实验名称

使用MapReduce将HDFS文件导入HBase数据库中

实验目的

熟练掌握MapReduce的编程模型

熟练开发基于HBase的MapReduce程序

实验背景

数据存储一般放在文件系统HDFS上，但计算后的数据为了方便查看会写入HBase

实验原理

MapReduce的编程模型，在Map阶段从HDFS中读数据，所以继承Mappper类，将计算结果写入HBase，所以Reduce阶段继承TableReducer

实验环境

ubuntu 22.10
hadoop 3.1.3
jdk 1.8
hbase 2.2.2

建议课时

2课时

实验步骤

1、启动Hadoop环境：

```
start-all.sh
```

获取源数据：

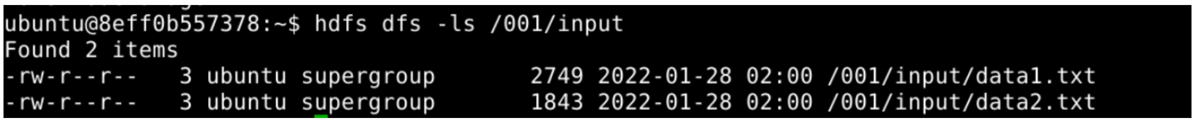
```
wget http://i9000.net:8888/sgn/BDHTech/08TrainingWeek/MR_student/data1.txt
wget http://i9000.net:8888/sgn/BDHTech/08TrainingWeek/MR_student/data2.txt
```

在HDFS上创建目录/{学号}/input，并将需要计算的文件上传至HDFS

```
hdfs dfs -mkdir -p /001/input
hdfs dfs -put ~/data1.txt /001/input
hdfs dfs -put ~/data2.txt /001/input
```

查看文件是否获取成功：

```
hdfs dfs -ls /001/input
```


```
ubuntu@8eff0b557378:~$ hdfs dfs -ls /001/input
Found 2 items
-rw-r--r--   3 ubuntu supergroup       2749 2022-01-28 02:00 /001/input/data1.txt
-rw-r--r--   3 ubuntu supergroup       1843 2022-01-28 02:00 /001/input/data2.txt
```

查询结果如上图所示则表示上传成功

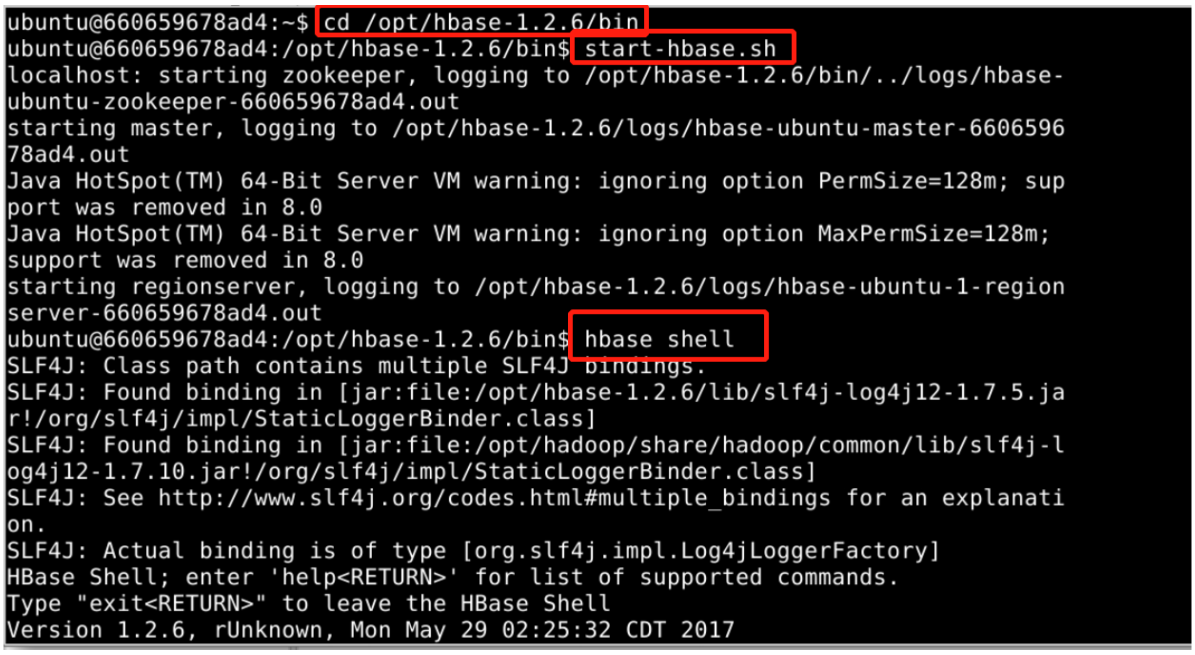
2、启动HBase

另打开一个终端

```
cd /opt/hbase-1.2.6/bin
start-hbase.sh
```

启动HBase Shell终端

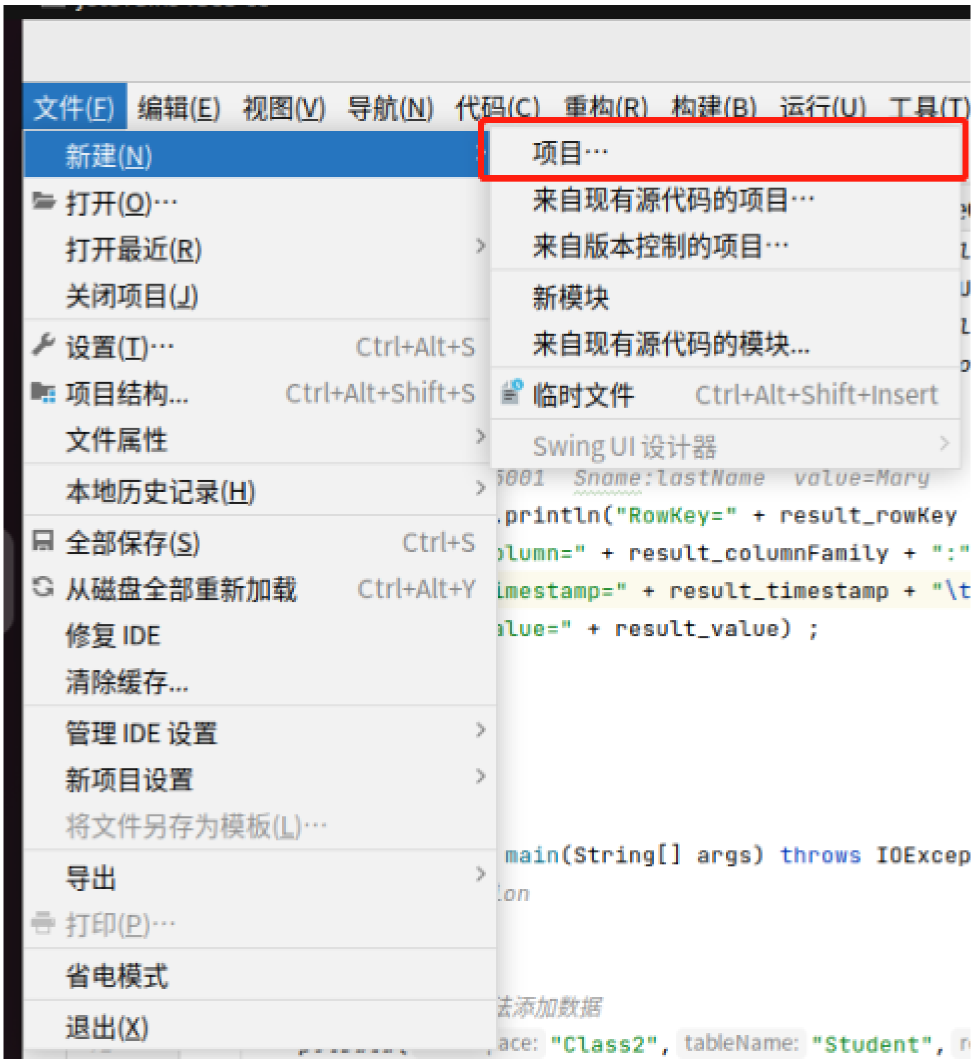
```
hbase shell
```


```
ubuntu@660659678ad4:~$ cd /opt/hbase-1.2.6/bin
ubuntu@660659678ad4:/opt/hbase-1.2.6/bin$ start-hbase.sh
localhost: starting zookeeper, logging to /opt/hbase-1.2.6/bin/../logs/hbase-
ubuntu-zookeeper-660659678ad4.out
starting master, logging to /opt/hbase-1.2.6/logs/hbase-ubuntu-master-6606596
78ad4.out
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option PermSize=128m; sup
port was removed in 8.0
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=128m;
support was removed in 8.0
starting regionserver, logging to /opt/hbase-1.2.6/logs/hbase-ubuntu-1-region
server-660659678ad4.out
ubuntu@660659678ad4:/opt/hbase-1.2.6/bin$ hbase shell
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hbase-1.2.6/lib/slf4j-log4j12-1.7.5.ja
r!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-l
og4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanati
on.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.6, rUnknown, Mon May 29 02:25:32 CDT 2017
```

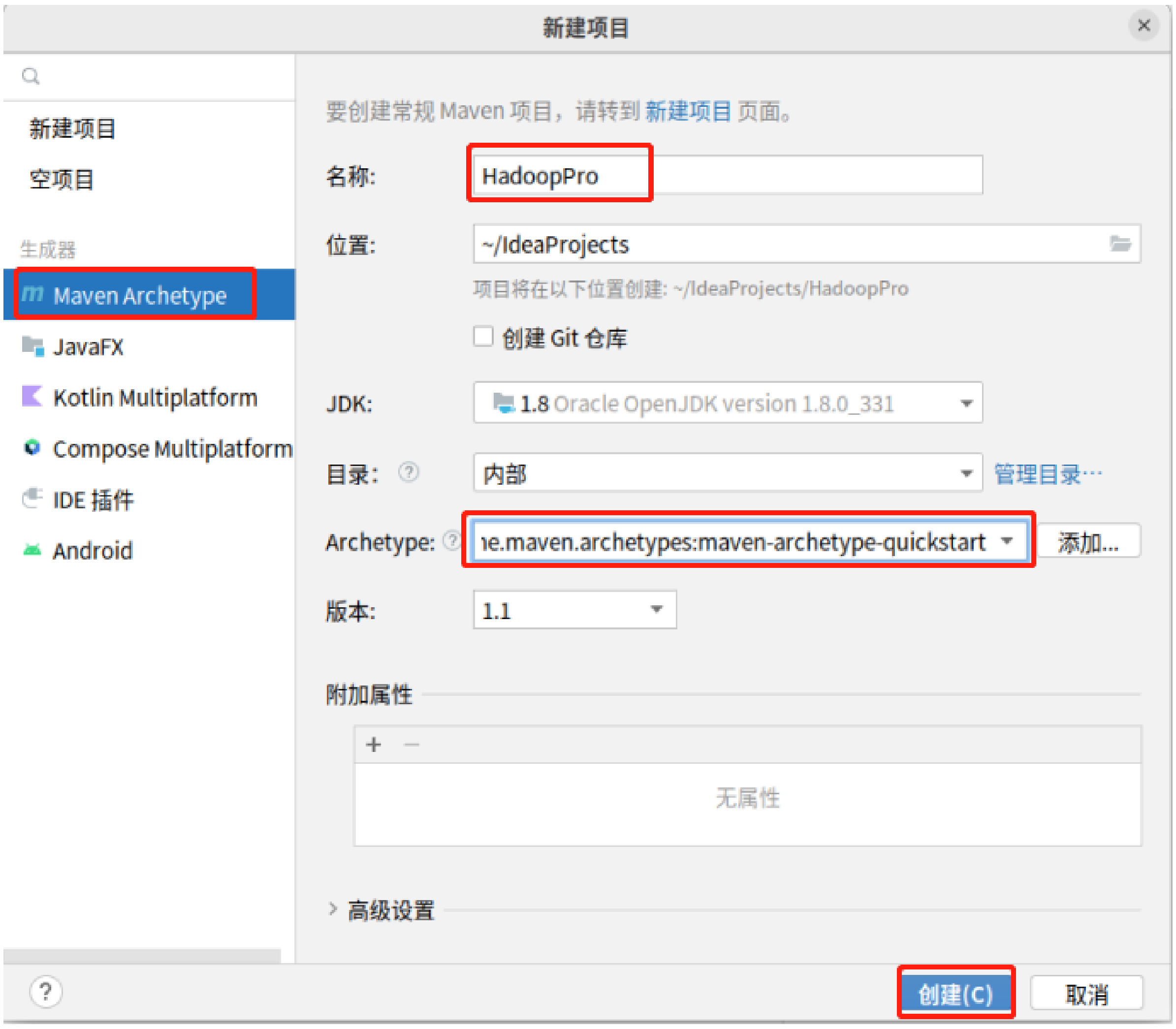
创建mytable表

```
create 'mytable','info'
```

3、打开idea，创建一个Maven project，命名为HadoopPro



```
5001  Sname:lastName  value=Mary
.println("RowKey=" + result_rowKey
olumn=" + result_columnFamily + ":"
imestamp=" + result_timestamp + "\t
alue=" + result_value) ;

 main(String[] args) throws IOExcep
ion

法添加数据
ace: "Class2", tableName: "Student",
```



要创建常规 Maven 项目，请转到 新建项目 页面。

名称: HadoopPro

位置: ~/IdeaProjects

项目将在以下位置创建: ~/IdeaProjects/HadoopPro

创建 Git 仓库

JDK: 1.8 Oracle OpenJDK version 1.8.0_331

目录: 内部

Archetype: ıe.maven.archetypes:maven-archetype-quickstart

版本: 1.1

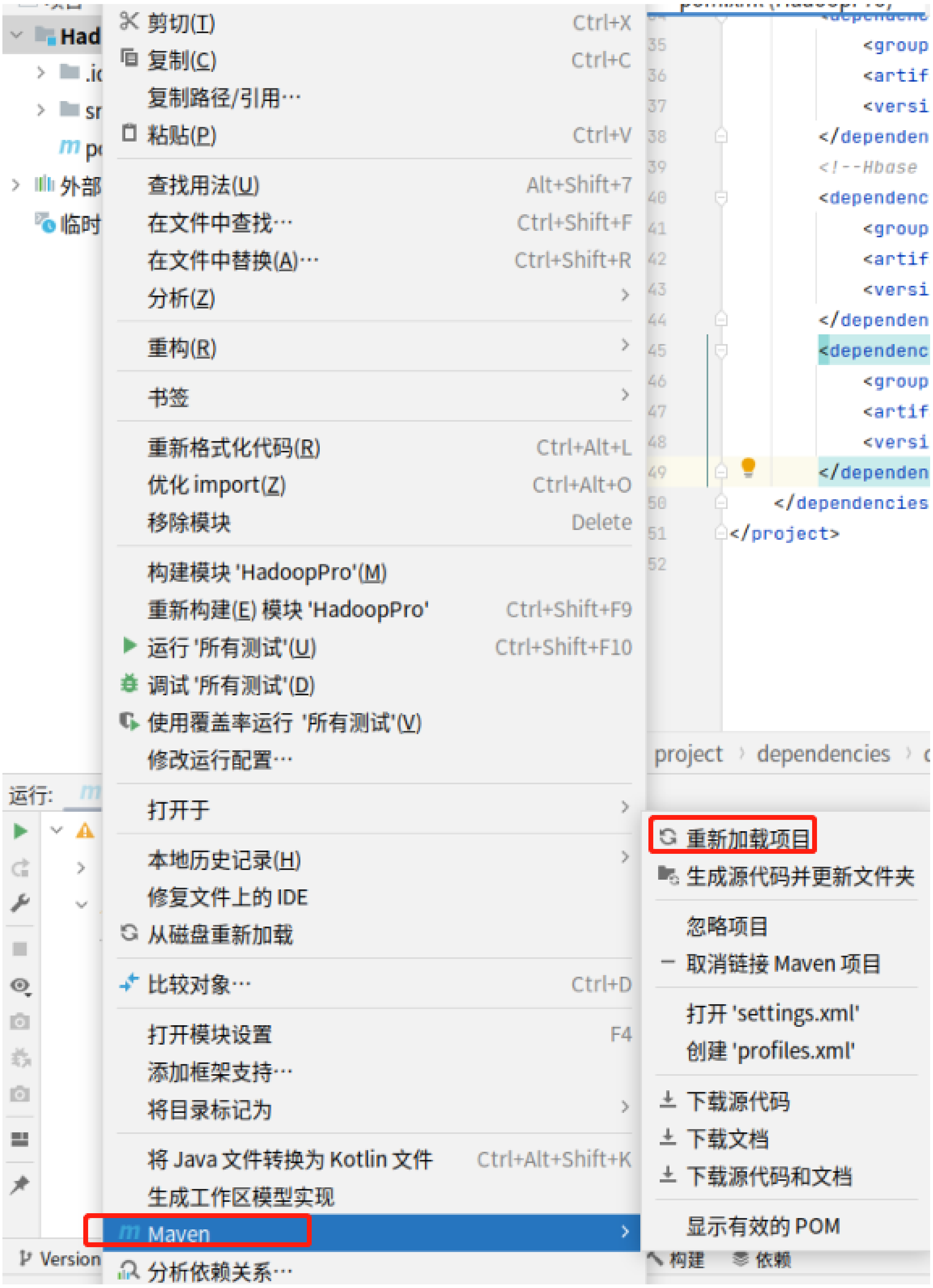
附加属性

创建成功后，在pom.xml中添加Hadoop相关依赖，在之间添加以下依赖

```
<dependencies>
      <!-- Hadoop 依赖-->
      <dependency>
          <groupId>org.apache.hadoop</groupId>
          <artifactId>hadoop-client</artifactId>
          <version>3.1.3</version>
      </dependency>
      <dependency>
          <groupId>org.apache.hadoop</groupId>
          <artifactId>hadoop-common</artifactId>
          <version>3.1.3</version>
      </dependency>
      <dependency>
          <groupId>org.apache.hadoop</groupId>
          <artifactId>hadoop-hdfs</artifactId>
          <version>3.1.3</version>
      </dependency>
      <dependency>
          <groupId>org.apache.hadoop</groupId>
          <artifactId>hadoop-mapreduce-client-core</artifactId>
          <version>3.1.3</version>
      </dependency>
      <!--Hbase 依赖-->
```

```xml
        <dependency>
            <groupId>org.apache.hbase</groupId>
            <artifactId>hbase-client</artifactId>
            <version>2.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hbase</groupId>
            <artifactId>hbase-server</artifactId>
            <version>2.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hbase</groupId>
            <artifactId>hbase-mapreduce</artifactId>
            <version>2.2.2</version>
        </dependency>
    </dependencies>
```

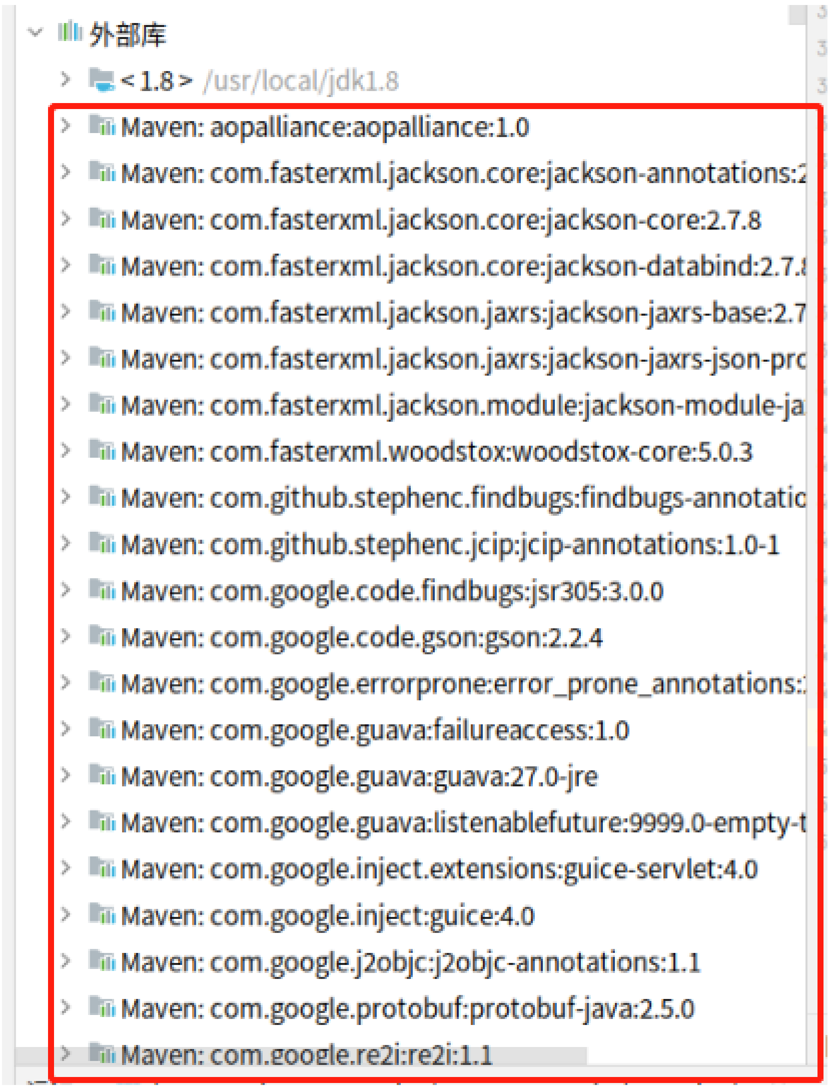
保存后，在工程名称下，右键鼠标，选择重新加载项目



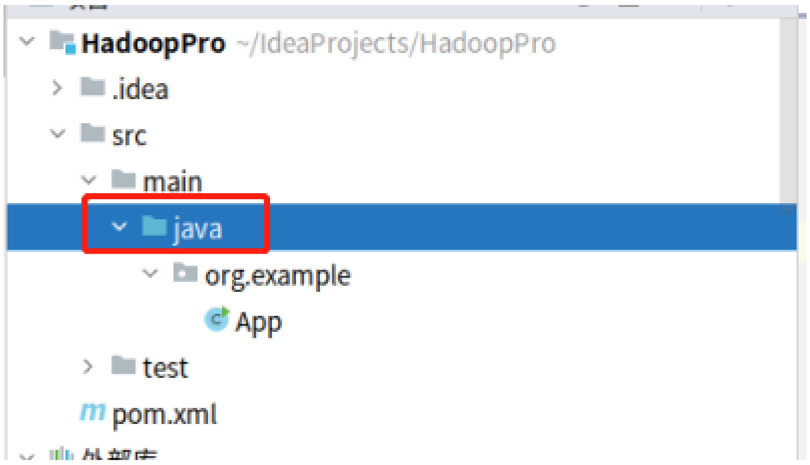
```
            <group
            <artif
            <versi
        </dependen
        <!--Hbase
        <dependenc
            <group
            <artif
            <versi
        </dependen
        <dependenc
            <group
            <artif
            <versi
        </dependen
    </dependencies
</project>
```

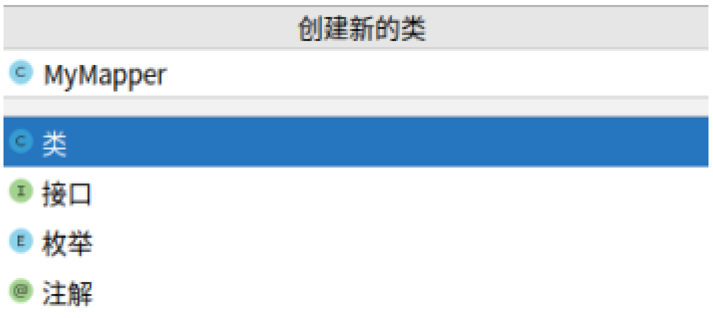
出现上图，说明正在下载

下载完成后，在左边的菜单栏中可以看到


外部库
< 1.8 > /usr/local/jdk1.8
Maven: aopalliance:aopalliance:1.0
Maven: com.fasterxml.jackson.core:jackson-annotations:2
Maven: com.fasterxml.jackson.core:jackson-core:2.7.8
Maven: com.fasterxml.jackson.core:jackson-databind:2.7.8
Maven: com.fasterxml.jackson.jaxrs:jackson-jaxrs-base:2.7
Maven: com.fasterxml.jackson.jaxrs:jackson-jaxrs-json-pro
Maven: com.fasterxml.jackson.module:jackson-module-ja
Maven: com.fasterxml.woodstox:woodstox-core:5.0.3
Maven: com.github.stephenc.findbugs:findbugs-annotatio
Maven: com.github.stephenc.jcip:jcip-annotations:1.0-1
Maven: com.google.code.findbugs:jsr305:3.0.0
Maven: com.google.code.gson:gson:2.2.4
Maven: com.google.errorprone:error_prone_annotations:
Maven: com.google.guava:failureaccess:1.0
Maven: com.google.guava:guava:27.0-jre
Maven: com.google.guava:listenablefuture:9999.0-empty-t
Maven: com.google.inject.extensions:guice-servlet:4.0
Maven: com.google.inject:guice:4.0
Maven: com.google.j2objc:j2objc-annotations:1.1
Maven: com.google.protobuf:protobuf-java:2.5.0
Maven: com.google.re2j:re2j:1.1

在项目中/src/main/java路径中新建类MyMapper

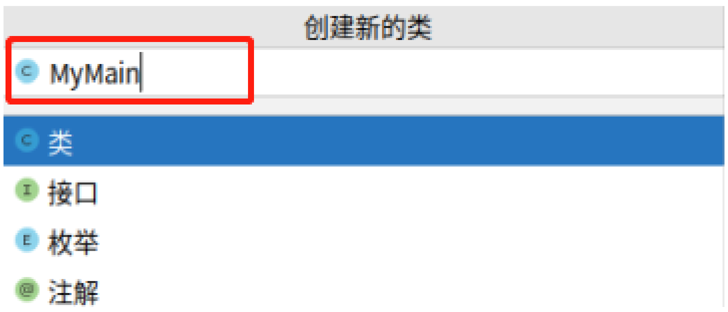

HadoopPro ~/IdeaProjects/HadoopPro
.idea
src
main
java
org.example
App
test
pom.xml


创建新的类
MyMapper
类
接口
枚举
注解

完整代码如下：

```
import java.io.IOException;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MyMapper extends
Mapper<LongWritable,Text,ImmutableBytesWritable,Put>{

    //定义主键类型
    ImmutableBytesWritable rowkey = new ImmutableBytesWritable();
    @Override
    protected void map(LongWritable key, Text value,
                       Mapper<LongWritable, Text, ImmutableBytesWritable,
Put>.Context context)
            throws IOException, InterruptedException {
        //1. 根据分割副切分数据
        String[] data =value.toString().split("\\s+");
        //获取rowkey
        rowkey.set(Bytes.toBytes(data[1]));

        //创建put对象
        Put put = new Put(Bytes.toBytes(data[1]));
        //设置添加列与对应的列值
        put.addColumn(Bytes.toBytes("info"),Bytes.toBytes("name"),
Bytes.toBytes(data[2]));
        put.addColumn(Bytes.toBytes("info"),Bytes.toBytes("sex"),
Bytes.toBytes(data[3]));
        put.addColumn(Bytes.toBytes("info"),Bytes.toBytes("course"),
Bytes.toBytes(data[4]));
        put.addColumn(Bytes.toBytes("info"),Bytes.toBytes("clas"),
Bytes.toBytes(data[5]));

        //输出到reduce端
        context.write(rowkey, put);

    }

}
```

接下来在项目中/src/main/java路径中新建类MyMain，并编写main()



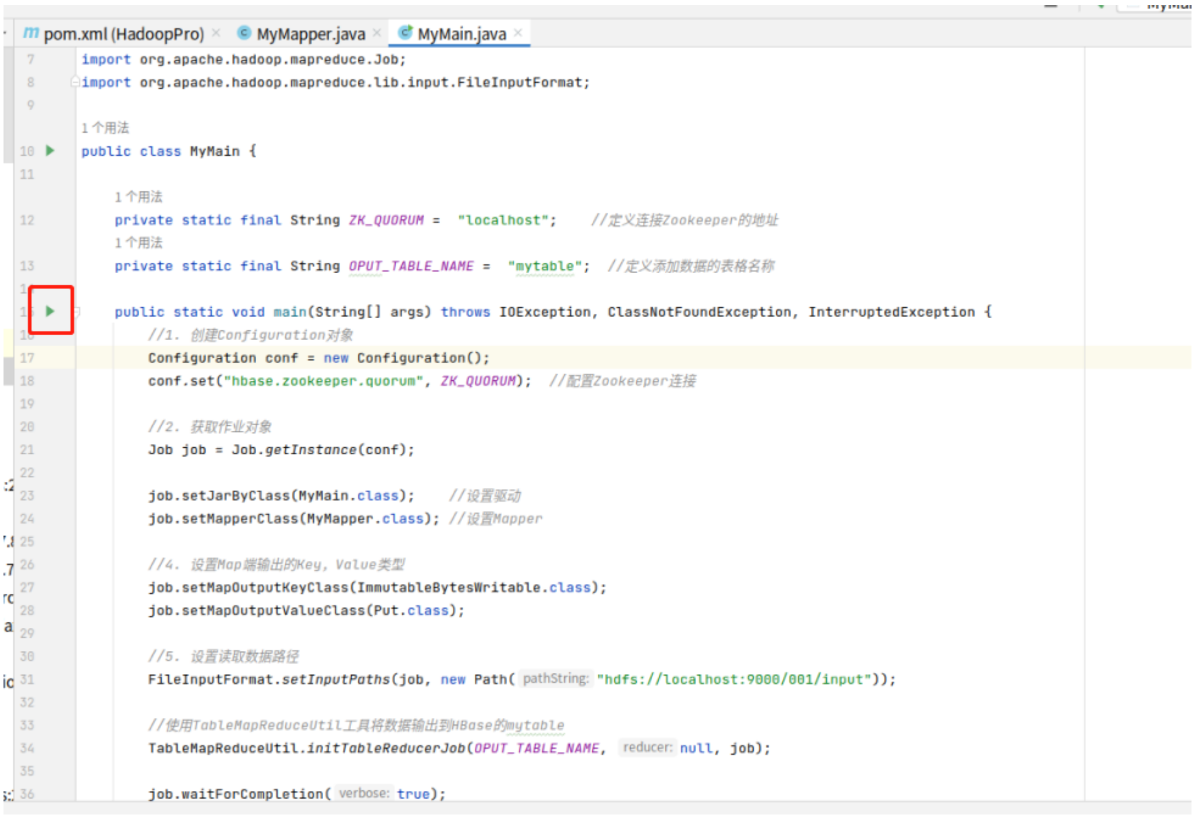
完整代码如下：

```
import java.io.IOException;
```

```java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

public class MyMain {

    private static final String ZK_QUORUM =  "localhost";    //定义连接Zookeeper的
地址
    private static final String OPUT_TABLE_NAME =  "mytable";  //定义添加数据的表格
名称

    public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
        //1. 创建Configuration对象
        Configuration conf = new Configuration();
        conf.set("hbase.zookeeper.quorum", ZK_QUORUM);  //配置Zookeeper连接

        //2. 获取作业对象
        Job job = Job.getInstance(conf);

        job.setJarByClass(MyMain.class);    //设置驱动
        job.setMapperClass(MyMapper.class); //设置Mapper

        //4. 设置Map端输出的Key，Value类型
        job.setMapOutputKeyClass(ImmutableBytesWritable.class);
        job.setMapOutputValueClass(Put.class);

        //5. 设置读取数据路径
        FileInputFormat.setInputPaths(job, new
Path("hdfs://localhost:9000/001/input"));

        //使用TableMapReduceUtil工具将数据输出到HBase的mytable
        TableMapReduceUtil.initTableReducerJob(OPUT_TABLE_NAME, null, job);

        job.waitForCompletion(true);

    }

}
```

至此代码编写完成。

执行main()

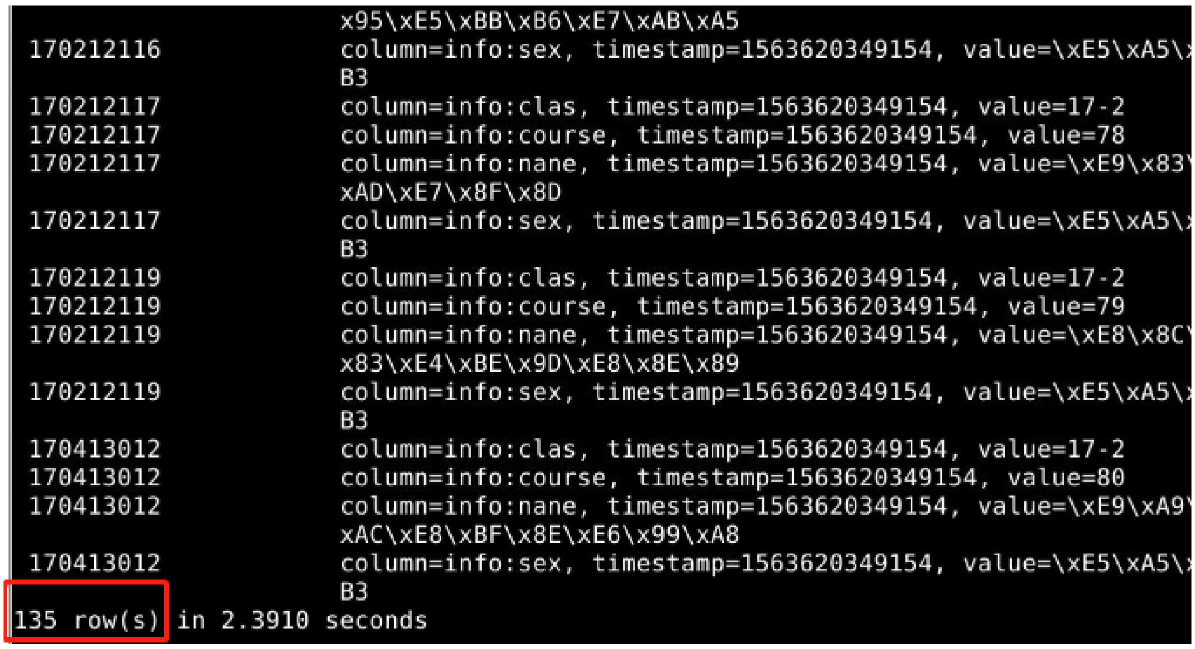


pom.xml (HadoopPro) × MyMapper.java × MyMain.java ×

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

1个用法
public class MyMain {

    1个用法
    private static final String ZK_QUORUM =  "localhost";    //定义连接Zookeeper的地址
    1个用法
    private static final String OPUT_TABLE_NAME =  "mytable";  //定义添加数据的表格名称

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        //1. 创建Configuration对象
        Configuration conf = new Configuration();
        conf.set("hbase.zookeeper.quorum", ZK_QUORUM);  //配置Zookeeper连接

        //2. 获取作业对象
        Job job = Job.getInstance(conf);

        job.setJarByClass(MyMain.class);    //设置驱动
        job.setMapperClass(MyMapper.class); //设置Mapper

        //4. 设置Map端输出的Key, Value类型
        job.setMapOutputKeyClass(ImmutableBytesWritable.class);
        job.setMapOutputValueClass(Put.class);

        //5. 设置读取数据路径
        FileInputFormat.setInputPaths(job, new Path( pathString: "hdfs://localhost:9000/001/input"));

        //使用TableMapReduceUtil工具将数据输出到HBase的mytable
        TableMapReduceUtil.initTableReducerJob(OPUT_TABLE_NAME,  reducer: null, job);

        job.waitForCompletion( verbose: true);
```

然后通过hbase shell查看是否导入成功

```
scan 'mytable'
```



```
                        x95\xE5\xBB\xB6\xE7\xAB\xA5
 170212116              column=info:sex, timestamp=1563620349154, value=\xE5\xA5\x
                        B3
 170212117              column=info:clas, timestamp=1563620349154, value=17-2
 170212117              column=info:course, timestamp=1563620349154, value=78
 170212117              column=info:nane, timestamp=1563620349154, value=\xE9\x83\
                        xAD\xE7\x8F\x8D
 170212117              column=info:sex, timestamp=1563620349154, value=\xE5\xA5\x
                        B3
 170212119              column=info:clas, timestamp=1563620349154, value=17-2
 170212119              column=info:course, timestamp=1563620349154, value=79
 170212119              column=info:nane, timestamp=1563620349154, value=\xE8\x8C\
                        x83\xE4\xBE\x9D\xE8\x8E\x89
 170212119              column=info:sex, timestamp=1563620349154, value=\xE5\xA5\x
                        B3
 170413012              column=info:clas, timestamp=1563620349154, value=17-2
 170413012              column=info:course, timestamp=1563620349154, value=80
 170413012              column=info:nane, timestamp=1563620349154, value=\xE9\xA9\
                        xAC\xE8\xBF\x8E\xE6\x99\xA8
 170413012              column=info:sex, timestamp=1563620349154, value=\xE5\xA5\x
                        B3
135 row(s) in 2.3910 seconds
```

到此实验结束

实验总结

以上我们介绍了如何使用mapreduce将HDFS上的文件导入至HBase，首先我们先将文件上传至HDFS，MapReduce分为两个阶段，Map和Reduce阶段，其中Map阶段是映射，Reduce阶段是聚合，因为该实验没有聚合计算，所以reduce函数我们可以不重写，故该实验只需要重写map()即可。