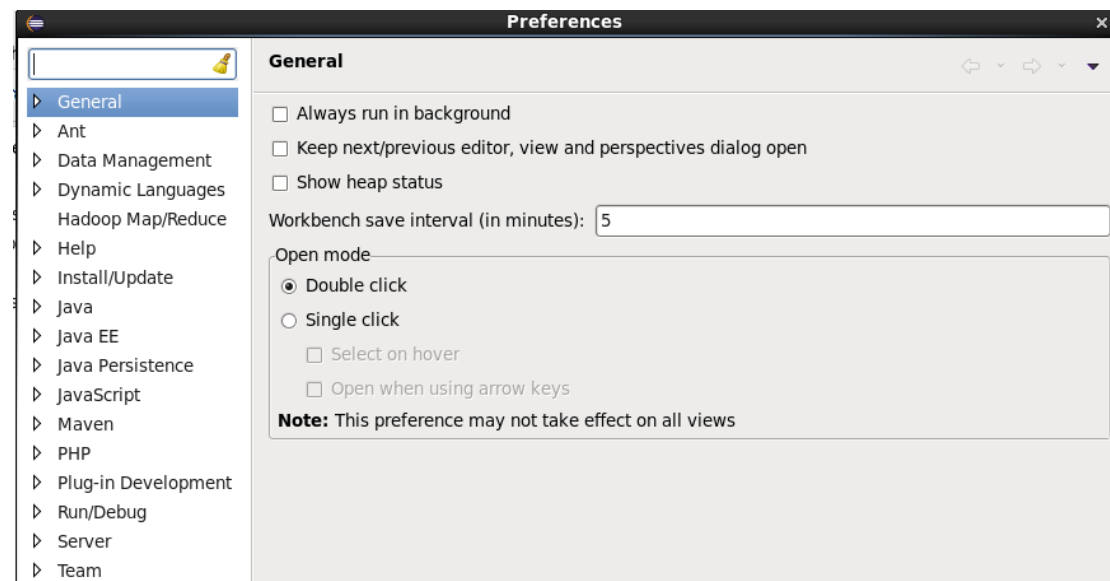
在 work with 中输入:

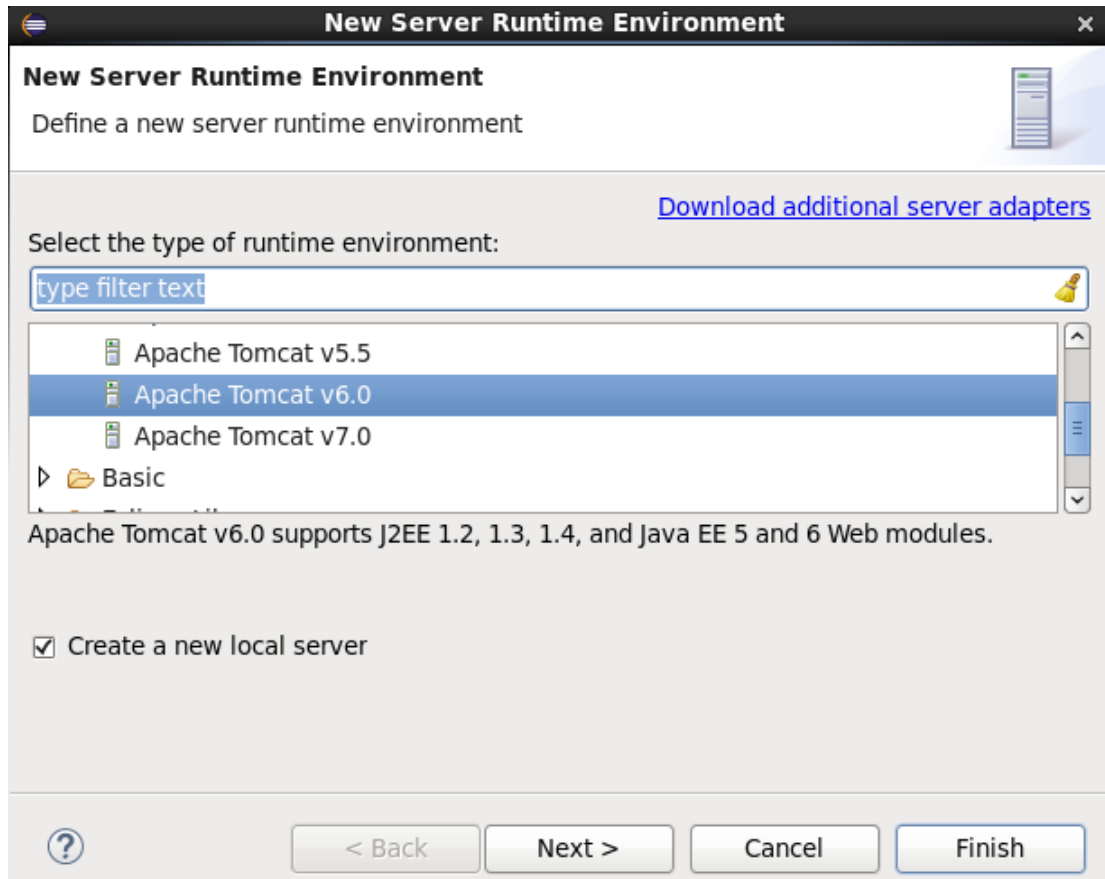
```
kepler - http://download.eclipse.org/releases/kepler
```

找到 Web, XML, Java EE and OSGi Enterprise Development 选项打钩, 然后接下来所有选择 next

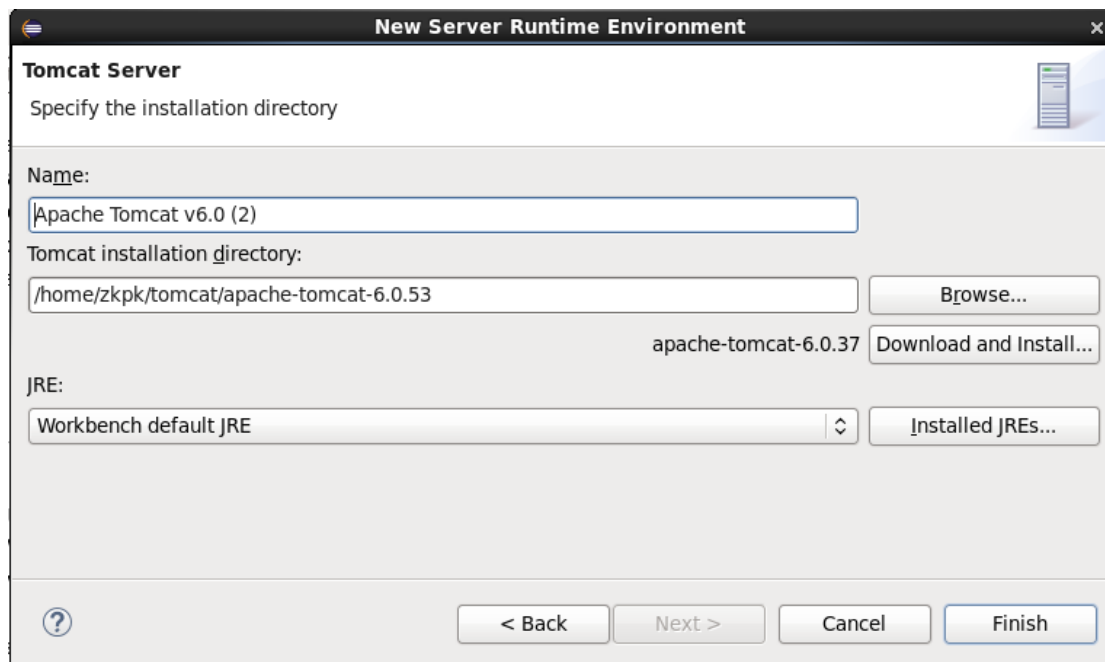


4.5.2 在 window->Preference 中选择 server:

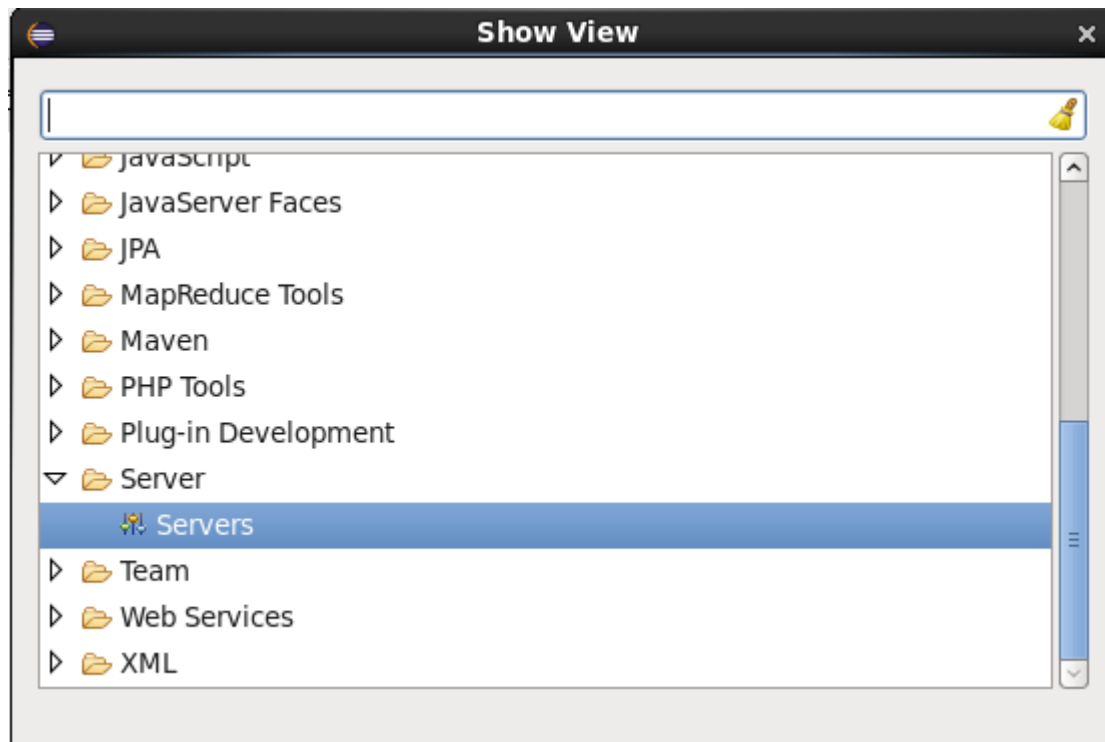




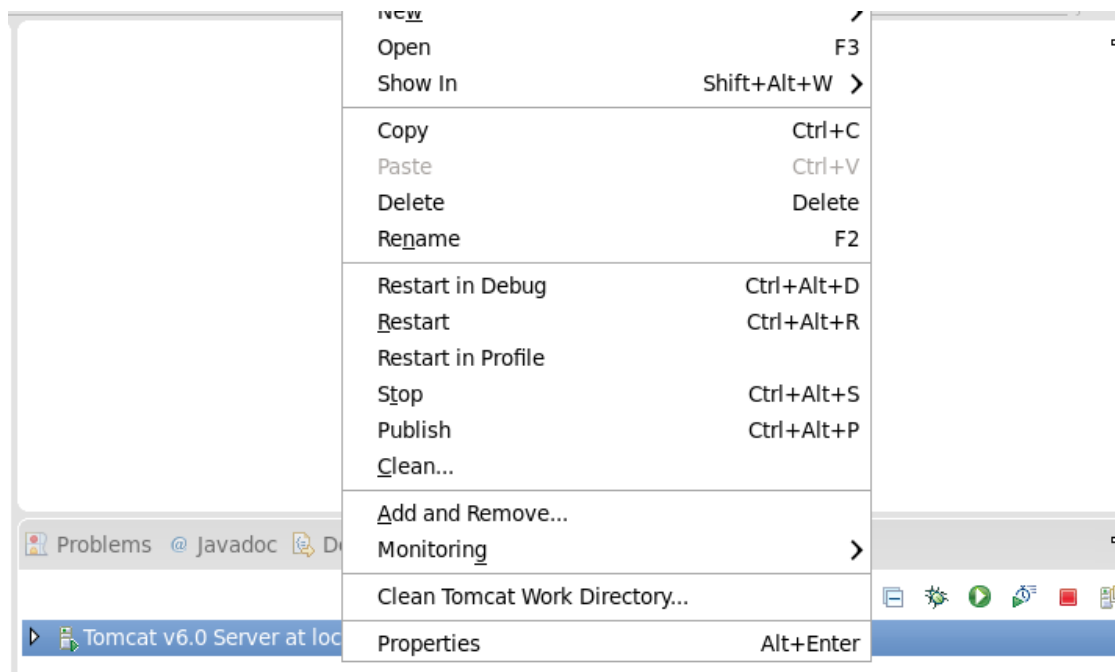
4. 5. 3 选择 tomcat 的安装路径:



4. 5. 4 使 server 选项在下方任务栏显示, 选中 show view, 在 server 中选中 server 选项, 将其加入快捷方式:



4.5.5 在下方任务栏中选中 server，开启 server（右键 start）：



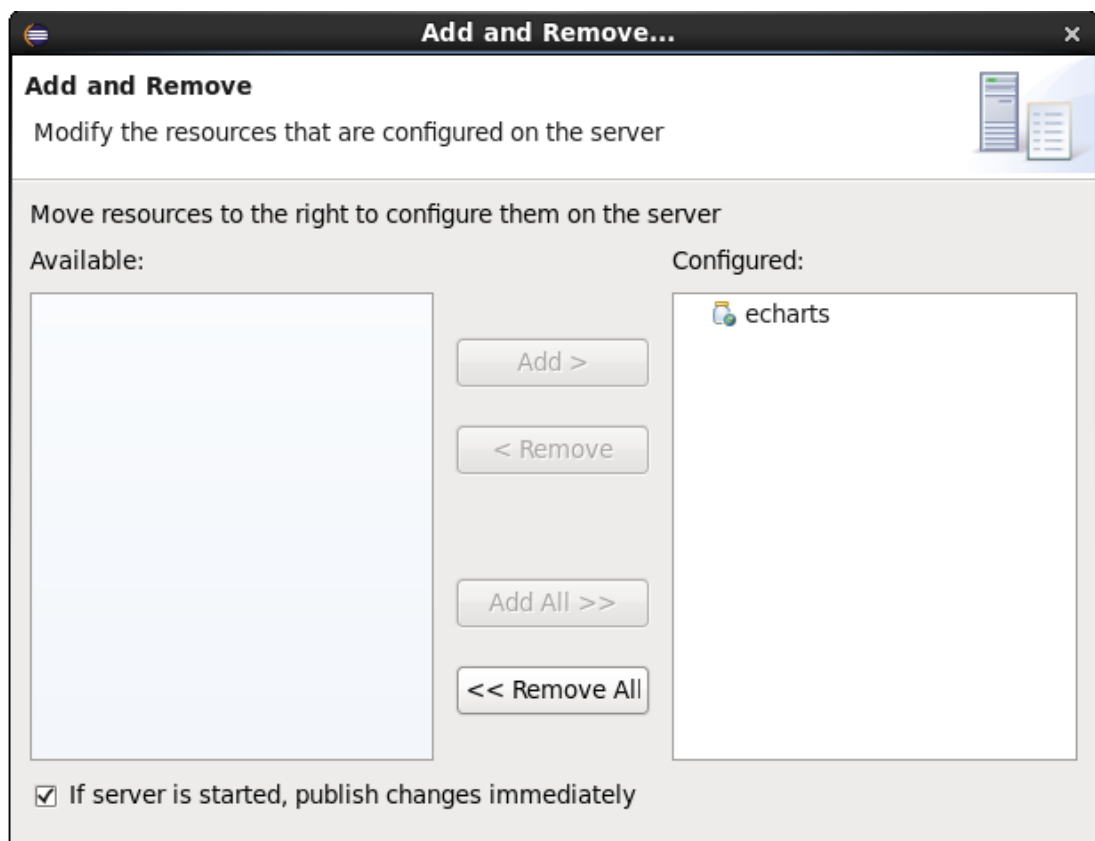
4.6 查看 MySQL 数据库中的 ismobilesql 表（select * from ismobilesql;）其中 1 代表使用的是移动端，0 代表使用的是 PC 端，可以看到两个数据分别为 836 和 164：

```
| user_level | num |
+-----+
4 rows in set (0.00 sec)

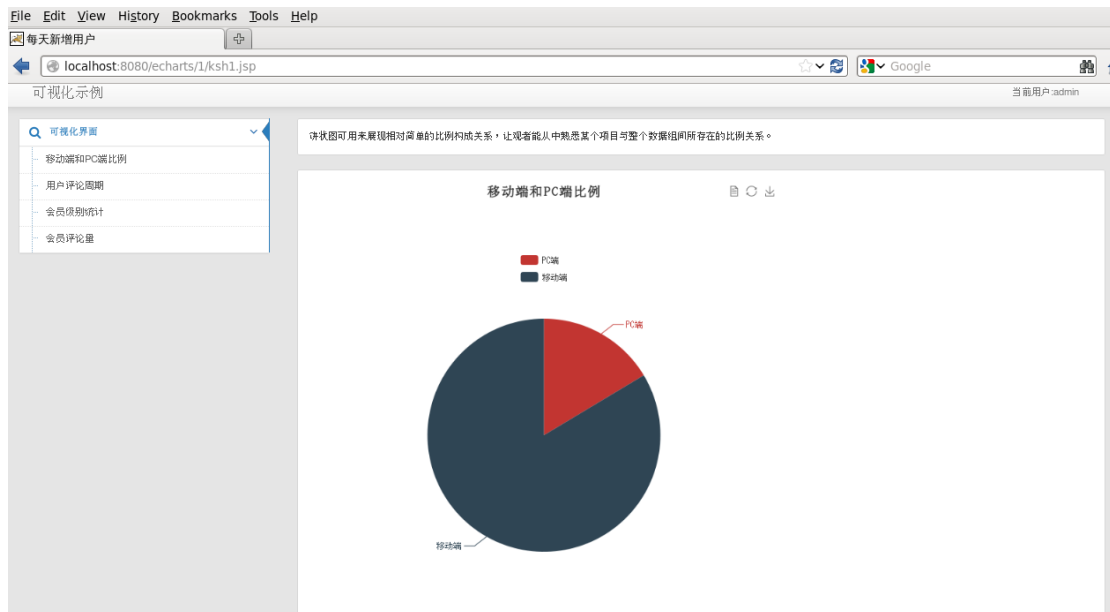
mysql> select * from ismobilesql;
+-----+
| ismobile | num |
+-----+
| 0        | 164 |
| 1        | 836 |
+-----+
2 rows in set (0.00 sec)

mysql> 
```

4.7 在下方的信息栏中选中 server，右键选中 Add and move 将 echarts 加入右方 configured（使该项目在服务器中运行）：



4.8 在浏览器中输入：<http://localhost:8080/echarts/1/ksh1.jsp> 查看效果，可以看到使用移动端和 PC 端的比例和在上方使用 mysql 查询到的数据是一致的，鼠标移动到饼图上面可以具体的看到每一部分的数量也是占比为 836 和 164。



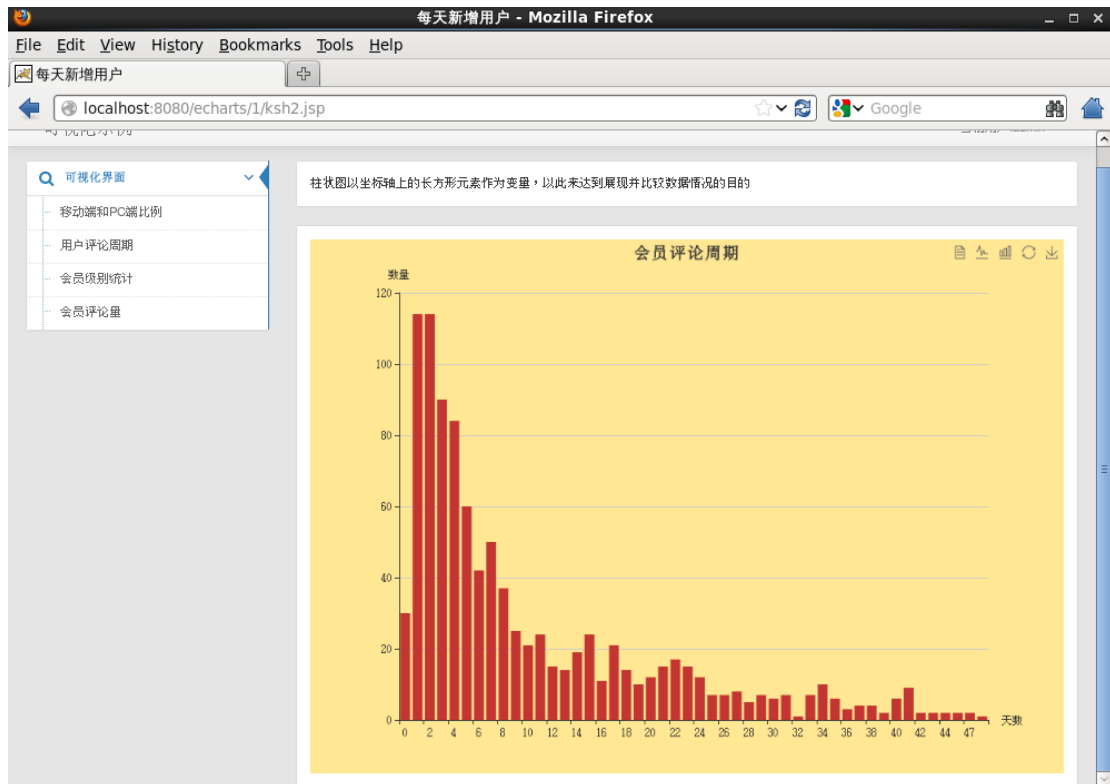
4.9 验证可视化比例是否正确，在 mysql 中输入下方语句，查看 dayssql 语句：

```
select * from dayssql;
```

```
mysql> select * from dayssql;
```

day	num
30	6
28	5
37	4
38	4
36	3
39	2
42	2
43	2
44	2
45	2
47	2
32	1
48	1
1	114
2	114
3	90
4	84
5	60

4.10 在浏览器中输入 <http://localhost:8080/echarts/1/ksh2.jsp> 查看每天的评论数量的直方图，显示效果如下图所示，可以看到和上面的使用 mysql 查出来的数据保持一致。



4.11 查看可视化结构是否正确，下图由于在创建数据库的时候编码错误因此出现乱码，在后面的过程中已经改正。

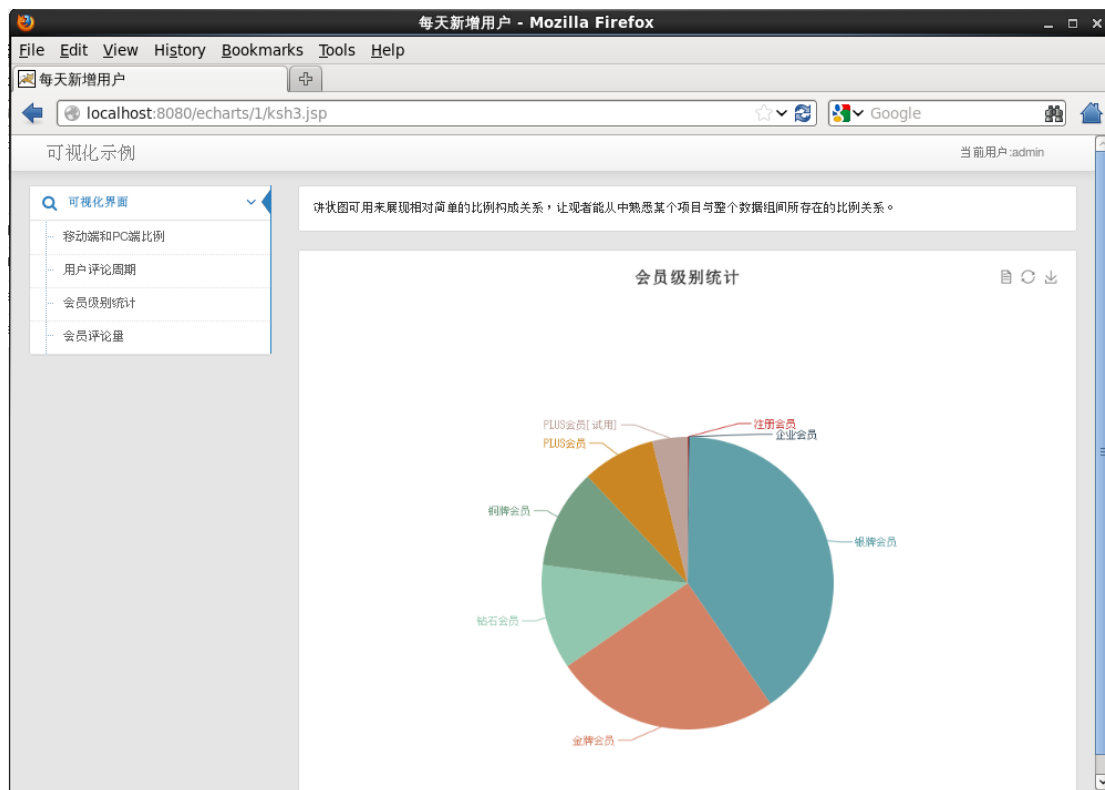
```
select * from userlevelnamesql;
```

```
zkpk@master:/home/zkpk/Desktop
File Edit View Search Terminal Help
+-----+
| 29 | 7 |
| 26 | 7 |
| 33 | 7 |
| 35 | 6 |
| 40 | 6 |
+-----+
48 rows in set (0.02 sec)

mysql> select * from userlevelnamesql;
+-----+
| userlevelname | num |
+-----+
| ???? | 1 |
| ???? | 1 |
| ???? | 402 |
| ???? | 250 |
| ???? | 116 |
| ???? | 110 |
| PLUS?? | 81 |
| PLUS??[??] | 39 |
+-----+
8 rows in set (0.03 sec)

mysql>
```

4.12 在浏览器中输入 <http://localhost:8080/echarts/1/ksh3.jsp> 查看各个等级会员所占的比例和上图一致。



4.13 使用 mysql 查看 creationtime 中的数据:

```
select * from creationtimesql;
```

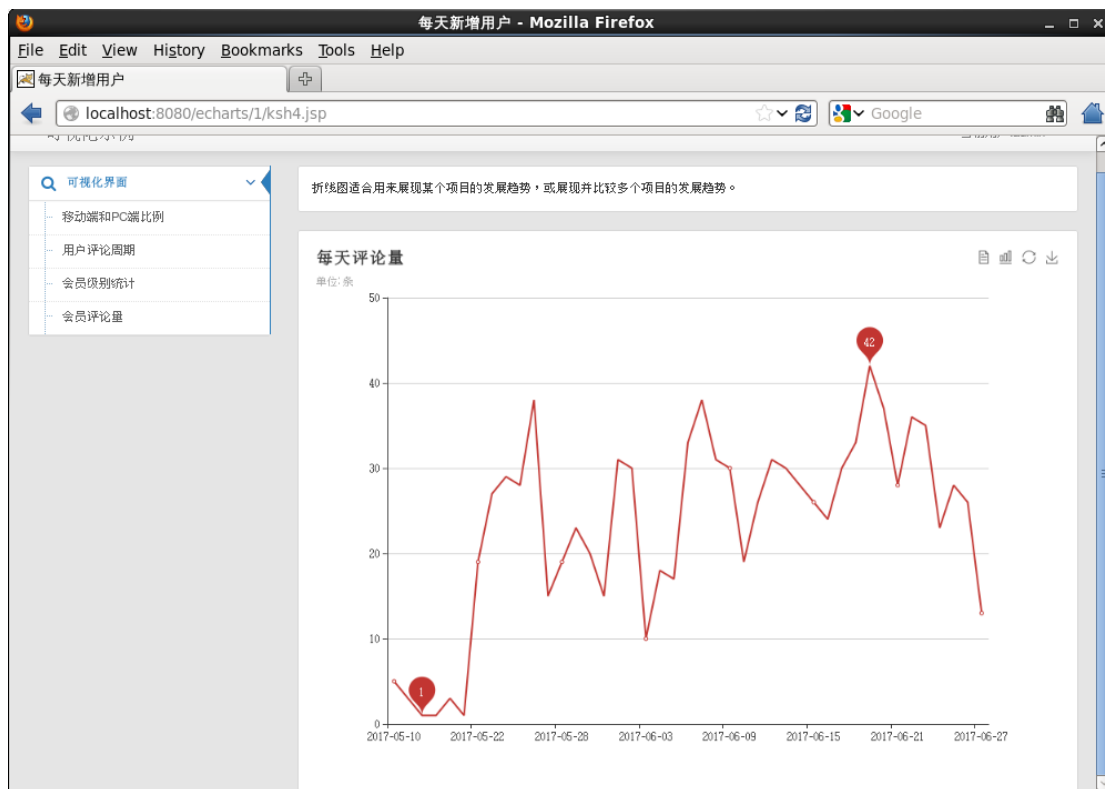
zkpk@master:/home/zkpk/Desktop

File Edit View Search Terminal Help

mysql> select * from creationtimesql;

creationtime	num
2017-06-19	42
2017-06-07	38
2017-05-26	38
2017-06-20	37
2017-06-22	36
2017-06-23	35
2017-06-06	33
2017-06-18	33
2017-06-08	31
2017-06-01	31
2017-06-12	31
2017-06-09	30
2017-06-02	30
2017-06-13	30
2017-06-17	30
2017-05-24	29
2017-06-14	28
2017-05-25	28
2017-06-21	28
2017-06-25	28

4.14 在浏览器中输入 <http://localhost:8080/echarts/1/ksh4.jsp> 查看每天的评论量。



5. 制作词云

5.1 配置字符编码，将数据库中的编码改为 UTF-8。

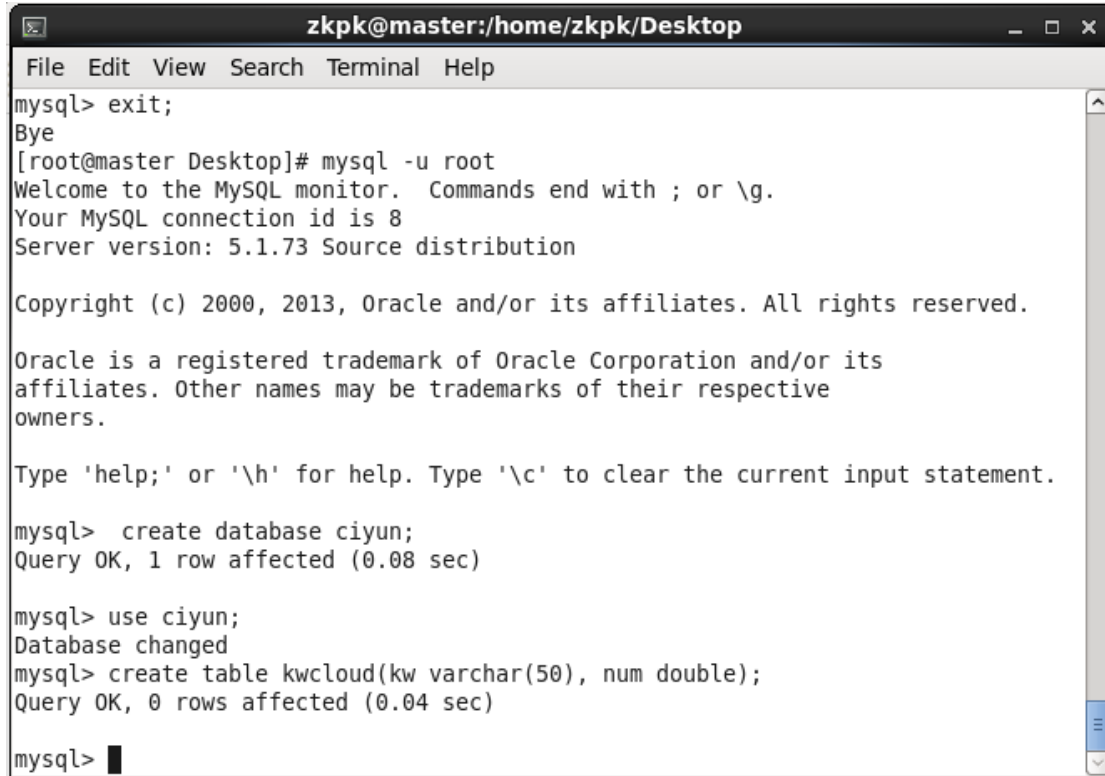
```
zkpk@master:/home/zkpk
File Edit View Search Terminal Help

[mysqld]
character-set-server=utf8
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
user=mysql
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0

[mysqld_safe]
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
[client]
default-character-set=utf8
[mysql]
default-character-set=utf8
~
~
~
```

5.2 重启 mysql 并建立数据库以及表：（注意在建立数据库的时候也要指定编码格式，不然在实验过程中仍然出错）。

```
create database ciyun default character set utf8 collate
utf8_general_ci;
```



```
zkpk@master:/home/zkpk/Desktop
File Edit View Search Terminal Help
mysql> exit;
Bye
[root@master Desktop]# mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database ciyun;
Query OK, 1 row affected (0.08 sec)

mysql> use ciyun;
Database changed
mysql> create table kwcloud(kw varchar(50), num double);
Query OK, 0 rows affected (0.04 sec)

mysql> █
```

5.3 向 kwcloud 表中插入如下数据:

```
mysql> insert into kwcloud(kw, num) values
-> ('合肥工业大学', 100),
-> ('计算机与信息学院', 99),
-> ('计算机系', 98),
-> ('16级', 97),
-> ('数据与智能工程', 96),
-> ('大数据处理技术', 95),
-> ('centos', 94),
-> ('hadoop', 93),
-> ('hdfs', 92),
-> ('mapreduce', 91),
-> ('hbase', 90),
-> ('hive', 89),
-> ('爬虫', 88),
-> ('可视化', 87),
-> ('词云', 86),
-> ('Java', 85),
-> ('mysql', 84),
-> ('tomcat', 83),
-> ('vmware', 82),
-> ('ssh', 81),
-> ('eclipse', 80),
-> ('jre', 79),
-> ('apache', 78),
-> ('分词', 77),
-> ('json', 76),
-> ('spark', 75),
-> ('pig', 74),
-> ('zookeeper', 73),
-> ('mahout', 72),
-> ('sqoop', 71),
-> ('kafka', 70),
-> ('storm', 69);
Query OK, 32 rows affected, 10 warnings (0.03 sec)
Records: 32 Duplicates: 0 Warnings: 0
```

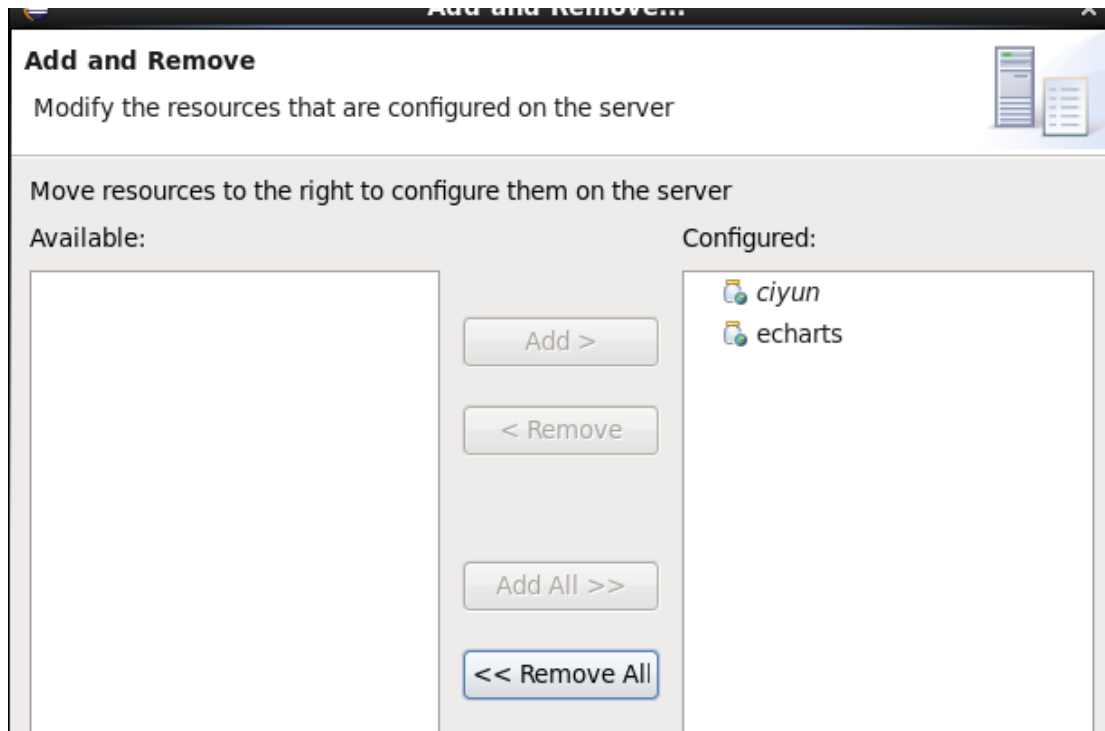
5.4 使用查询语句检查是否有乱码数据:

```
mysql> select * from kwcloud;
```

kw	num
合肥工业大学	100
计算机与信息学院	99
计算机系	98
16级	97
数据与智能工程	96
大数据处理技术	95
centos	94
hadoop	93
hdfs	92
mapreduce	91
hbase	90
hive	89
爬虫	88
可视化	87
词云	86
Java	85
mysql	84
tomcat	83
vmware	82
ssh	81
eclipse	80
jre	79
apache	78
分词	77
json	76
spark	75
pig	74
zookeeper	73
mahout	72
sqoop	71
kafka	70
storm	69

```
32 rows in set (0.00 sec)
```

5.5 将词云项目加到 configured 中，使其在 server 中运行：



5.6 在浏览器中输入 <http://localhost:8080/ciyun/1/ksh1.jsp> 查看词云展示。



六、感想、体会、建议

在本次实验中，我们使用了几种分词工具，比较了这几种分词工具的好坏，使用了 mysql, tomcat server, 也使用词云，使用了简易的爬虫工具，生成了网页。在这个过程中学到了很多东西，其中有些也是在后面的综合设计中会使用到的，例如可视化工具。在这个实验中也涉及到了一些以前没有接触过的东西，动手解决了一些没有接触到的方向中的问题，例如在导入 echart 包的时候，配置 jre System Library 但是没有教程中的解决方法，最后得出的解决方法为先配置 Tomcat Server，问题迎刃而解，这种情况也在其他地方出现过，最终凭借详细的实验指导得到解决。通过本课程/实验，我相信我具备了学习一种新的大数据技术的基本能力，达到了本课程/实验的目的。通过有意思的实验寓教于乐是我的一个很大的感受。

七、实验成绩

指导教师签名：

年 月 日