he **developer experience** like at your tion? Do you have a **platform** ring team? We want to hear from you!

Take the Survey

urvey.alchemer.com/s3/8146709/dzann)

Stop letting outdated access processes hold your team back. Learn how to build smarter, faster, and more secure DevOps infrastructures.

Save Your Seat

(https://cvent.me/0Zdom1?utm_source=TeleportAnnouncementBar&utm_medium=SiteAds&utm_campaign=TeleportAnnouncementBar)

For Java apps, containerization helps solve the majority of challenges related to portability and consistency. See how.

Read the Refcard

(https://dzone.com/link/2024-rc-java-application-containerization-and-deployment)

Far too many vulner have been introduce software products. **treat your supply c security as an afterthought.**

Read the Refc

(https://dzone.com/li rc-software-supply-c security-and-an)

RELATED

Redis-Based Tomcat Session Management (/articles/redis-based-tomcat-session-management)

Optimizing Java Applications for AWS Lambda (/articles/java-apps-aws-lambda)

Buildpacks: An Open-Source Alternative to Chainguard (/articles/buildpacks-open-source-alternative-to-chainguard)

Efficient Asynchronous Processing Using CyclicBarrier and CompletableFuture in Java (/articles/efficient-asynchronous-processing-using-cyclicbarr)

Partner Resources

DZone (https://dzone.com) / Data Engineering (https://dzone.com/data-engineering) / Big Data (https://dzone.com/big-data) / Word Count Program With MapReduce and Java

(/) REFCARDS (/REFCARDZ) TREND REPORTS (/TREND-REPORTS) EVENTS (/users/login.html)

Culture and Methodologies (/culture-and-methodologies) Data Engineering (/data-engineering) Software Design and Architecture (/software-design-and-architecture) Coding (/coding) Testing, Deployment, and Maintenance (/testing-deployment-and-maintenance)

Word Count Program With MapReduce and Java

In this post, we provide an introduction to the basics of MapReduce, along with a tutorial to create a word count app using Hadoop and Java.

By Shital Kat (/users/2752223/shitalkatkar.html) · Mar. 03, 16 · Tutorial

In Hadoop, MapReduce (https://dzone.com/articles/mapreduce-design-patterns-1) is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

MapReduce consists of 2 steps:

- **Map Function –** It takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (Key-Value pair).

  Example – (Map function in Word Count)

Input	Set of data	Bus, Car, bus, car, train, car, bus, car, train, bus, TRAIN,BUS, buS, caR, CAR, car, BUS, TRAIN
Output	Convert into another set of data (Key,Value)	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)

- **Reduce Function –** Takes the output from Map as an input and combines those data tuples into a smaller set of tuples.

  Example – (Reduce function in Word Count)

Input **(output of Map function)**	Set of Tuples	(Bus,1), (Car,1), (bus,1), (car,1), (train,1), (car,1), (bus,1), (car,1), (train,1), (bus,1), (TRAIN,1),(BUS,1), (buS,1), (caR,1), (CAR,1), (car,1), (BUS,1), (TRAIN,1)
		(BUS,7),

ABOUT US
About DZone (/pages/about)
Support and feedback (mailto:support@dzone.com)
Community research (/pages/dzone-community-research)
Sitemap (/sitemap)

ADVERTISE
Advertise with DZone (https://advertise.dzone.com)

CONTRIBUTE ON DZONE
Article Submission Guidelines (/articles/dzones-article-submission-guidelines)
Become a Contributor (/pages/contribute)
Core Program (/pages/core)
Visit the Writers' Zone (/writers-zone)

LEGAL
Terms of Service (https://technologyadvice.com/terms-conditions/)
Privacy Policy (https://technologyadvice.com/privacy-policy/)

CONTACT US
3343 Perimeter Hill Drive
Suite 100
Nashville, TN 37211
support@dzone.com (mailto:support@dzone.com)

Let's be friends:
(/pages/feeds) (https://twitter.com/DZoneInc) (https://www.facebook.com/DZoneInc) (https://www.linkedin.com/company/dzone/)

Fig. WorkFlow of MapReducing

Workflow of MapReduce consists of 5 steps:

1. **Splitting** – The splitting parameter can be anything, e.g. splitting by space, comma, semicolon, or even by a new line ('\n').

2. **Mapping** – as explained above.

3. **Intermediate splitting** – the entire process in parallel on different clusters. In order to group them in "Reduce Phase" the similar KEY data should be on the same cluster.

4. **Reduce** – it is nothing but mostly group by phase.

5. **Combining** – The last phase where all the data (individual result set from each cluster) is combined together to form a result.

Now Let's See the Word Count Program in Java

Fortunately, we don't have to write all of the above steps, we only need to write the splitting parameter, Map function logic, and Reduce function logic. The rest of the remaining steps will execute automatically.

Make sure that Hadoop is installed on your system with the Java SDK.

Steps

1. Open Eclipse> File > New > Java Project >(Name it – MRProgramsDemo) > Finish.
2. Right Click > New > Package (Name it - PackageDemo) > Finish.
3. Right Click on Package > New > Class (Name it - WordCount).
4. Add Following Reference Libraries:
    1. Right Click on Project > Build Path> Add External
        1. */usr/lib/hadoop-0.20/**hadoop-core.jar***
        2. *Usr/lib/hadoop-0.20/lib/**Commons-cli-1.2.jar***

5. Type the following code:

```
package PackageDemo;

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;




public class WordCount {
public static void main(String [] args) throws Exception
{
Configuration c=new Configuration();
String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
Path input=new Path(files[0]);
Path output=new Path(files[1]);
Job j=new Job(c,"wordcount");
j.setJarByClass(WordCount.class);
j.setMapperClass(MapForWordCount.class);
j.setReducerClass(ReduceForWordCount.class);
j.setOutputKeyClass(Text.class);
j.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(j, input);
FileOutputFormat.setOutputPath(j, output);
System.exit(j.waitForCompletion(true)?0:1);
}
public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>{
public void map(LongWritable key, Text value, Context con) throws IOException, InterruptedException
{
```

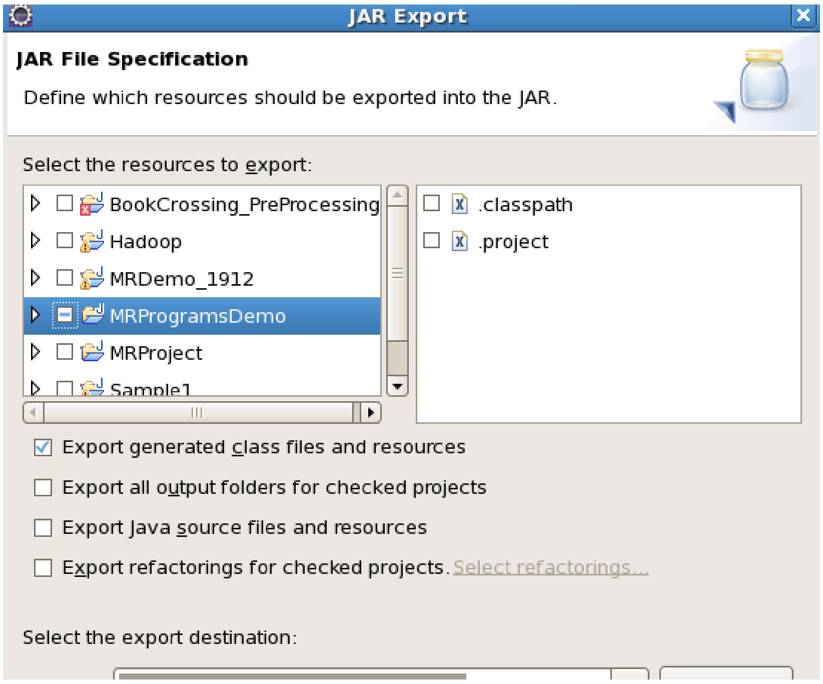
```
String line = value.toString();
String[] words=line.split(",");
for(String word: words )
{
      Text outputKey = new Text(word.toUpperCase().trim());
  IntWritable outputValue = new IntWritable(1);
  con.write(outputKey, outputValue);
}
}
}

public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text, IntWritable>
{
public void reduce(Text word, Iterable<IntWritable> values, Context con) throws IOException, InterruptedExc
{
int sum = 0;
   for(IntWritable value : values)
   {
   sum += value.get();
   }
   con.write(word, new IntWritable(sum));
}

}

}
```

The above program consists of three classes:

- Driver class (Public, void, static, or main; this is the entry point).
- The `Map` class which **extends** the public class Mapper<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the `Map` function.
- The `Reduce` class which extends the public class Reducer<KEYIN,VALUEIN,KEYOUT,VALUEOUT> and implements the `Reduce` function.

6. Make a jar file

Right Click on Project> Export> Select export destination as **Jar File** > next> Finish.



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

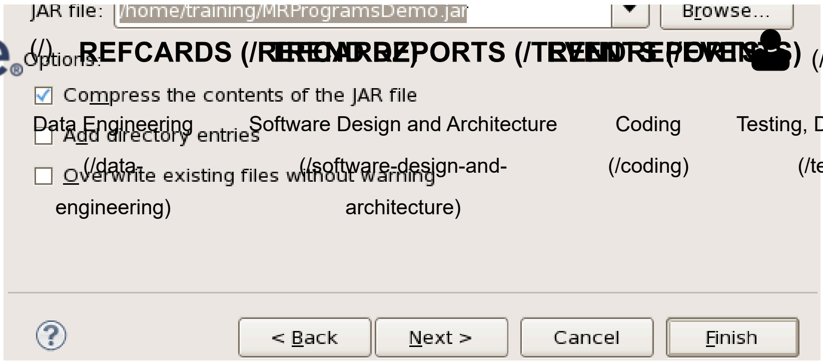
Select the resources to export:

☐ BookCrossing_PreProcessing
☐ Hadoop
☐ MRDemo_1912
⊟ MRProgramsDemo
☐ MRProject
☐ Sample1

☐ .classpath
☐ .project

☑ Export generated class files and resources
☐ Export all output folders for checked projects
☐ Export Java source files and resources
☐ Export refactorings for checked projects. Select refactorings...

Select the export destination:



JAR file: /home/training/MRProgramsDemo.jar

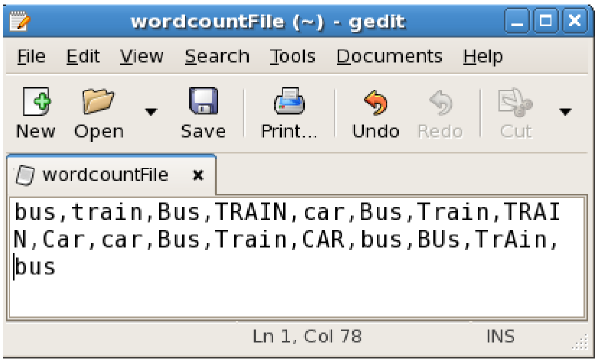
Options:

☑ Compress the contents of the JAR file

☐ Add directory entries

☐ Overwrite existing files without warning

7. Take a text file and move it into HDFS format:



```
bus,train,Bus,TRAIN,car,Bus,Train,TRAI
N,Car,car,Bus,Train,CAR,bus,BUs,TrAin,
bus
```

To move this into Hadoop directly, open the terminal and enter the following commands:

```
[training@localhost ~]$ hadoop fs -put wordcountFile wordCountFile
```

8. Run the jar file:

(Hadoop jar jarfilename.jar packageName.ClassName PathToInputTextFile PathToOutputDirectry)

```
[training@localhost ~]$ hadoop jar MRProgramsDemo.jar PackageDemo.WordCount wordCountFile MRDir1
```

9. Open the result:

```
[training@localhost ~]$ hadoop fs -ls MRDir1

Found 3 items

-rw-r--r--   1 training supergroup          0 2016-02-23 03:36 /user/training/MRDir1/_SUCCESS
drwxr-xr-x   - training supergroup          0 2016-02-23 03:36 /user/training/MRDir1/_logs
-rw-r--r--   1 training supergroup         20 2016-02-23 03:36 /user/training/MRDir1/part-r-00000
```

```
[training@localhost ~]$ hadoop fs -cat MRDir1/part-r-00000
BUS     7
CAR     4
TRAIN   6
```

Hadoop   MapReduce   Java (Programming Language)

Opinions expressed by DZone contributors are their own.

RELATED

Redis-Based Tomcat Session Management

Optimizing Java Applications for AWS Lambda

Buildpacks: An Open-Source Alternative to Chainguard

Efficient Asynchronous Processing Using CyclicBarrier and CompletableFuture in Java