HDFS操作实验（hdfs文件上传、使用JavaAPI判断文件存在，文件合并）

原创

C.js 于 2018-03-25 00:12:28 发布 10662 收藏 29

此博客为博主学习总结，内容为博主完成本周大数据课程的实验内容。实验内容分为两部分。

1. 在分布式文件系统中创建文件并用shell指令查看；

2. 利用Java API编程实现判断文件是否存在和合并两个文件的内容成一个文件。

感谢厦门大学数据库实验室的实验教程，对博主的学习有很大的帮助。

现在，就让我们一起完成实验内容吧！

创建文件

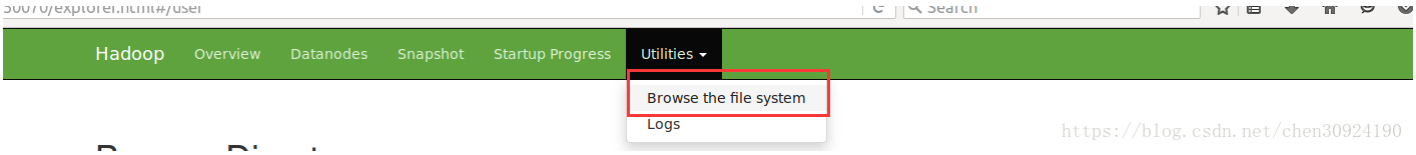
我们需要先启动下Hadoop，【Ctrl】+【Alt】+【t】打开终端，输入一下命令：

```
$ cd /usr/local/hadoop/
$ ./sbin/start-dfs.sh
```

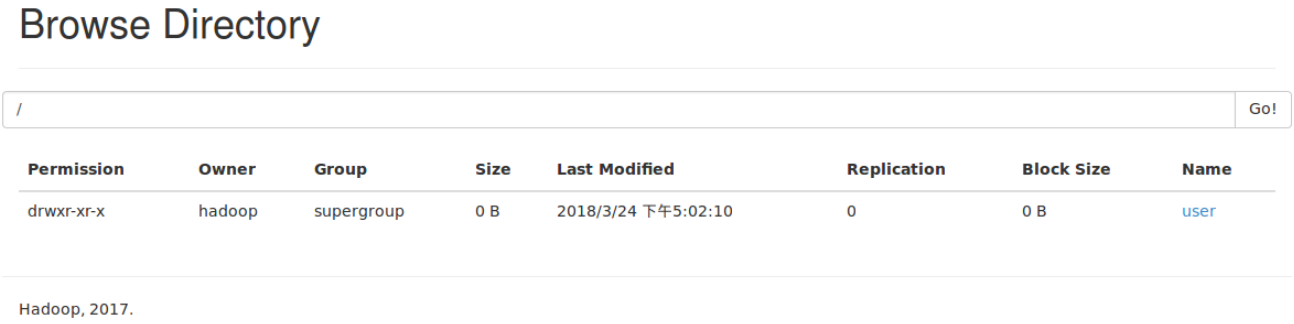
创建一个新用户，在这里我们起名为Hadoop，输入一下指令：

```
./bin/hdfs dfs -mkdir -p /user/hadoop
```

软后在浏览器输入:http://localhost:50070，进入‘Utilities’中的‘Browse the system’：



查看刚才创建的user/haddop，点击uesr，查看：



Browse Directory

/user Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hadoop	supergroup	0 B	2018/3/24 下午11:21:57	0	0 B	hadoop

在本地Linux文件系统的'/home/hadoop/'目录下创建两个文件AFile.txt、Bfile.txt：

```
touch /home/hadoop/AFile.txt
touch /home/hadoop/BFile.txt
```

里面可以随意输入一些单词，比如，第一个输入姓名，第二个输入一串号码：

```
gedit /home/hadoop/AFile.txt
gedit /home/hadoop/BFile.txt
```

然后，可以使用如下命令把本地文件系统的"/home/hadoop/AFile.txt"、"/home/hadoop/BFile.txt"上传到HDFS中的当前用户目录的input目录下，也就是上传到HDFS的"/user/hadoop/"目录下：

```
./bin/hdfs dfs -put /home/hadoop/AFile.txt
./bin/hdfs dfs -put /home/hadoop/BFile.txt
```

查看刚才的两个文件上传是否成功：

```
./bin/hdfs dfs -ls
```

```
hadoop@kevin-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -ls
Found 6 items
-rw-r--r--   1 hadoop supergroup          7 2018-03-24 17:29 AFile.txt
-rw-r--r--   1 hadoop supergroup          5 2018-03-24 17:30 BFile.txt
```


同样，我们也可以查看http://localhost:50070中的'Utilities'中的'Browse the system'：

Browse Directory

/user/hadoop Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	7 B	2018/3/24 下午5:29:24	1	128 MB	AFile.txt
-rw-r--r--	hadoop	supergroup	5 B	2018/3/24 下午5:30:47	1	128 MB	BFile.txt

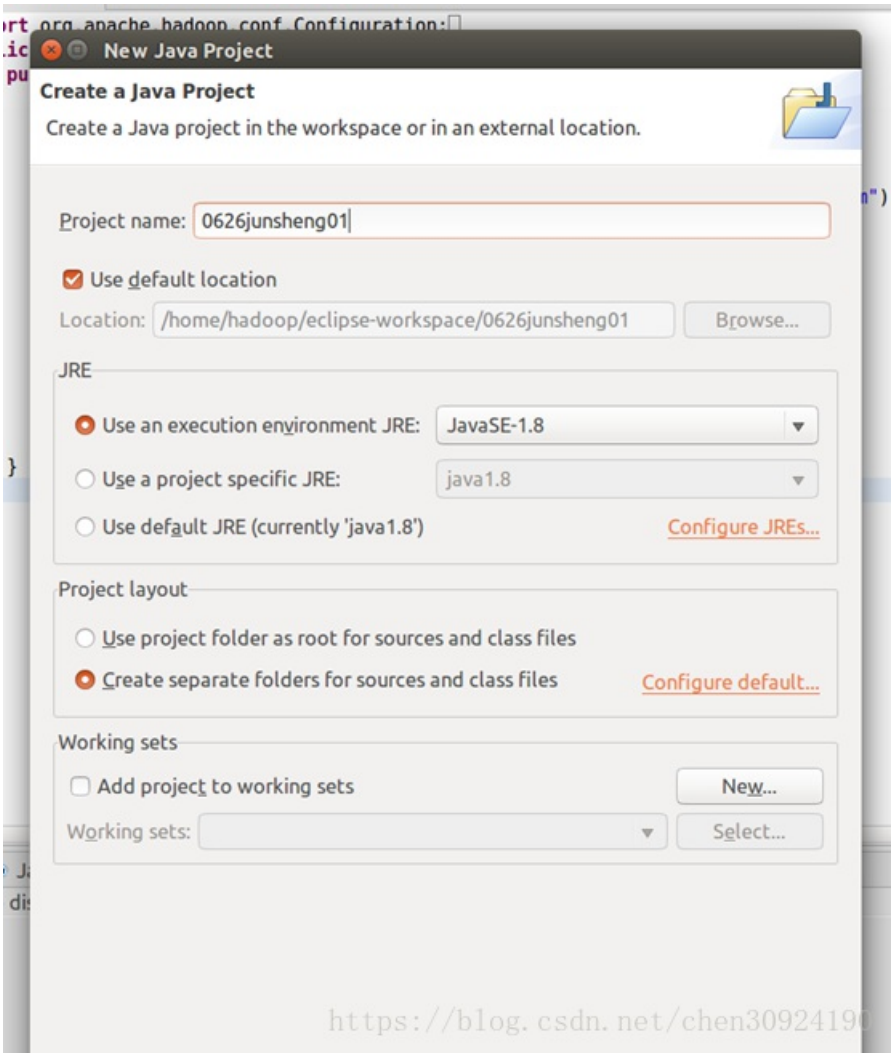
上传成功后我们接下来就是查看HDFS当前用户中所拥有的文件内容：

```
./bin/hdfs dfs -cat AFile.txt
./bin/hdfs dfs -cat BFile.txt
```

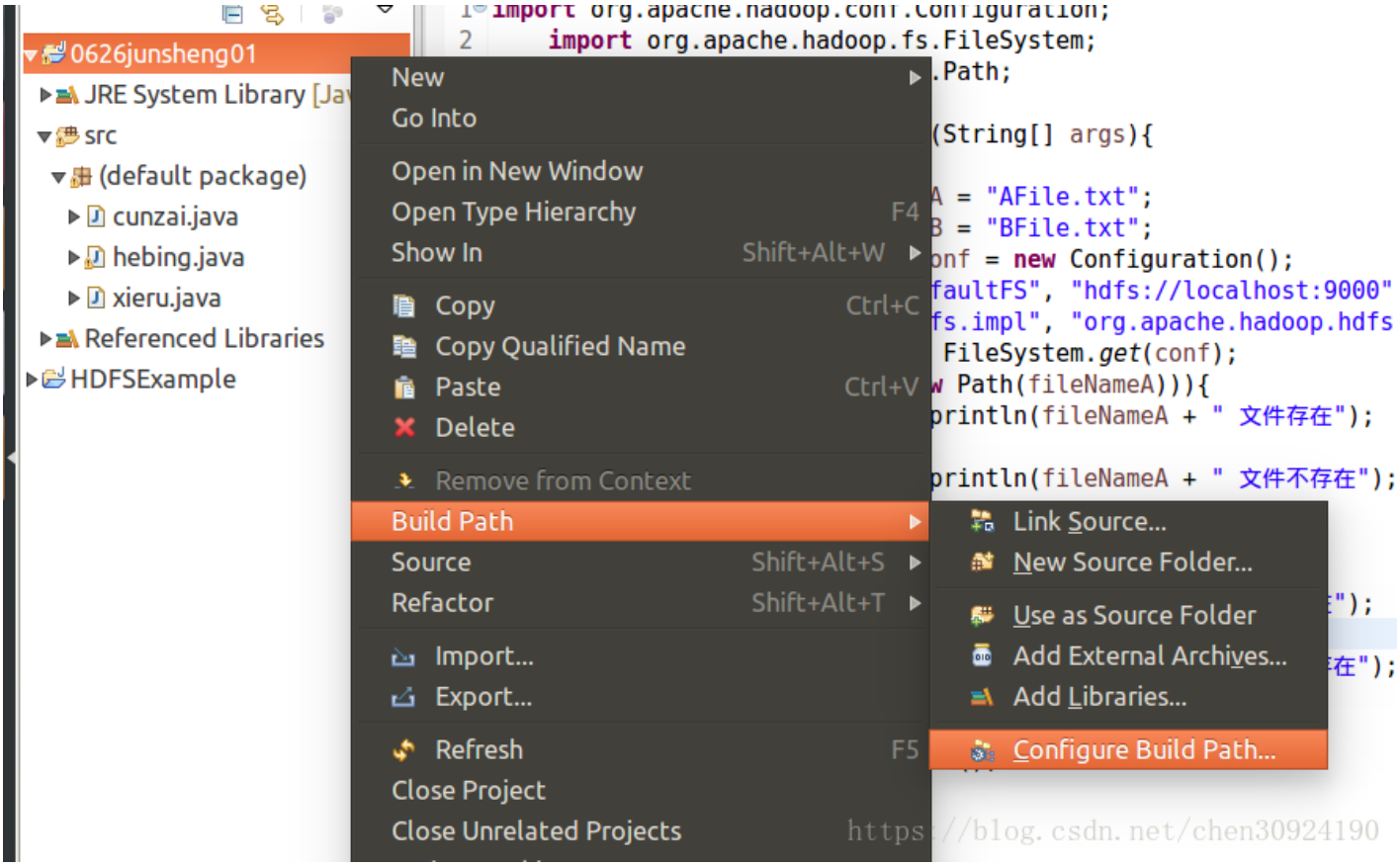
```
hadoop@kevin-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat BFile.txt
0626
hadoop@kevin-virtual-machine:/usr/local/hadoop$ ./bin/hdfs dfs -cat AFile.txt
军生
```

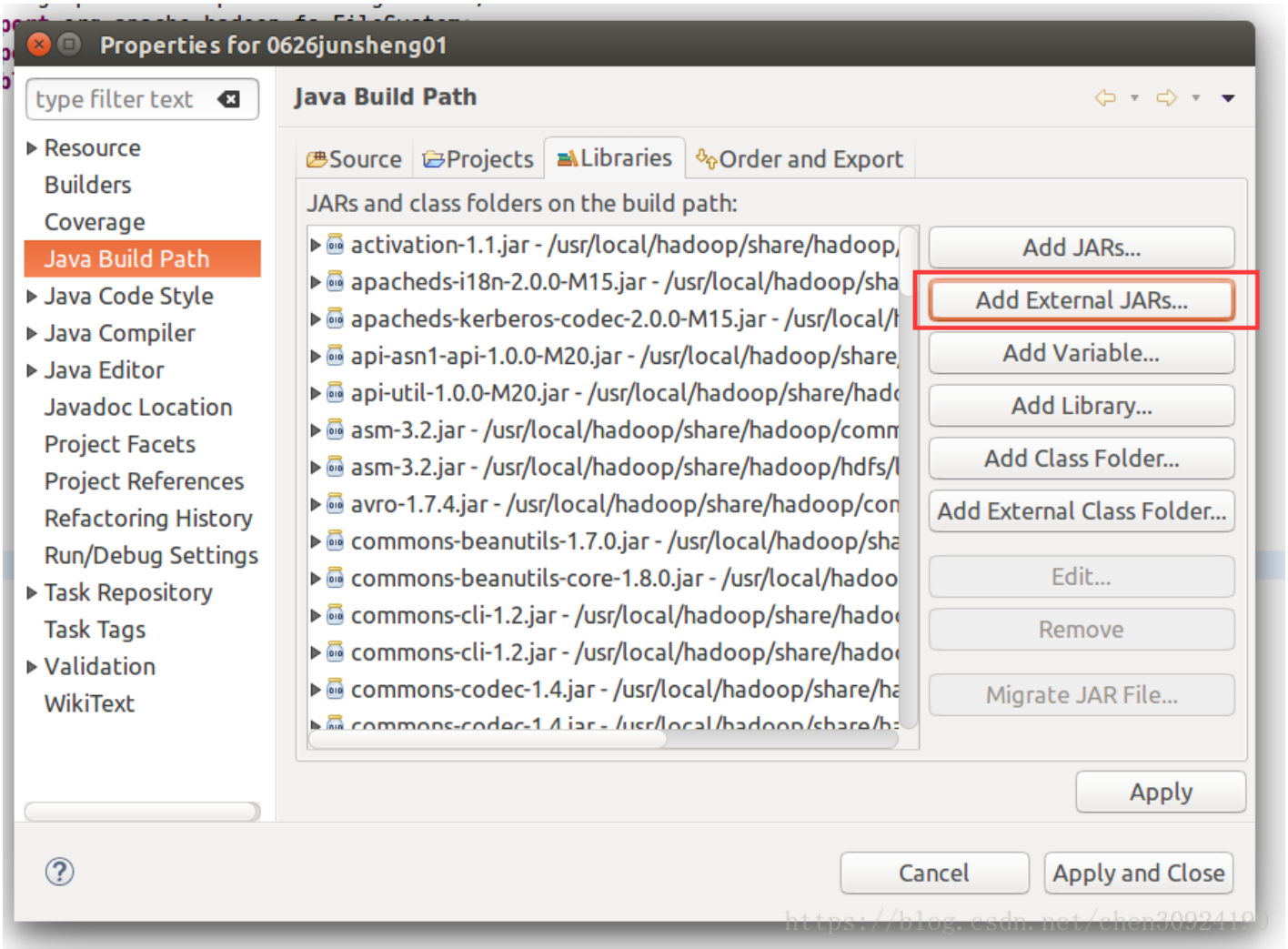

利用Java API编程实现

创建一个java Project，名字自拟。



右键单击项目名称，进入设置，





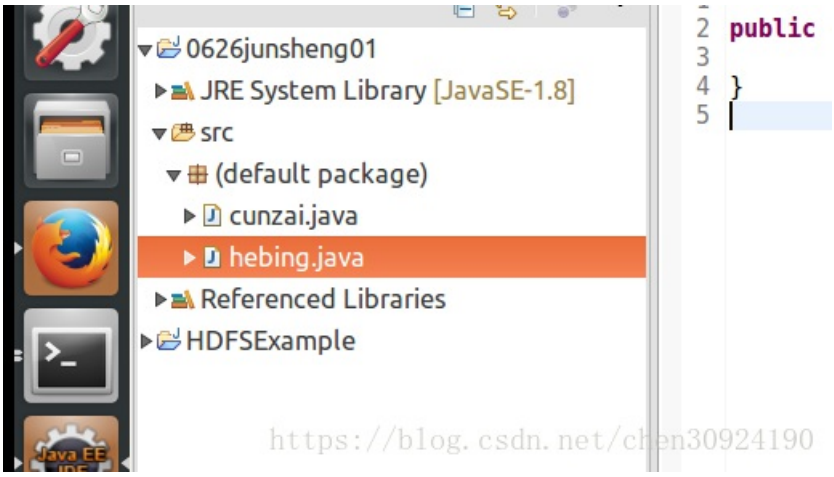
点击进入‘Add External JARs...’根据路径加入以下的JAR包；

为了编写一个能够与HDFS交互的Java应用程序，一般需要向Java工程中添加以下JAR包：

（1）”/usr/local/hadoop/share/hadoop/common”目录下的hadoop-common-2.7.1.jar和haoop-nfs-2.7.1.jar；
（2）/usr/local/hadoop/share/hadoop/common/lib”目录下的所有JAR包；
（3）“/usr/local/hadoop/share/hadoop/hdfs”目录下的haoop-hdfs-2.7.1.jar和haoop-hdfs-nfs-2.7.1.jar；
（4）“/usr/local/hadoop/share/hadoop/hdfs/lib”目录下的所有JAR包。

添加完后应用并退出。

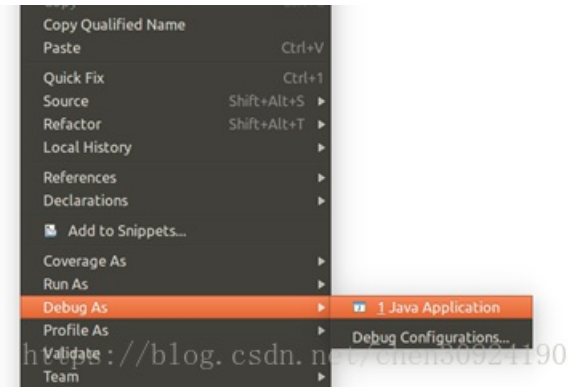
创建两个class：



我们现在做第一个实验，判断文件是否存在：

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
public class cunzai{
    public static void main(String[] args){
        try{
            String fileNameA = "AFile.txt";
            String fileNameB = "BFile.txt";
            Configuration conf = new Configuration();
            conf.set("fs.defaultFS", "hdfs://localhost:9000");
            conf.set("fs.hdfs.impl", "org.apache.hadoop.hdfs.DistributedFileSystem");
            FileSystem fs = FileSystem.get(conf);
            if(fs.exists(new Path(fileNameA))){
                System.out.println(fileNameA + " 文件存在");
            }else{
                System.out.println(fileNameA + " 文件不存在");
            }

            if(fs.exists(new Path(fileNameB))){
                System.out.println(fileNameB + " 文件存在");
            }else{
                System.out.println(fileNameB + " 文件不存在");
            }
        }catch (Exception e){
            e.printStackTrace();
        }
    }

}
```

输入以上代码后，然后右键代码部分空白处，选择Debug As->Java Application

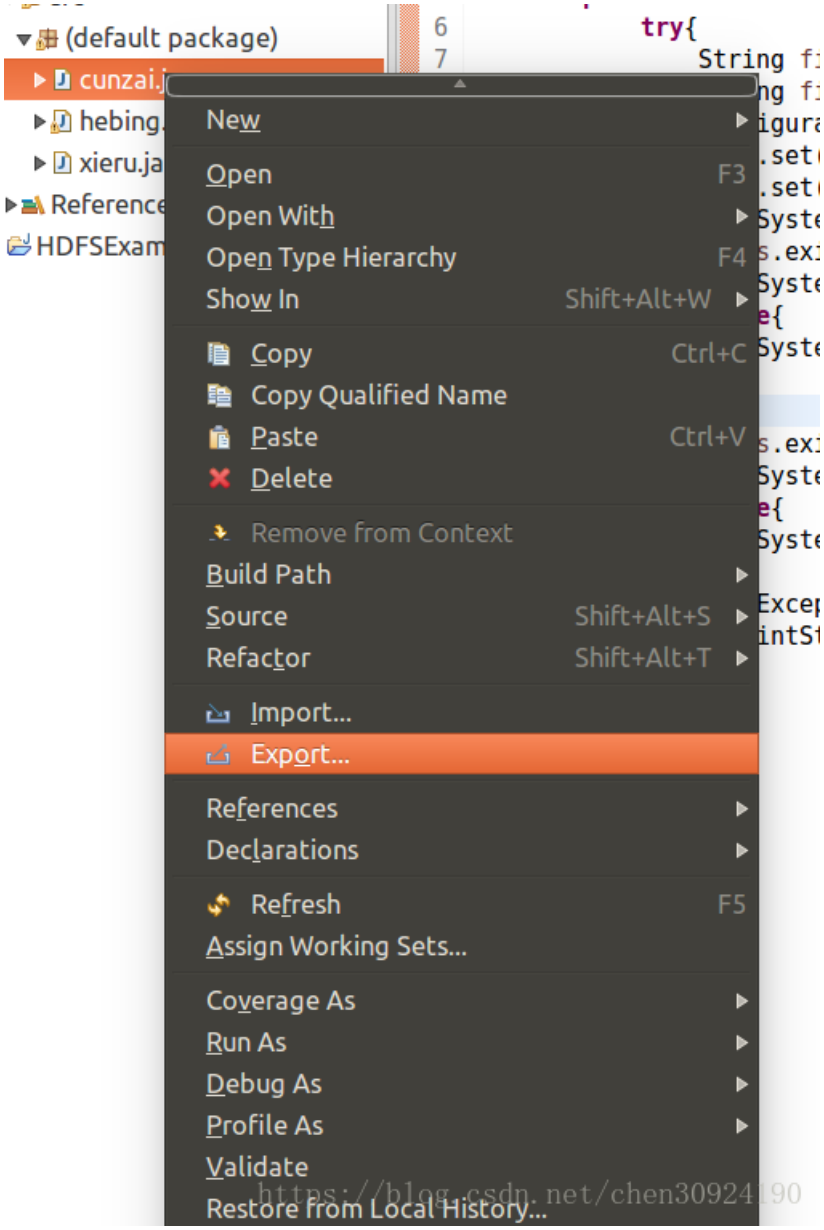


查看结果：

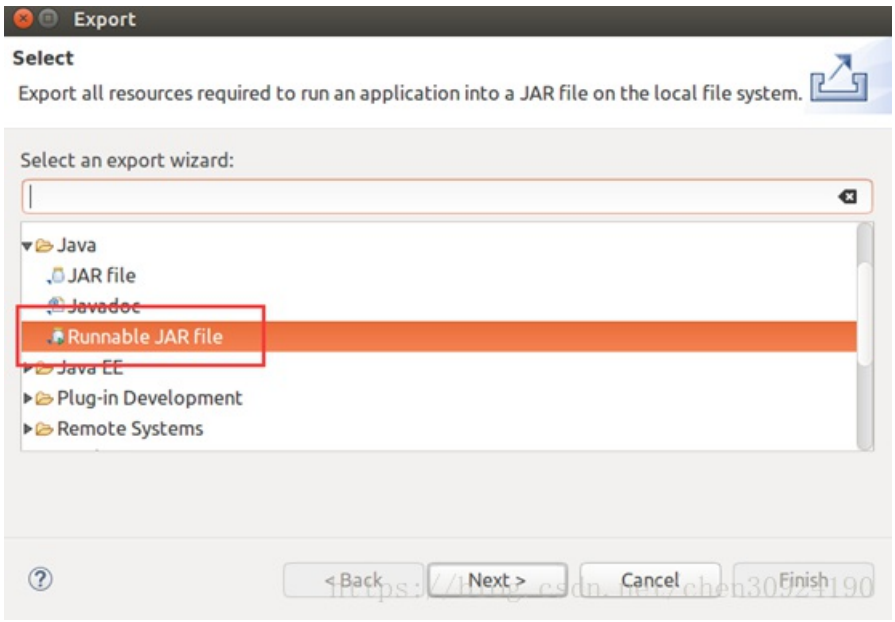
执行成功后，我们下一步将将class文件按一个可执行jar包的格式导入到我们的hadoop路径中，先创建一个路径，在hadoop路径下执行以下命令：

```
mkdir myapp
```

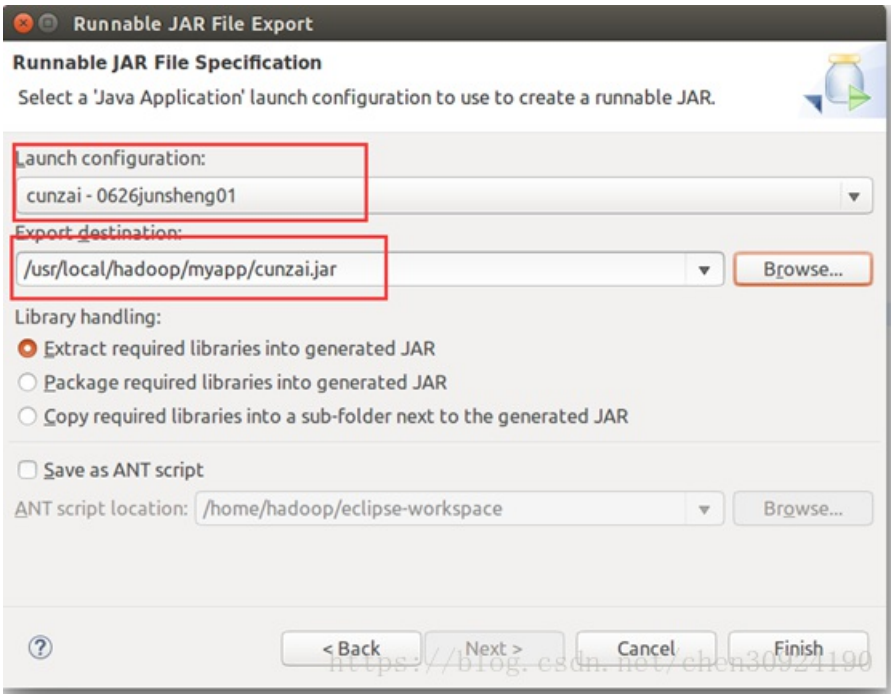
然后，我们回到eclipse中，右键单击名为cunzai的class类：



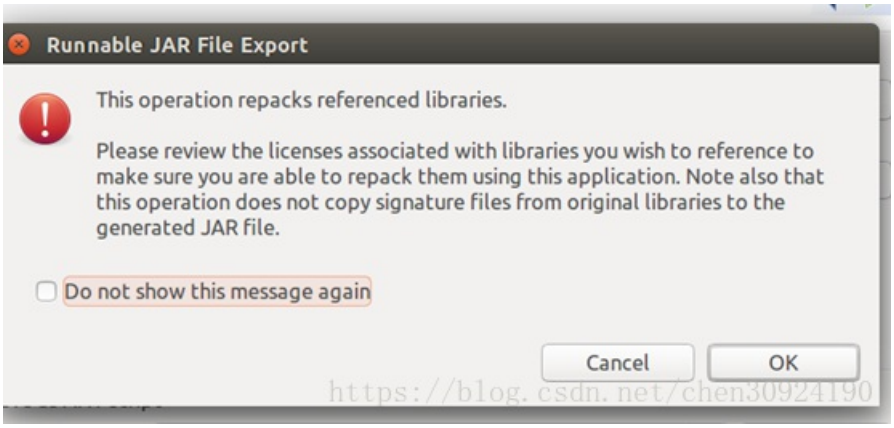
点击export...；选择Runnable JAR file：

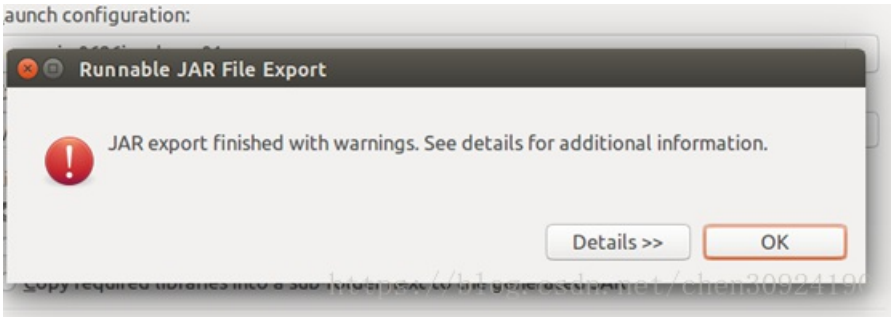


选择class文件名和文件路径：



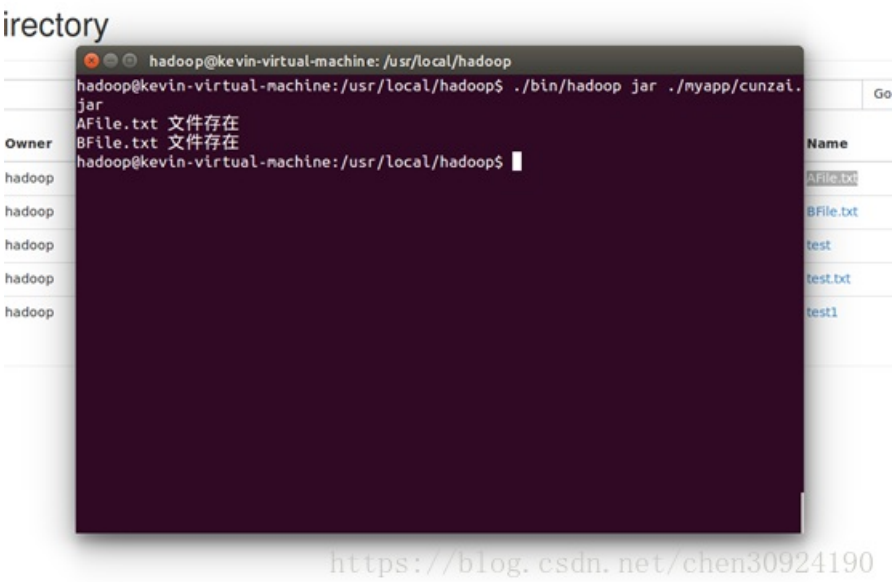
点击ok，ok





我们回到终端检查一下刚才导出的jar包是否可以运行：

```
./bin/hadoop jar ./myapp/cunzai.jar
```



完成jar包读取文件任务。

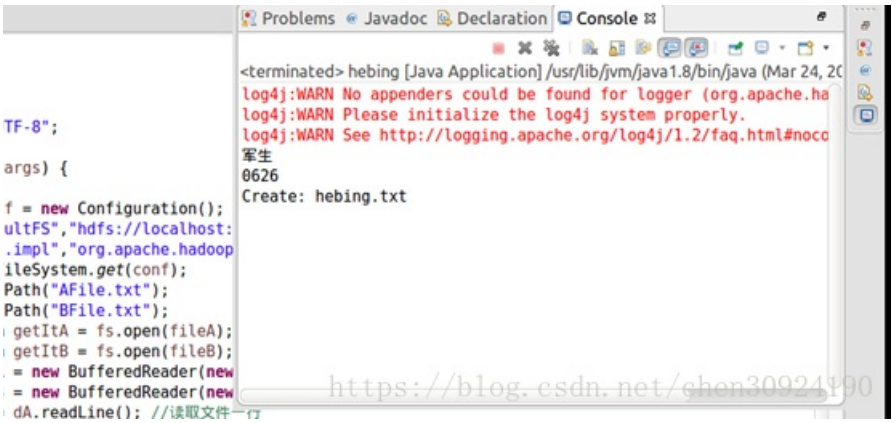
==========

接下来我们来完成合并文件的任务。

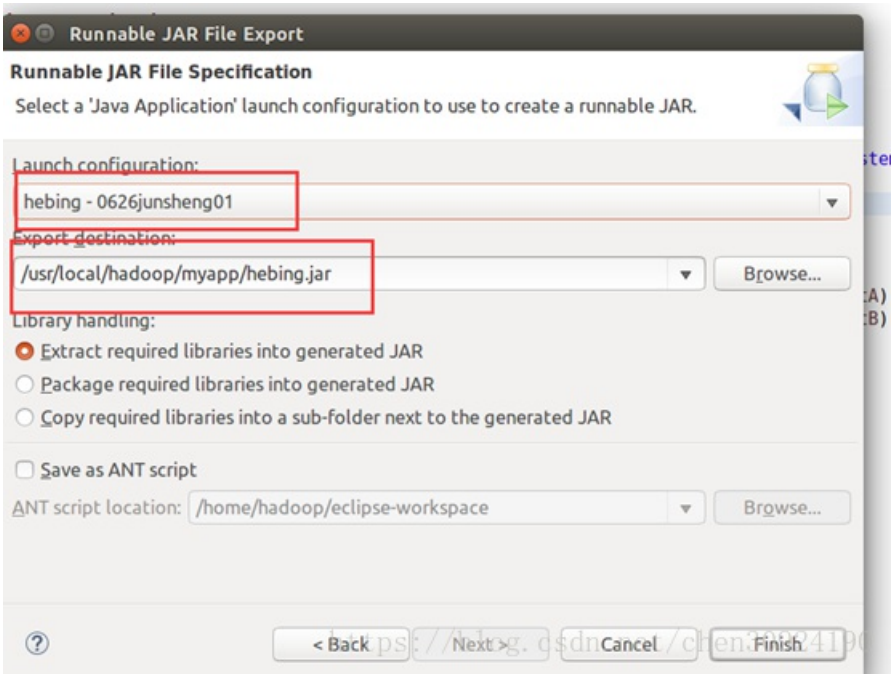
将以下代码复制到hebing.class中：

```
import java.io.BufferedReader;
        import java.io.InputStreamReader;

        import org.apache.hadoop.conf.Configuration;
        import org.apache.hadoop.fs.FileSystem;
        import org.apache.hadoop.fs.Path;
        import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;

        public class hebing {

         private static final String utf8 = "UTF-8";

                public static void main(String[] args) {
                        try {
                                Configuration conf = new Configuration();
                                conf.set("fs.defaultFS","hdfs://localhost:9000");
                                conf.set("fs.hdfs.impl","org.apache.hadoop.hdfs.DistributedFileSystem");
                                FileSystem fs = FileSystem.get(conf);
                                Path fileA = new Path("AFile.txt");
                                Path fileB = new Path("BFile.txt");
                                FSDataInputStream getItA = fs.open(fileA);
                                FSDataInputStream getItB = fs.open(fileB);
                                BufferedReader dA = new BufferedReader(new InputStreamReader(getItA));
                                BufferedReader dB = new BufferedReader(new InputStreamReader(getItB));
                                String contentA = dA.readLine(); //读取文件一行
                             System.out.println(contentA);
                             String contentB = dB.readLine(); //读取文件一行
                             System.out.println(contentB);

                             String mem = new String();
                                mem = contentA+contentB;

                                String filename = "hebing.txt"; //要写入的文件名
                                FSDataOutputStream os = fs.create(new Path(filename));
                                os.writeUTF(mem);
                                System.out.println("Create: "+ filename);


                                os.close(); //关闭文件os
                                dA.close(); //关闭文件A
                                dB.close(); //关闭文件B
                                fs.close(); //关闭hdfs
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        }
```

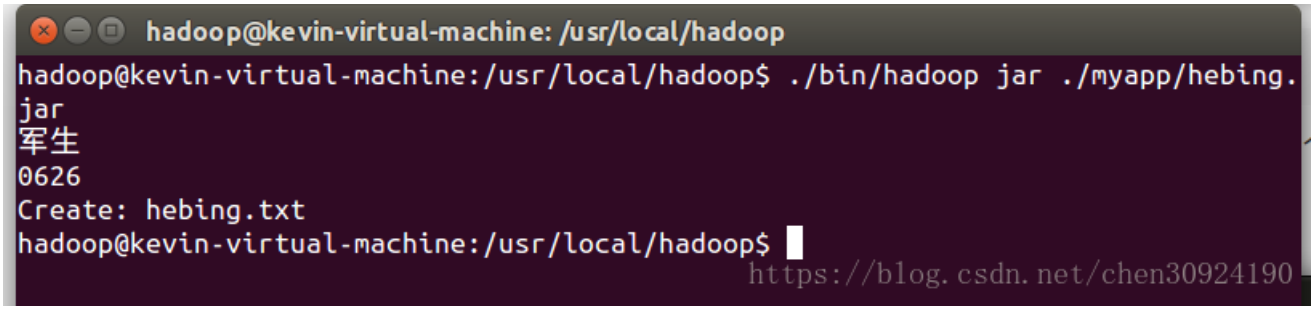
依旧debug运行下，看下结果：



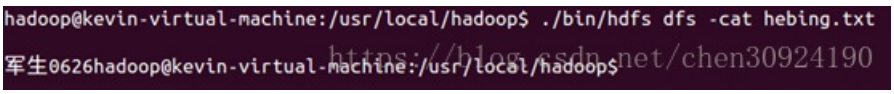
执行成功后我们接着将这个class导出，路径如下：



导出成功后我们在终端执行下指令，运行下刚才导出的jar包：



然后查看一下我们刚才合成的新文件内容：



文件合成成功。

今日实验到此完成，如若对实验内容有异议，请评论留言，若有差池，环境大家批评指正，谢谢各位。