

1 实验目标

1.1 实验背景介绍

本实验为课程《ACA21107 深入理解阿里云产品-Data IDE》中的配套实验。主要阐述如何使用 ODPS 的 Eclipse 插件和 Data IDE 来完成 ODPS MR 的开发，学员可以根据本实验手册，去学习 ODPS MR 的开发流程以及实现一个 wordcount 实例：统计一张表中单词出现的次数。

1.2 实验环境架构

实验环境架构：阿里云开放数据处理服务 ODPS

1.3 云端服务环境

暂无

1.4 云端开发桌面环境

2 实验内容

2.1 总体流程

安装配置环境→开发 MR 程序→本地模式测试脚本→导出 jar 包→上传资源
→配置 ODPS MR 节点。

2.2 配置环境并本地模式测试 MR

2.2.1 配置 Java 编程环境

step1：官网导航中找到并下载 ODPS for eclipse 插件。

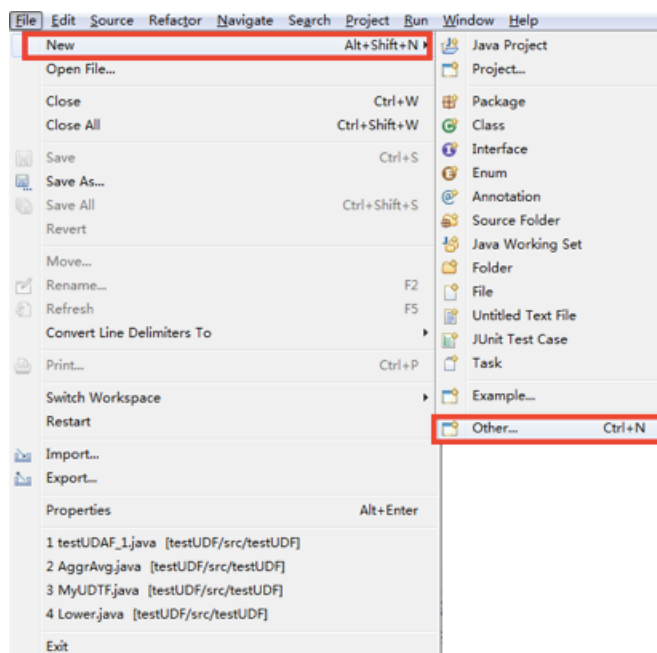
(https://help.aliyun.com/document_detail/odps/download/download.html)

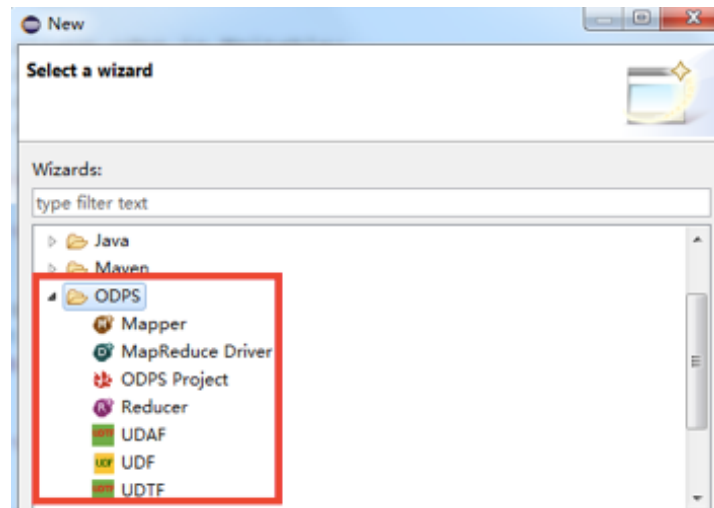
step2：将插件解压并复制到 eclipse 安装目录下的 plug-in 子目录下。

step3：启动 eclipse。

step3：检查 Wizard 选项里面是否有 ODPS 的目录。

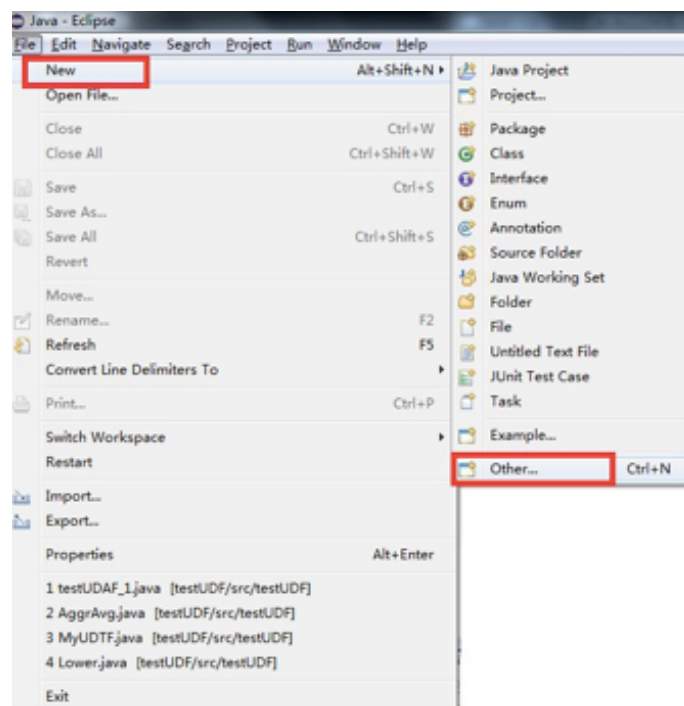
点击 New>Other...在 Wizard 弹出框中有 ODPS 文件夹选项，则表示安装成功。

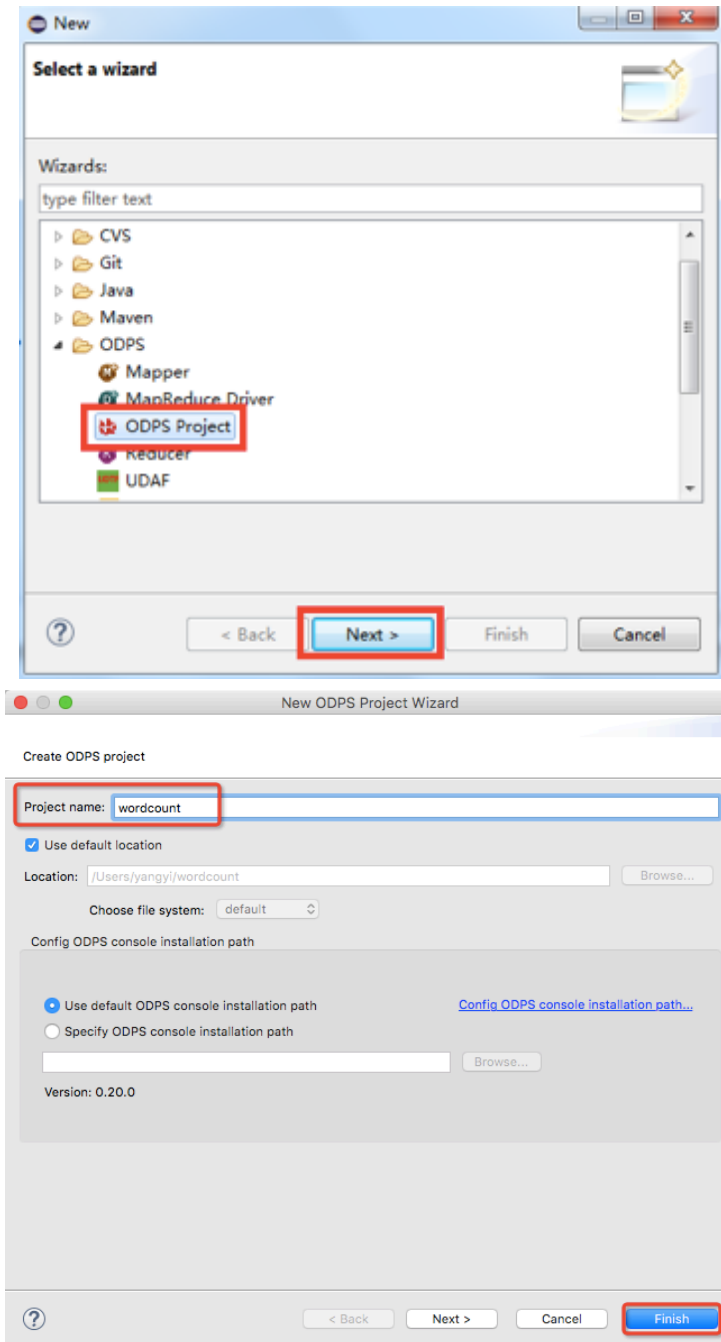




2.2.2 新建项目

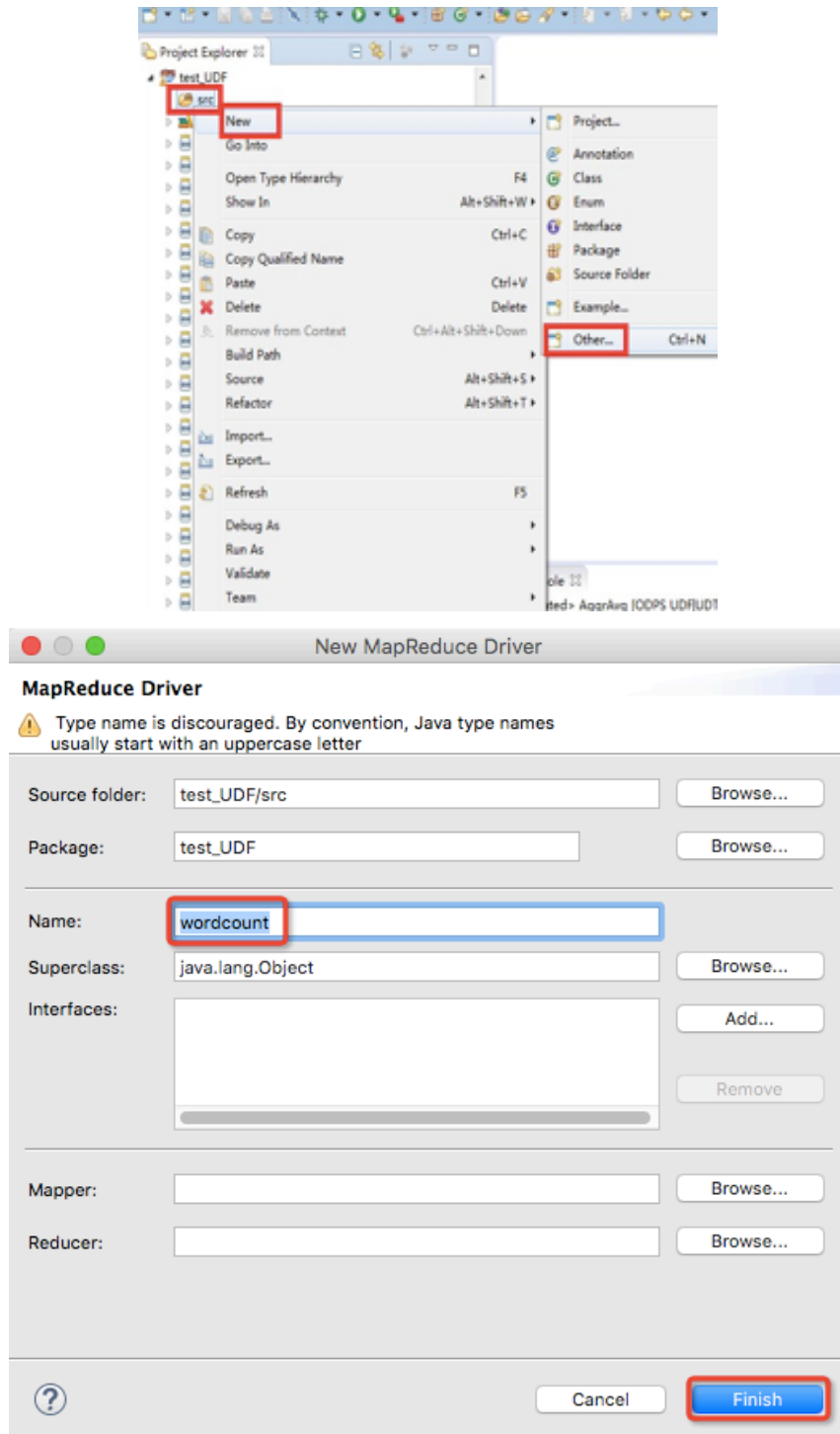
step1 : 点击 **New>Other...>ODPS Project** , 点击 **Next>**。在对话框中输入 Project name , 此处为 wordcount , 再点击 **Finish**。





2.2.3 新增 MapReduce 类

点击新建的 test_UDF 项目下的 src 文件夹，右键 new>Other...，然后在 New MapReduce Driver 弹出框 Name 选项框中输入类名，该示例为 wordcount。



2.2.4 添加逻辑处理

代码示例如下：

```
package wordcount;
```

```
import java.io.IOException;
import java.util.Iterator;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.MapperBase;
import com.aliyun.odps.mapred.ReducerBase;
import com.aliyun.odps.mapred.TaskContext;
import com.aliyun.odps.mapred.conf.JobConf;
import com.aliyun.odps.mapred.utils.InputUtils;
import com.aliyun.odps.mapred.utils.OutputUtils;
import com.aliyun.odps.mapred.utils.SchemaUtils;

public class WordCount {

    public static class TokenizerMapper extends MapperBase {
        private Record word;
        private Record one;

        @Override
        public void setup(TaskContext context) throws IOException {
            word = context.createMapOutputKeyRecord();
            one = context.createMapOutputValueRecord();
            one.set(new Object[] { 1L });
            System.out.println("TaskID:" +
context.getTaskID().toString());
        }

        @Override
        public void map(long recordNum, Record record, TaskContext
context)
            throws IOException {
```

```

        for (int i = 0; i < record.getColumnCount(); i++) {
            word.set(new Object[] { record.get(i).toString() });
            context.write(word, one);
        }
    }
}

/**
 * A combiner class that combines map output by sum them.
 */
public static class SumCombiner extends ReducerBase {
    private Record count;

    @Override
    public void setup(TaskContext context) throws IOException {
        count = context.createMapOutputValueRecord();
    }

    @Override
    public void reduce(Record key, Iterator<Record> values,
TaskContext context)
        throws IOException {
        long c = 0;
        while (values.hasNext()) {
            Record val = values.next();
            c += (Long) val.get(0);
        }
        count.set(0, c);
        context.write(key, count);
    }
}

/**

```

```

    * A reducer class that just emits the sum of the input values.
    **/

public static class SumReducer extends ReducerBase {
    private Record result = null;

    @Override
    public void setup(TaskContext context) throws IOException {
        result = context.createOutputRecord();
    }

    @Override
    public void reduce(Record key, Iterator<Record> values,
TaskContext context)
        throws IOException {
        long count = 0;
        while (values.hasNext()) {
            Record val = values.next();
            count += (Long) val.get(0);
        }
        result.set(0, key.get(0));
        result.set(1, count);
        context.write(result);
    }
}

public static void main(String[] args) throws Exception {
    if (args.length != 2) {
        System.err.println("Usage: WordCount <in_table>
<out_table>");
        System.exit(2);
    }

    JobConf job = new JobConf();

```



```
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(SumCombiner.class);
    job.setReducerClass(SumReducer.class);

    job.setMapOutputKeySchema(SchemaUtils.fromString("word:string"
    ));

    job.setMapOutputValueSchema(SchemaUtils.fromString("count:bigi
    nt"));

    InputUtils.addTable(TableInfo.builder().tableName(args[0]).bui
    ld(), job);

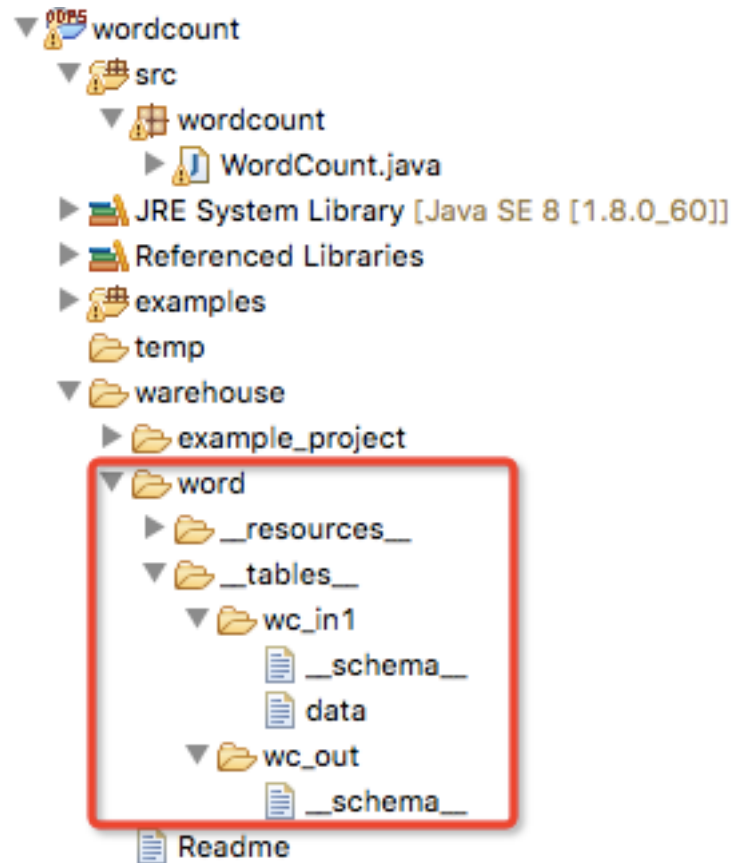
    OutputUtils.addTable(TableInfo.builder().tableName(args[1]).bu
    ild(), job);

    JobClient.runJob(job);
}
}
```

2.2.5 准备本地测试数据

找到wordcount项目下的warehouse按照如下方式准备本地测试数据 如下：

step1：在_schema_文件中编写如下内容。



wc_in1 的_schema_内容：

```
project=word  
  
table=wc_in1  
  
columns=col1:STRING,col2:STRING
```

step2：在 wc_in1 的 data 文件中输入测试数据，示例如下。

```
A1,A2  
A1,A2  
A1,A2  
A1,A2
```

step3：wc_out 的_schema_。

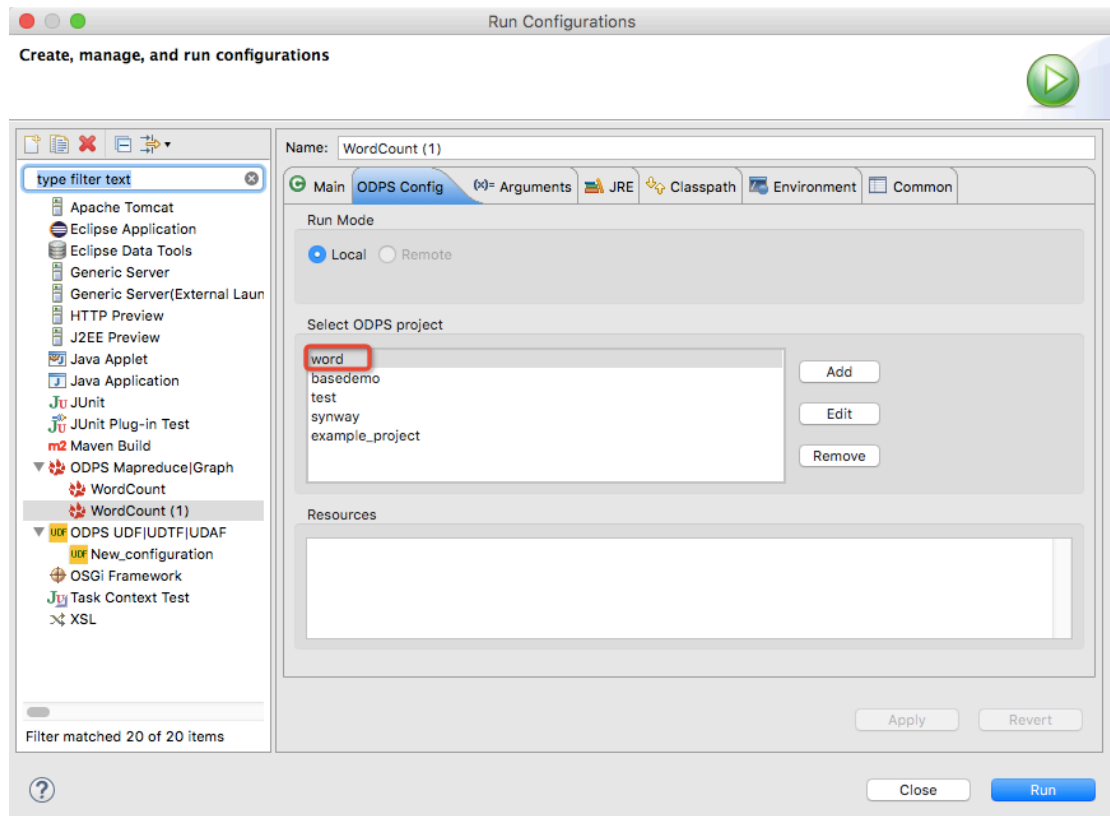
```
project=word  
  
table=wc_out
```

```
columns=word:STRING,cnt:BIGINT
```

2.2.6 编译调试

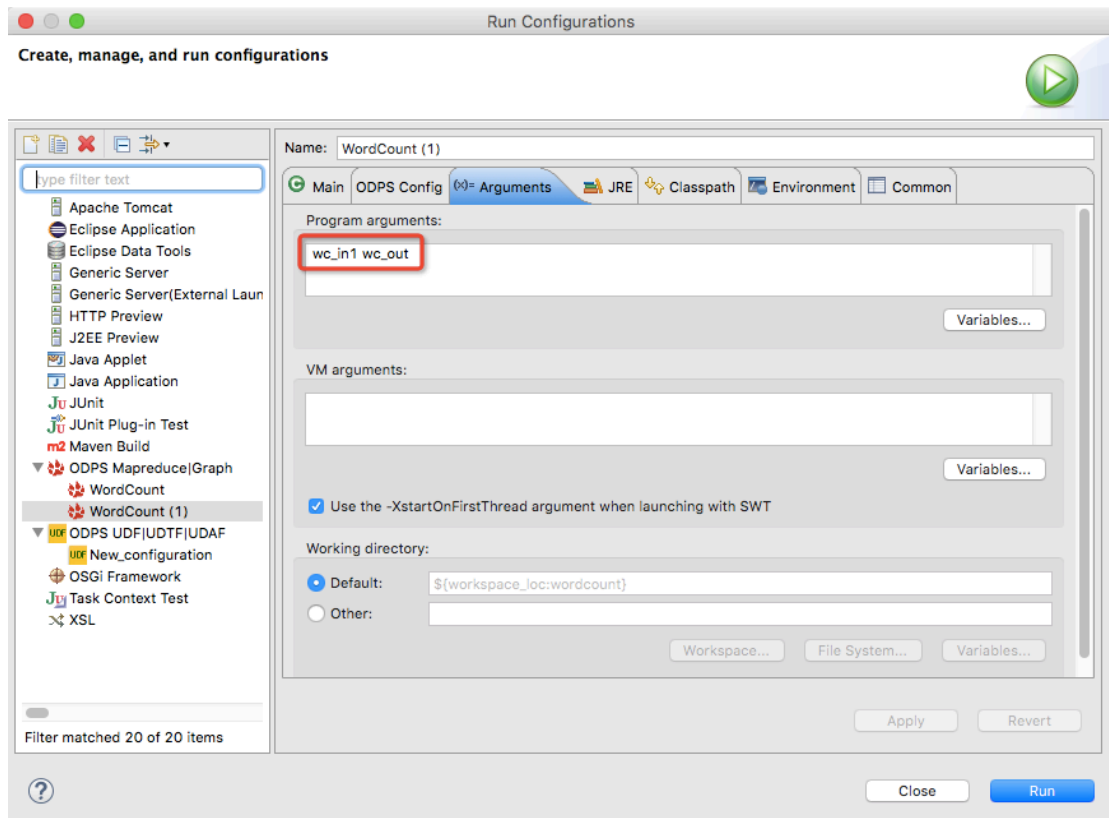
step1 : 从菜单栏选择 **Run-->Run Configurations...**。

step2 : 在 Debug Configurations 弹出框中找到 wordcount , 在 ODPS config 配置页面中 add ODPS project 为 word 与数据目录保持一致。

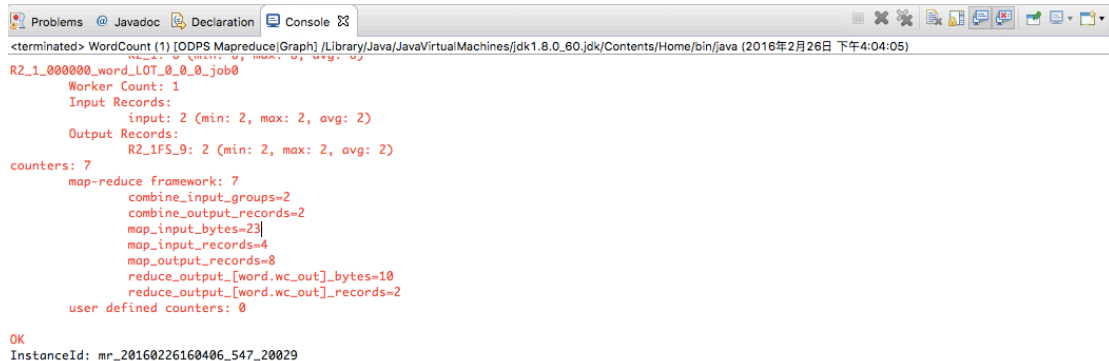


step3 : 切换至 Argument 中 , 编写输入表名和输出表名 , 中间以空格隔开。

本示例为 wc_in1 wc_out。

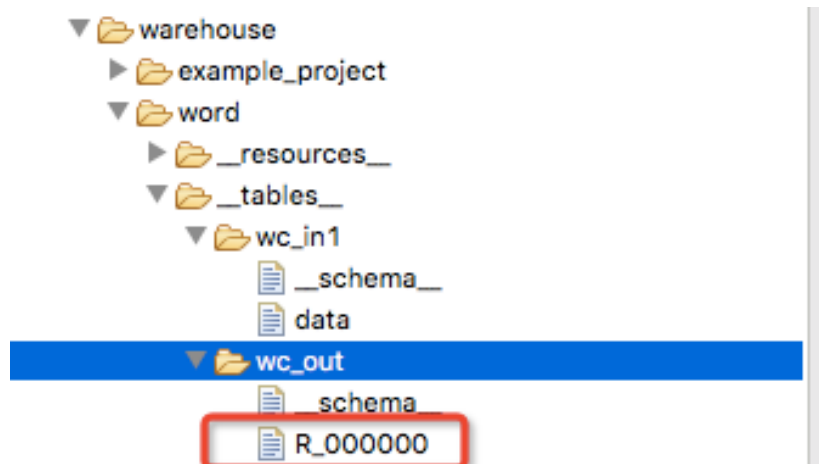
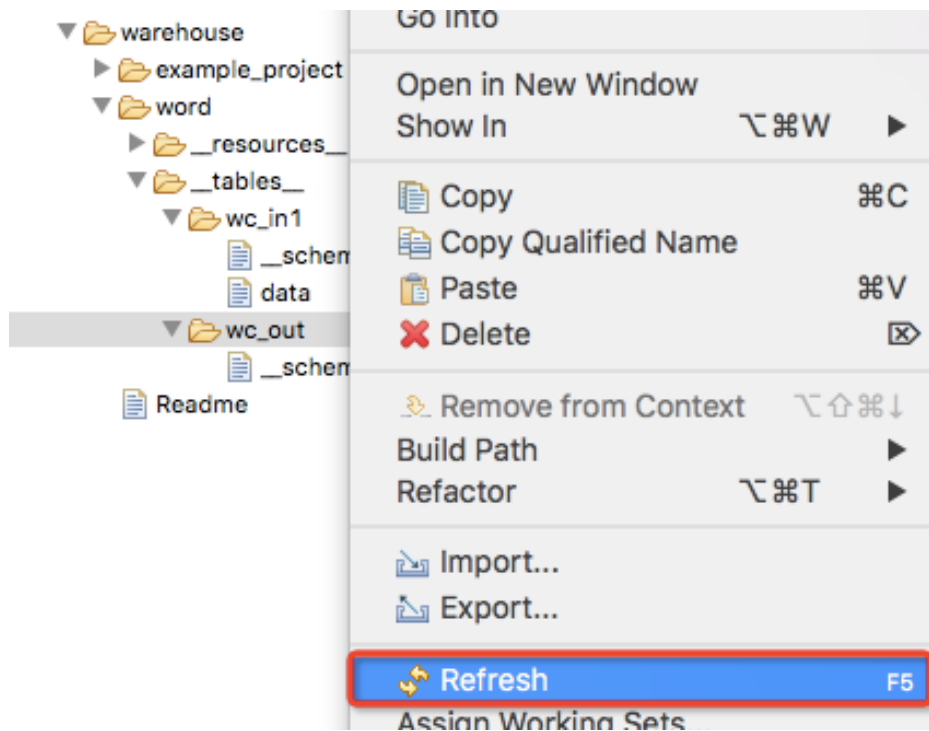


step4 : 配置好信息后，点击 **Run** 进行本地调试。

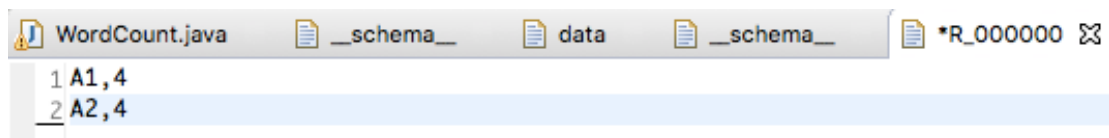


2.2.7 查看测试结果集

step1 : 右键选择 `wc_out` 表，点击刷新，可以看到 `wc_out` 有个结果集产生，如下图所示。

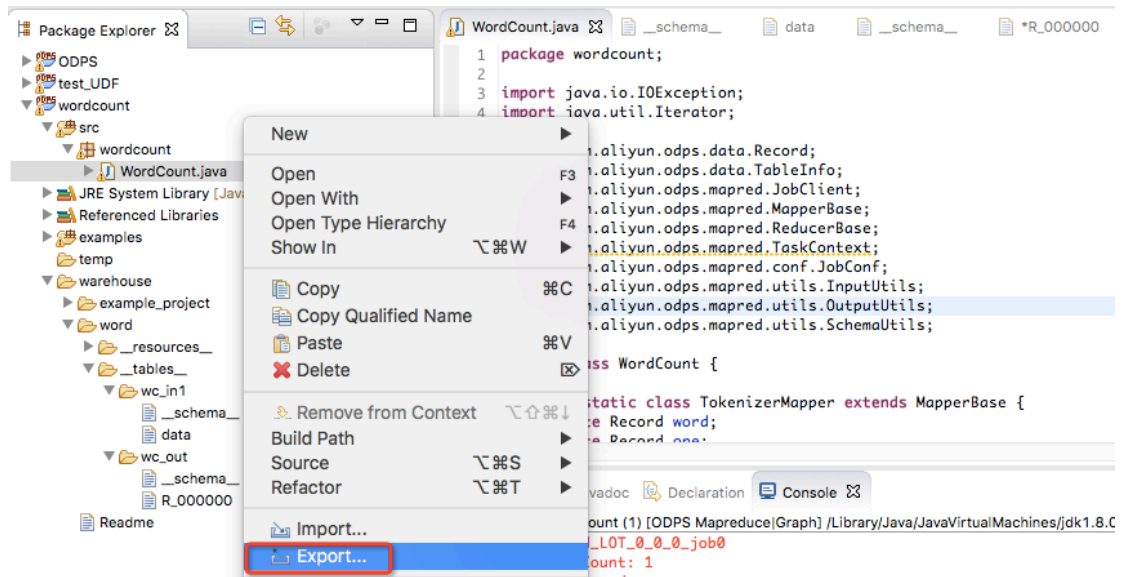


step2 : 双击打开 R_000000 查看结果。

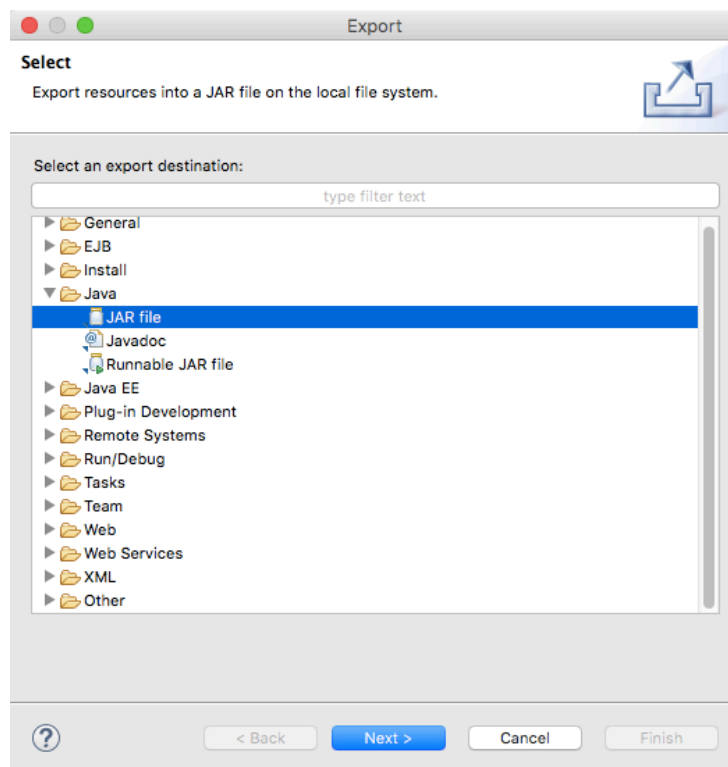


2.3 导出 jar 包

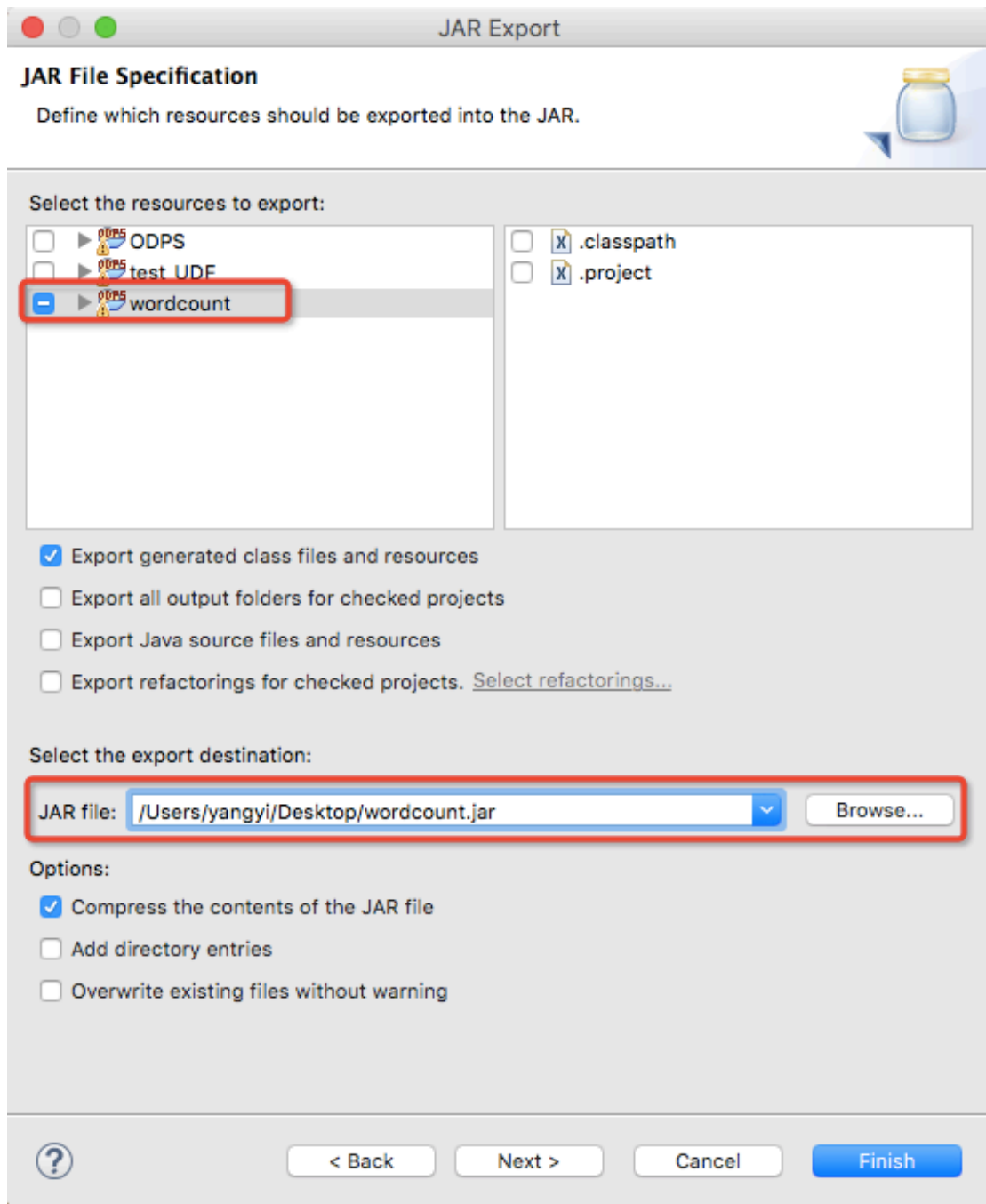
step1 : 找到 wordcount.java 文件，右键选择 Export...



step2 : 在弹出框中选择 JAR file。



step3 : 点击 Next> , 选择需打包的字眼 , 并选择导出路径并命名 jar 包名称 , 本示例选择有意义的名称 , wordcount.jar。



step4：点击 Finish。

2.4 数据表准备

2.4.1 创建表

以新建 wc_in 数据表为例，具体步骤如下：

step 1：以**开发者身份**登录阿里云大数据平台。

step 2：点击顶部菜单栏中的**数据管理**，导航至**数据表管理**。

step 3：点击**新建表**。

step 4：点击 **DDL 建表**。

step 5：填写 ODPS SQL 建表语句。



DDL建表

```
1 CREATE TABLE wc_in (key STRING, value STRING);
```

取消 提交

/*示例代码*/：

```
CREATE TABLE wc_in (key STRING, value STRING);
```

step 6：点击**提交**。

step 7：补充**基础信息**中未被自动填充的配置项。

基础信息

字段和分区信息

新建成功

1 基本信息设置

DDL建表

*表名:

wc_in

别名:

输入表

*项目名:

odps.comtestproj111_dev

所属类目:

测试一级类目

测试二级类目

描述:

MR示例输入表

2 存储生命周期设置

*生命周期:

永久

取消

下一步

step 8 : 点击下一步。

step 9 : 补充新建表字段和分区信息页面中的配置项。

基础信息

字段和分区信息

新建成功!

3 字段信息设置

字段英文名	中文名	字段类型	描述	设置权限	操作
key		STRING			上移 下移 编辑 删除
value		STRING			上移 下移 编辑 删除

+ 新增字段

4 是否设置分区: ☐ 否 ☒ 是

分区信息设置

字段英文名	字段类型	描述	操作
-------	------	----	----

+ 新增分区

取消

上一步

提交

step 8 : 点击提交。



图 2-1 创建数据表成功

输出表 wc_out 建表方法同上，具体 DDL 语句如下：

```
CREATE TABLE wc_out (key STRING, cnt BIGINT);
```

2.4.2 插入示例数据

为感知 ODPS MR 程序在大数据平台上运行的结果，需向输入表（wc_in）中插入示例数据，具体步骤如下：

step 1 :导航至**开发面板>新建或管理面板>脚本文件管理**，点击**新建脚本文件**。

step 2：在**新建脚本文件**弹出框中填写各配置项。

新建脚本文件

*文件名称

wc_in插入示例数据

*类型

ODPS SQL

*描述

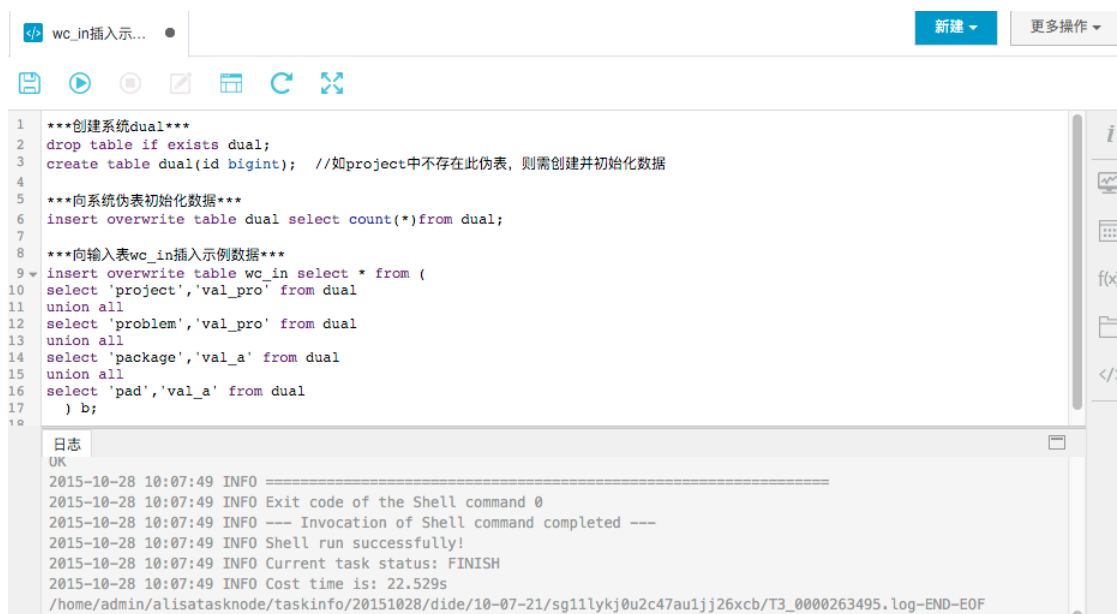
wc_in插入示例数据

取消

提交

step 3：点击**提交**。

step 4 : 在 ODPS 代码编辑器中编写 ODPS SQL。



The screenshot shows the ODPS code editor with a file named 'wc_in插入示...'. The editor contains the following SQL code:

```
1 ***创建系统dual***
2 drop table if exists dual;
3 create table dual(id bigint); //如project中不存在此伪表，则需创建并初始化数据
4
5 ***向系统伪表初始化数据***
6 insert overwrite table dual select count(*)from dual;
7
8 ***向输入表wc_in插入示例数据***
9 insert overwrite table wc_in select * from (
10 select 'project','val_pro' from dual
11 union all
12 select 'problem','val_pro' from dual
13 union all
14 select 'package','val_a' from dual
15 union all
16 select 'pad','val_a' from dual
17 ) b;
```

Below the code editor is a log window titled '日志' (Log) showing the execution details:

```
UK
2015-10-28 10:07:49 INFO =====
2015-10-28 10:07:49 INFO Exit code of the Shell command 0
2015-10-28 10:07:49 INFO --- Invocation of Shell command completed ---
2015-10-28 10:07:49 INFO Shell run successfully!
2015-10-28 10:07:49 INFO Current task status: FINISH
2015-10-28 10:07:49 INFO Cost time is: 22.529s
/home/admin/alisatasknode/taskinfo/20151028/dide/10-07-21/sg11lykj0u2c47au1jj26xcb/T3_0000263495.log-END-EOF
```

提供 ODPS SQL 脚本如下：

*****创建系统 dual*****

drop table if exists dual;

create table dual(id bigint); //如 project 中不存在此伪表，则需创建并初始化数据

*****向系统伪表初始化数据*****

insert overwrite table dual select count(*)from dual;

*****向输入表 wc_in 插入示例数据*****

insert overwrite table wc_in select * from (

select 'project','val_pro' from dual

union all

select 'problem','val_pro' from dual

union all

```
select 'package','val_a' from dual

union all

select 'pad','val_a' from dual

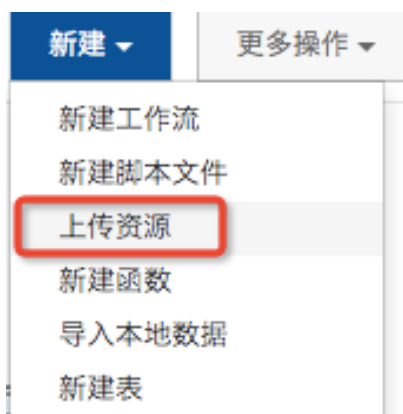
) b;
```

可以预览已经插入的示例数据，如下图：

select * from wc_in;			
日志	结果[1]×		
序号	key	value	
1	project	val_pro	
2	problem	val_pro	
3	package	val_a	
4	pad	val_a	

2.5 注册 ODPS 资源

step1：进入 Data IDE，点击新建选择上传资源。



step2 在资源上传页面中选择 章节 2.3 中导出的 jar 包，并选择类型为 jar，同时勾选上传为 ODPS 资源，然后选择资源存放的文件目录。

资源上传

*名称:

WordCount.jar

*类型:

jar

*上传:

选择文件

WordCount.jar

*描述:

统计表中单词出现的次数

☒ 上传为ODPS资源

选择目录:

/ test_folder111 /

资源管理

test_folder111

取消

提交

step3：点击提交。

2.6 创建 workflow 并配置 ODPS MR

step1：进入 Data IDE，点击新建选择新建 workflow。

新建工作流

*工作流名称:

wordcount_MR

*描述:

MR测试

*调度类型:

☒ 一次性调度

☐ 周期调度

选择目录:

/ test_folder111 /

任务开发

hnj

test_folder111

【备注】：此处为了便于测试 MR，选择了一次性调度。

step2：向工作流设计器中拖入 ODPS MR 节点，配置节点属性，并点击创建。如下：

新建节点

*名称:

wordcount_MR

*类型:

ODPS_MR

描述:

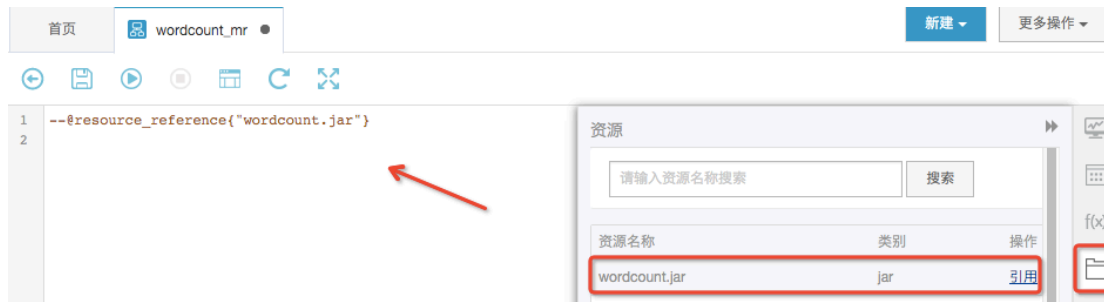
wordcount_MR测试

取消

创建

step3：双击 wordcount_MR 节点进入 MR 代码编辑界面。

step4：展开 ODPS MR 代码编辑器右侧资源 tab，点击 wordcount.jar 后的引用。如下图：



step4 : 编写 ODPS MR 语句。



```
jar -resources wordcount.jar -classpath wordcount.jar com.ali.data.meta.example.WordCount wc_in wc_out;
```

step5 : 点击运行。

step6 : 待运行成功后，在脚本文件中直接执行 `select * from wc_out` 查看示例结果。

```
select key as word,cnt as 出现次数 from wc_out;
```

日志	结果[1]×		
序号	word	出现次数	
1	package	1	
2	pad	1	
3	problem	1	
4	project	1	
5	val_a	2	
6	val_pro	2	

2.7 创建工作流并配置 OPEN MR

在 Data IDE 中用户也可以通过工单系统来申请开通使用 OPEN MR ,其使用方法如下。其使用上与 ODPS MR 类似，OPEN MR 提供了表单模式来配置，具体如下：

step1：向工作流设计器中拖入 OPEN MR 节点，并双击进入配置界面。

step2：选择 MR Jar 包，并填输入表、mapper 类、reducer 类同时指定输出表表名、输出 key 与 value。

MRJar包	<input type="text" value="wordcount.jar"/>	✕ 🔍 + -
--------	--	---

资源	<input type="text" value="wordcount.jar"/>	+
----	--	----------------

输入表	<input type="text" value="wc_in"/>	+ -
mapper	<input type="text" value="com.ali.data.meta.example.WordCount\$TokenizerMapper"/>	必选
reducer	<input type="text" value="com.ali.data.meta.example.WordCount\$SumReducer"/>	
combiner	<input type="text" value="请输入combiner"/>	
输出表	<input type="text" value="wc_out"/>	
输出Key	<input type="text" value="key:string"/>	
输出Val	<input type="text" value="cnt:bigint"/>	

OutputKeySortColumns	<input type="text" value="如：col_1,col_2"/>	OutputGroupingColumns	<input type="text" value="如：col_1,col_2"/>
PartitionColumns	<input type="text" value="如：col_1,col_2"/>	SplitSize	<input type="text" value="请输入数字，单位：MB"/>
NumReduceTasks	<input type="text" value="请输入数字"/>	MemoryForMapTask	<input type="text" value="请输入数字，单位：MB"/>
MemoryForReduceTask	<input type="text" value="请输入数字，单位：MB"/>		

step3：点击**保存**，并执行。



【备注】同时可以点击右上角来切换表单模式和 xml 模式。

