MaxCompute

工具及下载

工具及下载

本文档将为您介绍如何借助客户端命令行工具使用 MaxCompute 服务的基础功能。在使用 MaxCompute 客户端前，请首先 安装并配置客户端。

注意：

- 请不要依赖客户端的输出格式来做任何的解析工作。客户端的输出格式不承诺向前兼容，不同版本间的客户端命令格式及行为有差异。
- 关于客户端的基本命令介绍，请参见 基本命令。

安装并配置好客户端后，您可借助命令行工具进行以下操作：

获取帮助

若想显示客户端的帮助信息，命令格式如下所示：

```
odps@ >./bin/odpscmd -h;
```

您也可以在交互模式下键入“h;”或“help;”（不区分大小写）。

客户端还提供了 help [keyword]; 命令，可获取到与关键字有关的命令提示。例如：输入 help table; 可以得到与 table 操作相关的命令提示，如下所示：

```
odps@ odps> help table;
Usage: alter table <tablename> merge smallfiles
Usage: show tables [in <projectname>]
list|ls tables [-p,-project <projectname>]
Usage: describe|desc [<projectname>.]<tablename> [partition(<spec>)]
Usage: read [<project_name>.]<table_name> [(<col_name>[,..])] [PARTITION (<partition_spec>)] [line_num]
```

启动参数

在启动时，您可指定一系列参数，如下所示：

```
Usage: odpscmd [OPTION]...
where options include:
--help (-h)for help
--project=<prj_name> use project
```

```
--endpoint=<http://host:port> set endpoint
-u <user_name> -p <password> user name and password
-k <n> will skip begining queries and start from specified position
-r <n> set retry times
-f <"file_path;"> execute command in file
-e <"command;[command;]..."> execute command, include sql command
-C will display job counters
```

以 -f 参数为例，操作如下：

准备本地脚本文件 script.txt，假设在 D 盘，文件内容如下所示：

```
DROP TABLE IF EXISTS test_table_mj;
CREATE TABLE test_table_mj (id string, name string);
DROP TABLE test_table_mj;
```

运行如下命令：

```
odpscmd\bin>odpscmd -f ./script.sql;
```

交互模式

直接运行客户端即可进入到交互模式，如下所示：

```
[admin: ~]$odpscmd
Aliyun ODPS Command Line Tool
Version 1.0
@Copyright 2012 Alibaba Cloud Computing Co., Ltd. All rights reserved.
odps@ odps> INSERT OVERWRITE TABLE DUAL SELECT * FROM DUAL;
```

在光标位置输入命令（以分号作为语句的结束标志），回车即可运行。

续跑

- 在用 -e 或 -f 模式运行时，如果有多条语句，想从中间某条语句开始运行，可以指定参数 -k， 表示忽略前面的语句，从指定位置的语句开始运行。当指定参数 <= 0 时，从第一条语句开始执行。
- 每个以分号分隔的语句被视为一条有效语句，在运行时会打印出当前运行成功或者失败的是第几条语句。

示例如下：

文件/tmp/dual.sql 中有三条 SQL 语句，如下所示：

```
drop table dual;
create table dual (dummy string);
insert overwrite table dual select count(*) from dual;
```

若想忽略前两条语句，直接从第三条语句开始执行，命令格式如下所示：

```
odpscmd –k 3 –f dual.sql
```

获取当前登录用户

若想获取当前登录用户，命令格式如下所示：

```
whoami;
```

示例如下：

```
odps@ hiveut>whoami;
Name: odpstest@aliyun.com
End_Point: http://service.odps.aliyun.com/api
Project: lijunsecuritytest
```

通过以上命令，即可获取当前登录用户的云账号、使用的 End_Point 配置和项目名。

退出

若想退出客户端，命令格式如下所示：

```
odps@ > quit;
```

或输入：

```
odps@ > q;
```

MaxCompute Studio

快速开始

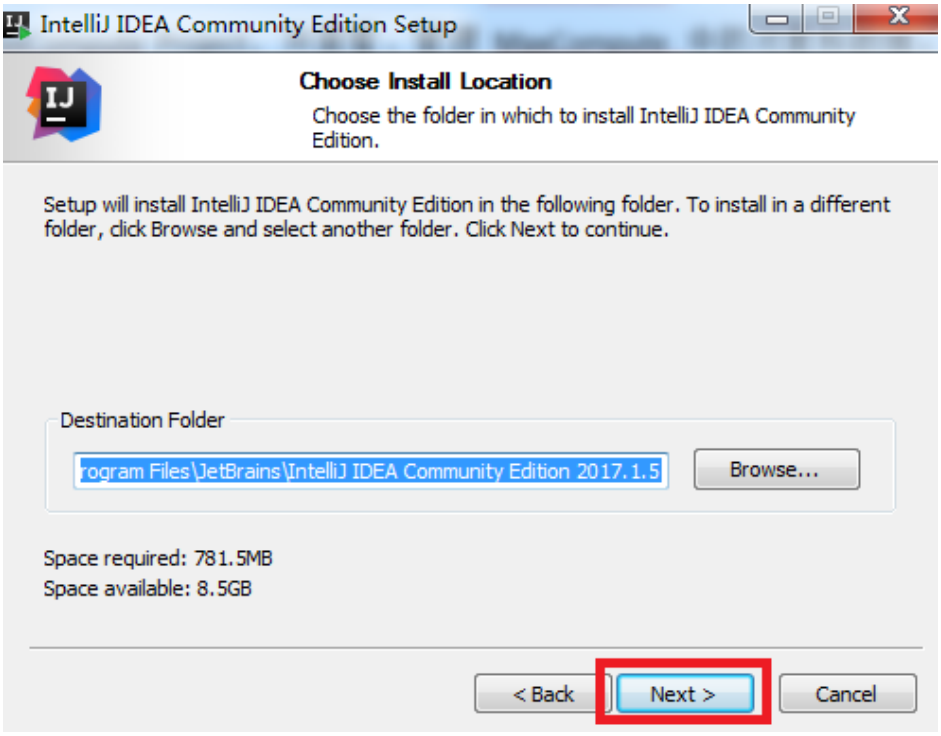
MaxCompute Studio 是阿里云 MaxCompute 平台提供的安装在开发者客户端的大数据集成开发环境（IDE）工具，是一套基于流行的集成开发平台 IntelliJ IDEA 的开发插件，帮助用户方便地进行数据开发。本文将介绍如何安装 MaxCompute Studio 的基础平台 IntelliJ IDEA。

操作步骤

单击 **此链接**，下载 IntelliJ IDEA。

下载完成后，双击安装程序，进入安装界面，单击 **Next**。如下图所示：

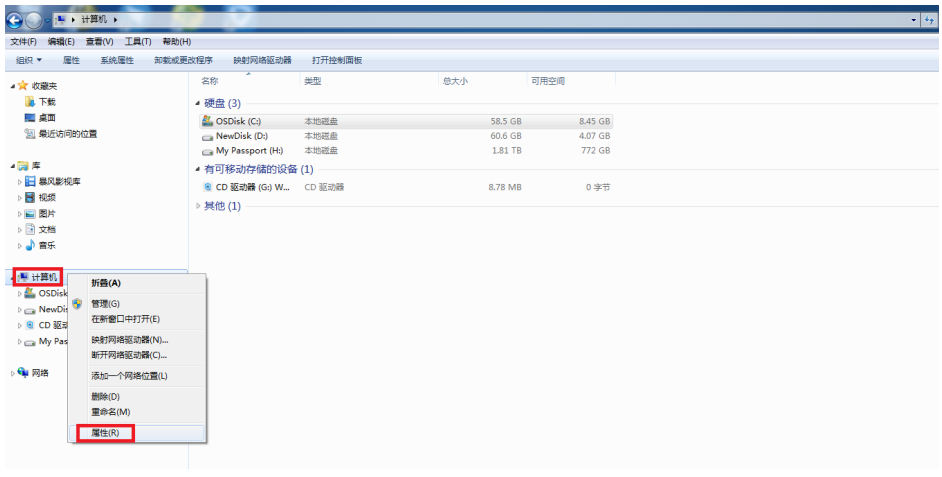
指定安装目录后，单击 **Next**。如下图所示：



根据本地操作系统的版本选择需要安装 32 位或者 64 位的 IntelliJ IDEA。

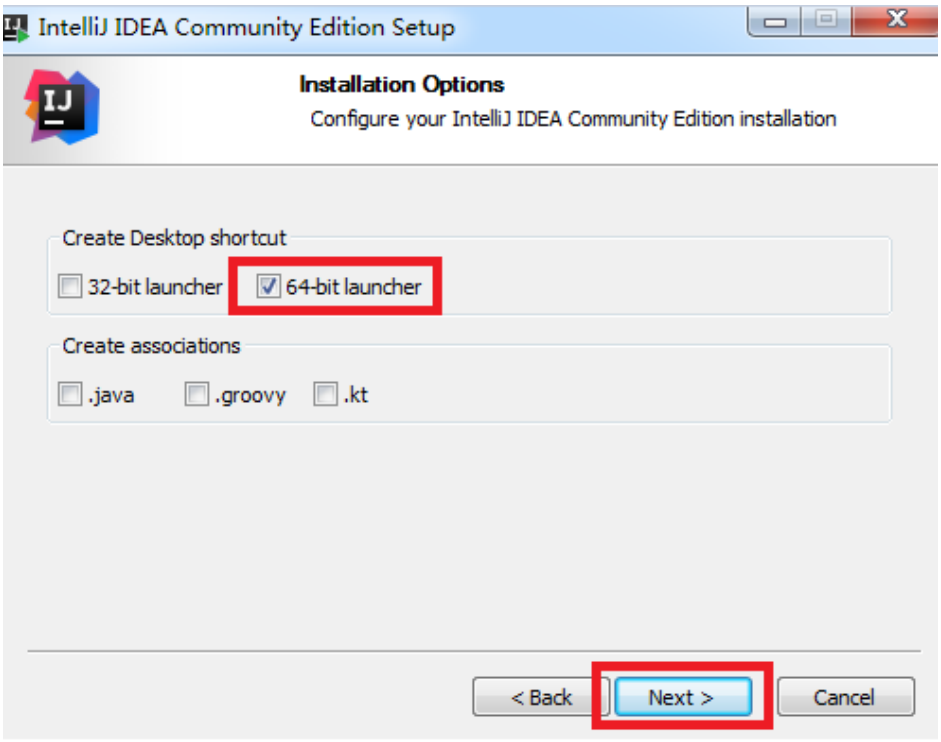
如何查看本地操作系统的版本？

打开 windows 资源管理器，右击计算机，在菜单中选中属性。如下图所示：

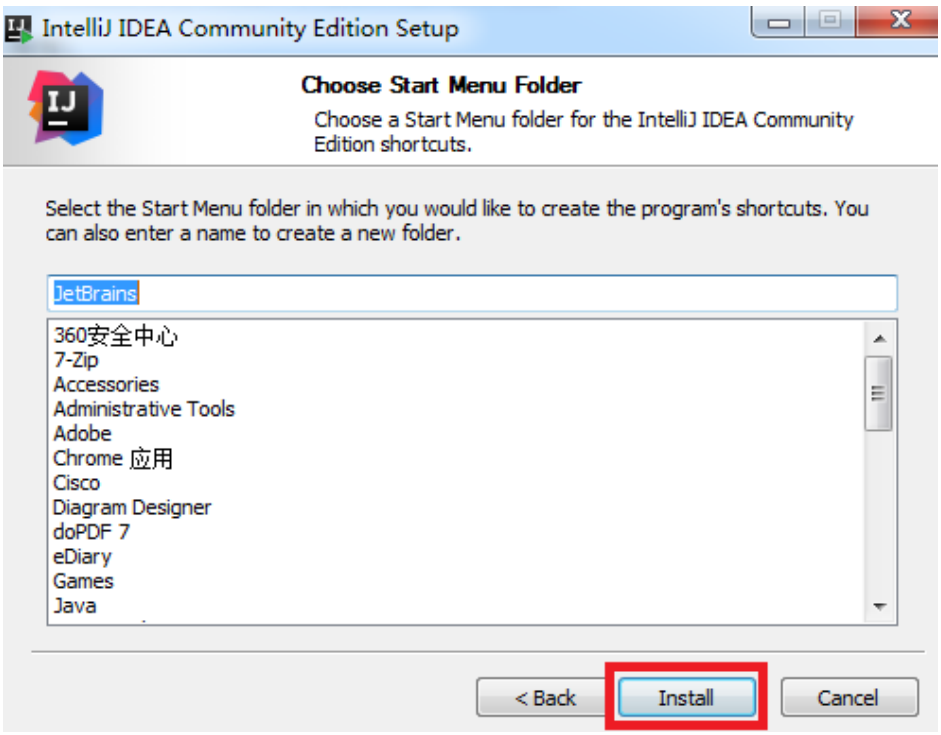


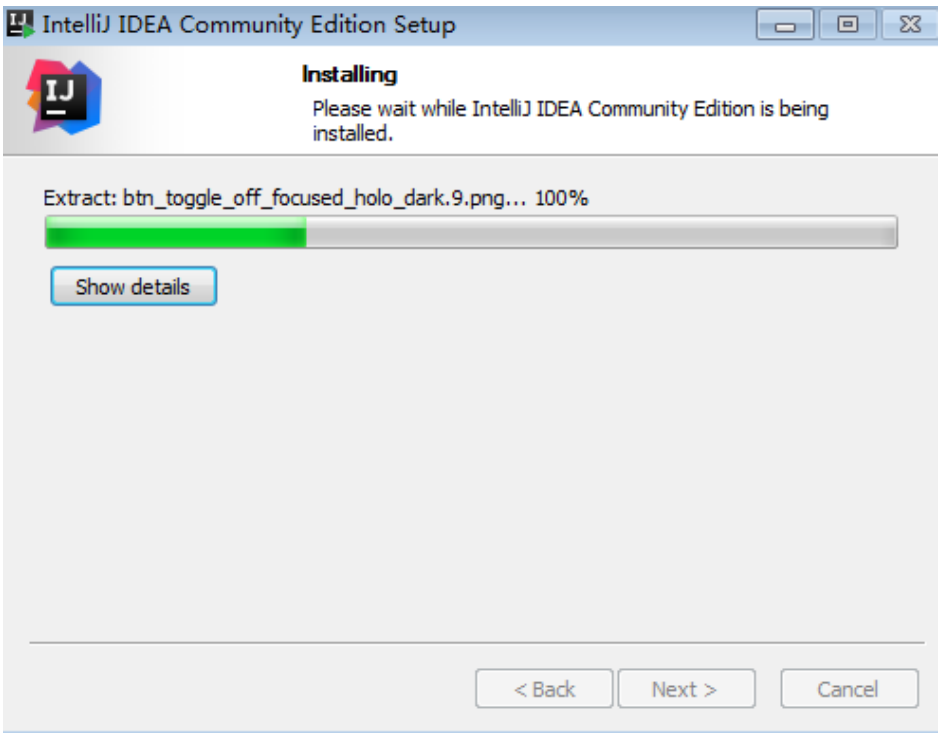
在弹出界面中查看操作系统位数，如下图所示：

选择相应的系统类型，单击 **Next**。如下图所示：



单击 **Install**，开始安装。如下图所示：





安装完成后，单击 **Finish**。

MaxCompute Studio 是阿里云 MaxCompute 平台提供的安装在开发者客户端的大数据集成开发环境（IDE）工具，是一套基于流行的集成开发平台 IntelliJ IDEA 的开发插件，帮助用户方便地进行数据开发。

本文将快速介绍 MaxCompute Studio 的功能界面和常用的使用场景，包括以下部分：

- 基本用户界面
- 连接 MaxCompute 项目空间
- 管理数据
- 编辑 SQL 脚本
- SQL 代码智能提示
- 编译和提交作业
- 查看历史作业
- 开发 MapReduce 程序和 UDF 程序
- 连接 MaxCompute 客户端

基本用户界面

MaxCompute Studio 是 IntelliJ IDEA 平台上的一套插件，共享了 IntelliJ IDEA 的基本开发界面。对 IDEA 界面不熟悉的用户，可以参考 IDEA 界面操作文档

MaxCompute Studio 在 IntelliJ 的基础上提供了以下的功能界面：

- **SQL 编辑器（SQL Editor）**：提供 SQL 语法高亮、代码补全、实时错误提示、本地编译、作业提交等功能
    - **编译器视图（Compiler View）**：显示本地编译的提示信息和错误信息，在编辑器中定位代

码

- **项目空间浏览器（Project Explorer)**: 连接 MaxCompute 项目空间，浏览项目空间表结构、自定义函数、资源文件
    - **表详情视图（Table Details View）**: 提供表、视图等资源的详情显示和示例数据（Sample Data）
- **作业浏览器（Job Explorer）**: 浏览、搜索 MaxCompute 的历史作业信息
    - **作业详情视图（Job Details View）**: 显示作业的运行详细信息，包括执行计划和每个执行任务的详细信息，包括 logview 工具能够显示的全部信息
    - **作业输出视图（Job Output View）**: 显示正在运行的作业的输出信息
    - **作业结果视图（Job Result View）**: 显示 SELECT 作业的输出结果
- **MaxCompute 控制台（MaxCompute Console）**: 集成了 MaxCompute 客户端，可以输入和执行 MaxCompute 客户端命令

连接 MaxCompute 项目空间

大部分 Studio 的功能需要用户首先建立 MaxCompute 项目空间的连接。建立了项目空间连接以后，就可以在 **项目空间浏览器（Project Explorer）** 中查看相关的数据结构和资源信息了。Studio 会自动为每一个项目空间连接建立一个本地的元数据备份，以能够大大提高对 MaxCompute 元数据的访问频率和降低延时。

用户通过 Studio 完成编辑 SQL 脚本、提交作业、查看 Job 信息、打开 MaxCompute 控制台等功能都需要指定作为目标的项目连接，因此首先建立一个 MaxCompute 项目空间的连接是非常必要的。

如果您对 MaxCompute 项目空间的概念不熟悉，请参考 MaxCompute 基本概念的介绍。
如果您希望了解在 Studio 中管理项目空间的更多内容，请参考项目空间管理部分的文档

管理数据

用户可以通过 Studio 的 **项目空间浏览器（Project Explorer）** 快速浏览项目空间的表结构、自定义函数、资源文件。通过树形控件，可以列出所有项目空间连接下的数据表、列、分区列、虚拟视图、自定义函数名称、函数签名、资源文件及类型等。支持 Quick Search 快速定位。

双击某个数据表，可以打开 **表详情视图（Table Details View）** 查看数据表的元信息、表结构和示例数据。如果用户没有项目空间的相应权限，Studio 会提示对应的错误信息。

Studio 集成了 MaxCompute Tunnel 工具，可以支持本地数据的上传和下载，具体功能可以参考文档

编写 SQL 脚本

在 Studio 中编写 MaxCompute SQL 脚本非常方便，通过菜单 **File | New | Project...** 或者 **File | New | Module...**创建一个 “MaxCompute Studio” 类型的项目或者模块， 然后通过菜单 **File | New | MaxCompute Script** 或者右键菜单 **New | MaxCompute Script**就可以创建一个 MaxCompute SQL 脚本文件。

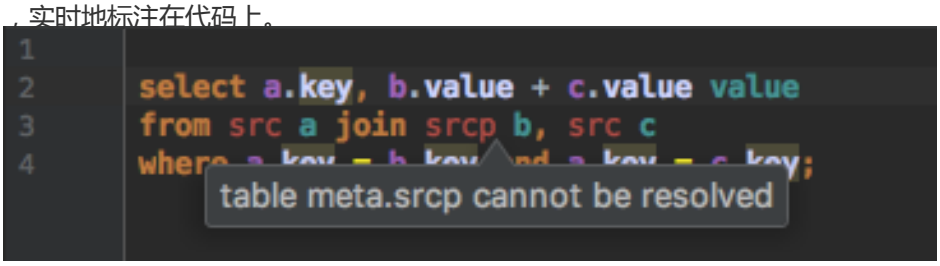
创建 MaxCompute SQL 脚本的时候，Studio 会提示用户选择一个关联的 MaxCompute 项目空间，用户也可

以通过 SQL 编辑器上的工具条最右侧的项目空间选取器进行更改。编辑器会根据 SQL 脚本关联的项目空间对 SQL 语句自动进行元数据（比如表结构等）的检查并汇报错误，提交运行时也会发送到关联的项目空间执行。

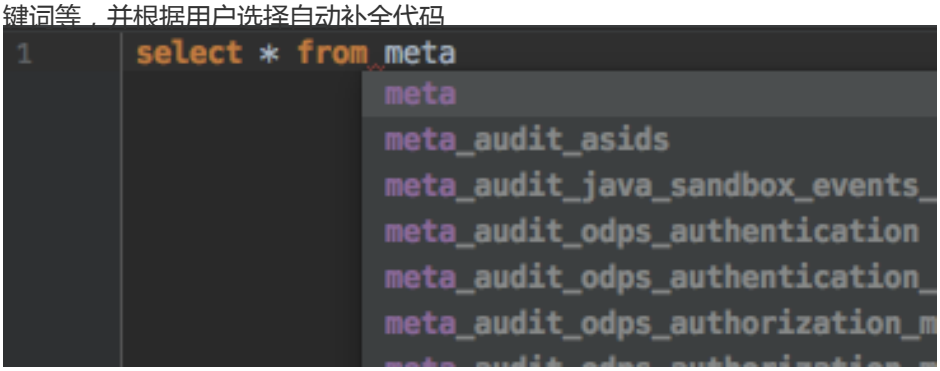
参考 编写SQL脚本

SQL 代码智能提示

Studio 提供的 SQL 编辑器可以随着用户键入代码，智能提示 SQL 语句的语法错误、类型匹配错误或者警告等，实时地标注在代码上。

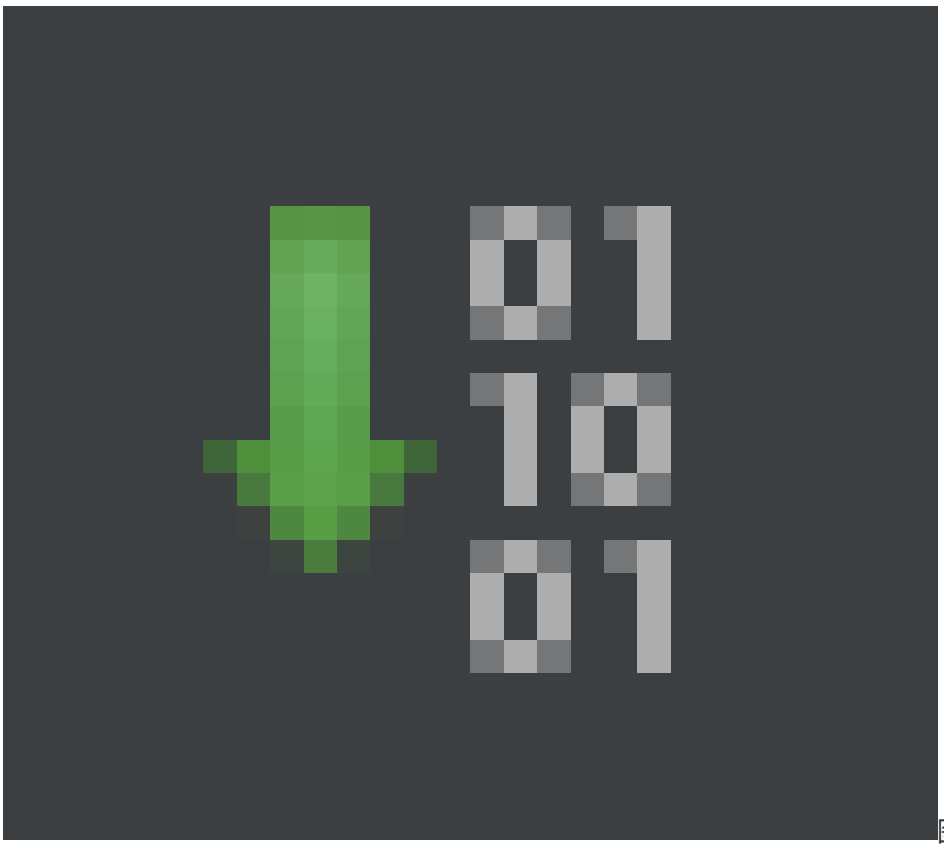


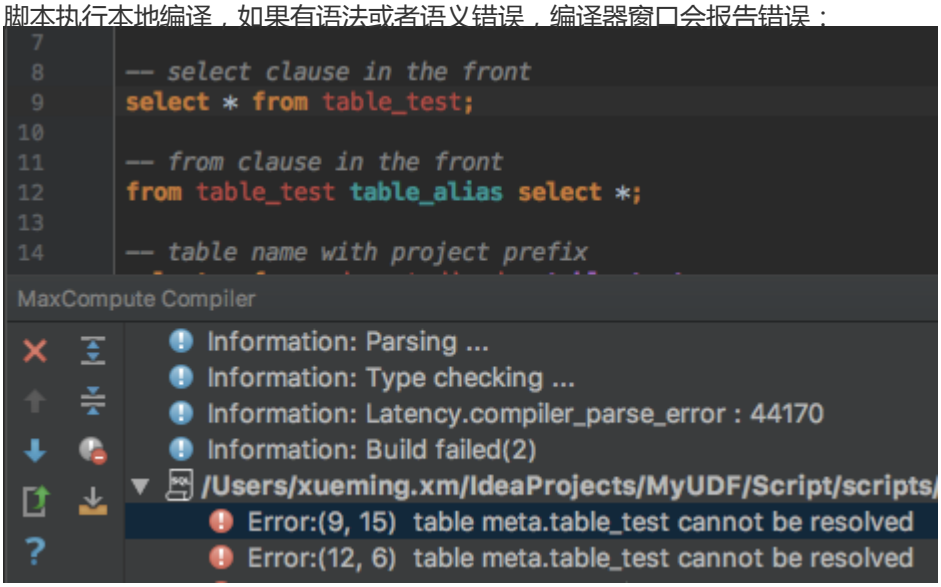
通过代码补全功能，Studio 可以根据代码上下文，给用户提示项目空间名称、表、字段、函数、类型、代码关键词等，并根据用户选择自动补全代码。



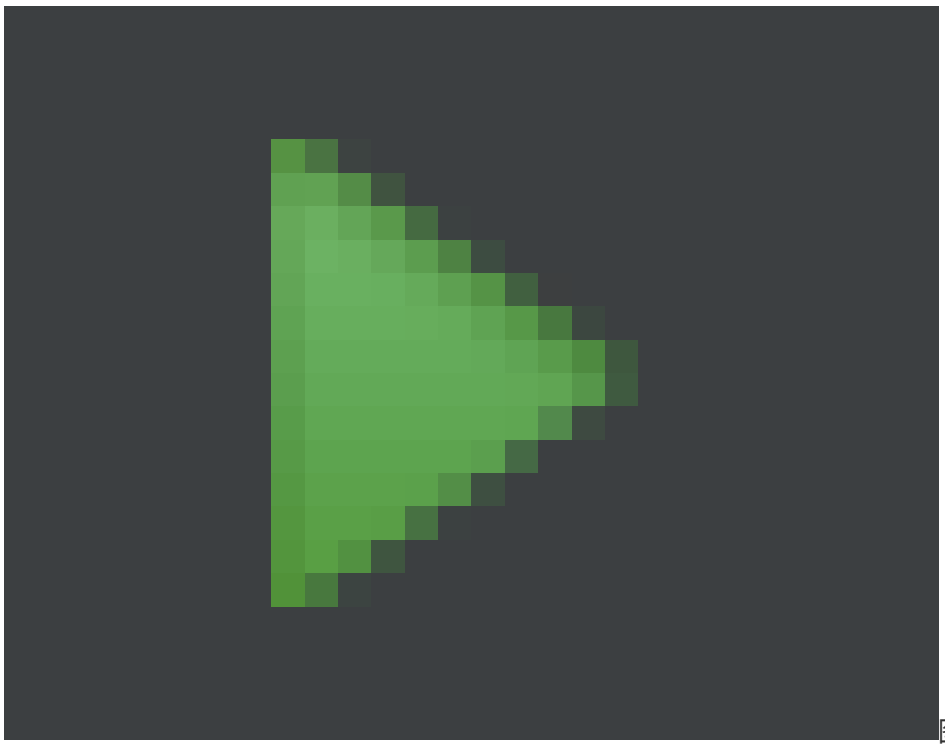
编译和提交作业

点击 SQL 编辑器工具条上的


图标，可以对 SQL 脚本执行本地编译，如果有语法或者语义错误，编译器窗口会报告错误：

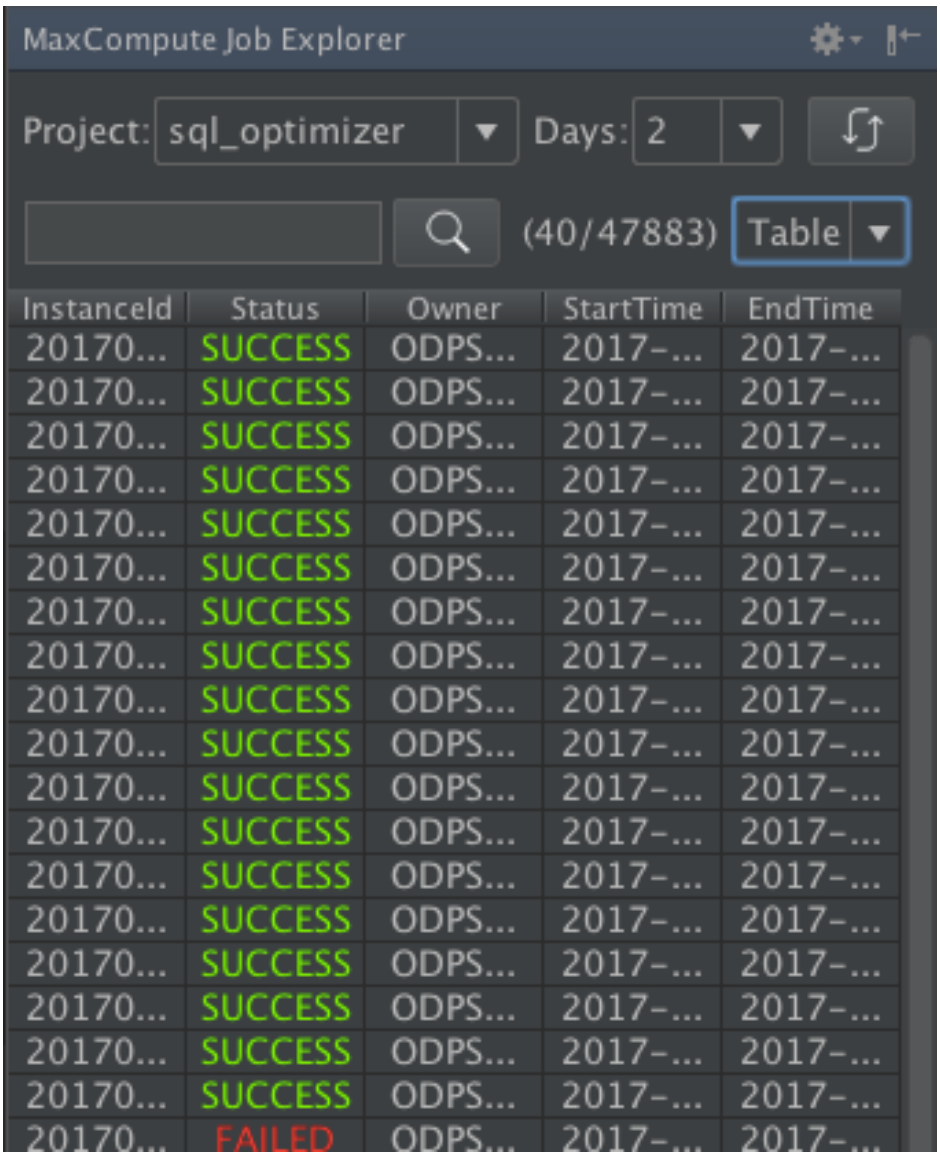


点击 SQL 编辑器工具条上的

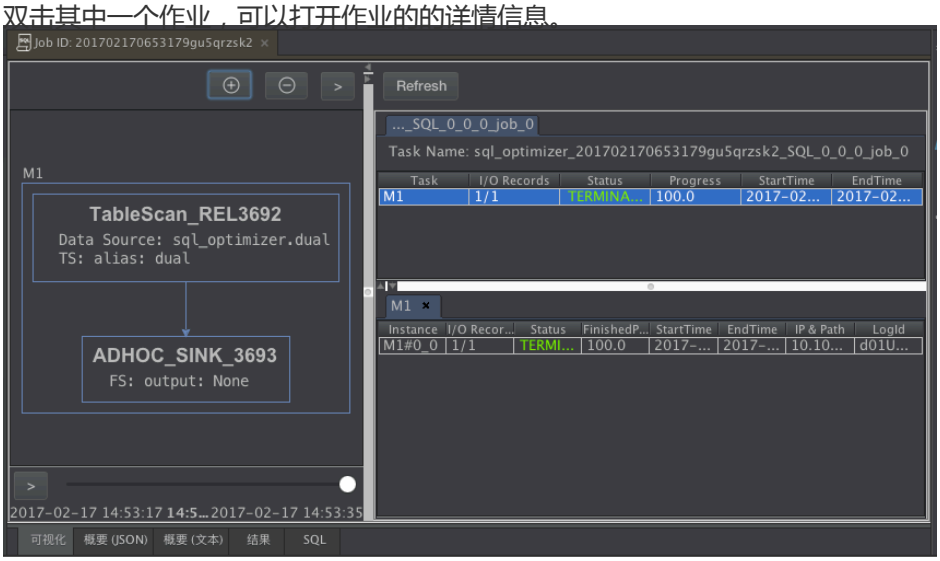
 图标，会在本地编译之后，把 SQL 脚本提交到 MaxCompute 指定的项目空间排队执行

查看历史作业

打开 **作业浏览器（Job Explorer）** 可以查看指定项目空间上近期执行的作业。 注意，这个列表只能显示以当前连接使用的用户 ID 提交的作业。



双击其中一个作业，可以打开作业的的详情信息。



如果知道一个任务的 Logview URL，可以使用菜单 **MaxCompute | Open Logview** 打开该任务的详情页面。

开发 MapReduce 程序和 UDF

Studio 还支持：

- MapReduce 程序的开发，参考文档，
- Java UDF 的开发，参考文档,

连接 MaxCompute 客户端

Studio 集成了最新版本的 MaxCompute 客户端。 也可以在 Studio 的配置页面中指定本地已经安装好的 MaxCompute 客户端路径。在 **项目空间浏览器（Project Explorer）** 中选定一个项目空间，右键菜单选择 **Open n Console** 可以打开 **MaxCompute 控制台（MaxCompute Console）** 窗口。

下一步

- 立即安装 MaxCompute Studio

环境要求

IntelliJ IDEA 支持在 *Windows*， *Mac*， *Linux* 操作系统上安装，硬件及系统环境要求请参见 https://www.jetbrains.com/help/idea/2016.3/requirements-for-intellij-idea.html 。基于 IntelliJ IDEA 平台的 MaxCompute Studio 也可以安装在这些操作系统的客户端上。

MaxCompute Studio 对用户环境有以下要求：

- Windows，Mac OS，或者 Linux 系统客户端。
- 安装 IntelliJ IDEA 14.1.4 以上版本（支持 Ultimate 版本、PyCharm 版本和免费的 Community 版本）。
- 已安装 JRE 1.8（最新的 IntelliJ IDEA 版本捆绑了 JRE 1.8）。
- 已安装 JDK 1.8（*可选*：如果需要开发和调试 Java UDF，需要安装 JDK）。

安装方式

MaxCompute Studio 是 IntelliJ IDEA 的插件，有以下两种安装方式：

- 通过插件库在线安装（推荐）。
- 通过本地文件安装。

在线安装（推荐）

MaxCompute Studio 插件已对全部公网用户开放，您可以通过 IntelliJ 官方插件库安装。

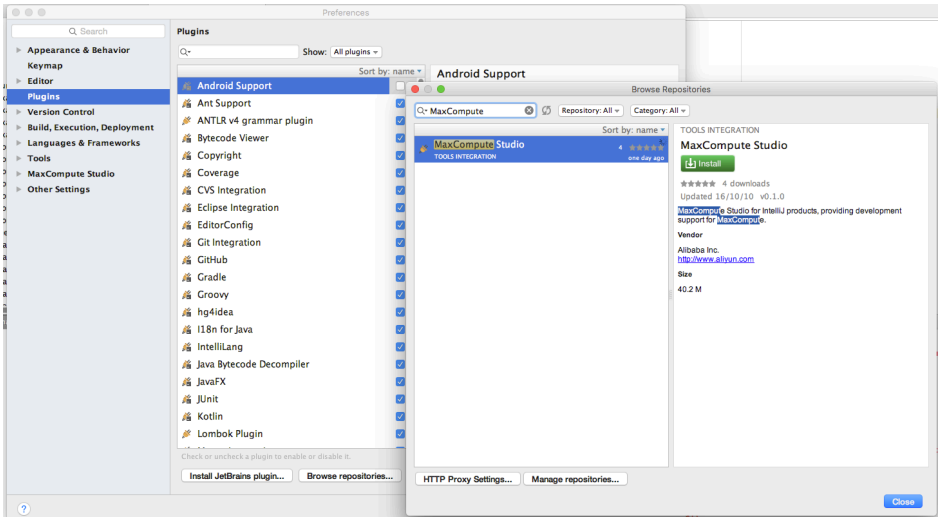
操作步骤

1. 在 IntelliJ IDEA 中打开插件配置页面（Windows/Linux 用户导航至 **File > Settings > Plugins**，Mac 用户导航至 **IntelliJ IDEA > Preferences > Plugins**）。

   单击 **Browse repositories...** 按钮，然后搜索 MaxCompute Studio。

   找到 MaxCompute Studio 插件页面，单击绿色 **Install** 按钮进行安装。

   确认安装后，重新启动 IntelliJ IDEA，完成安装。



本地安装

MaxCompute Studio 也可以在本地环境中进行安装。

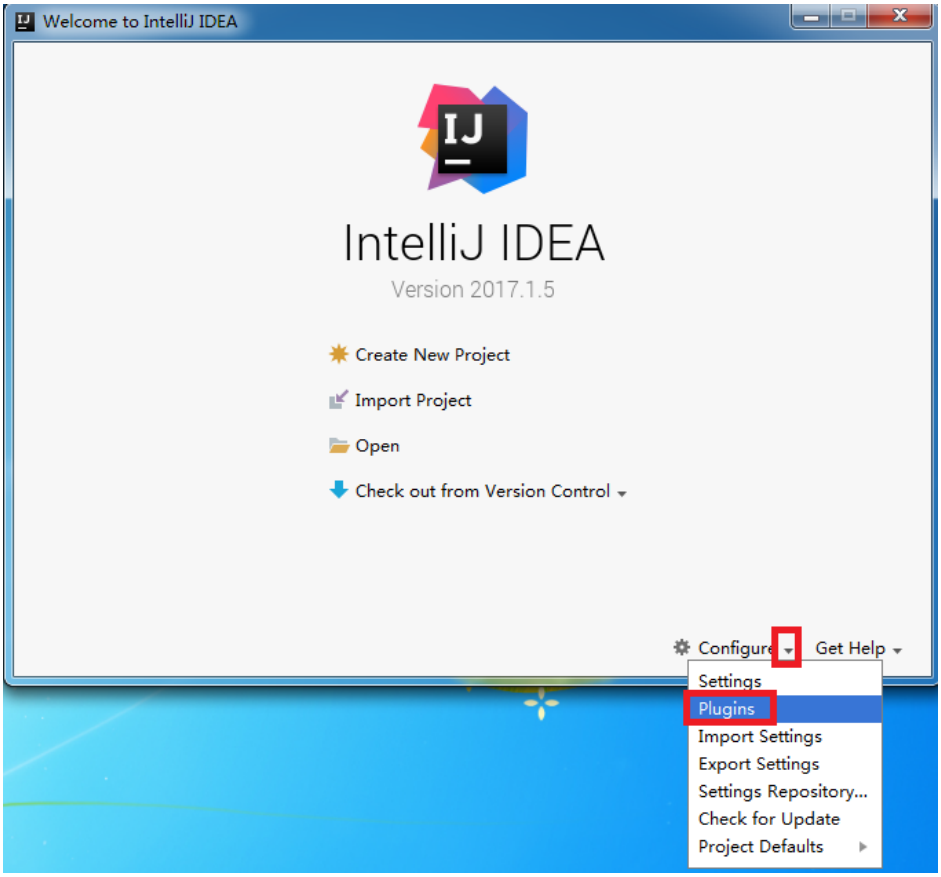
操作步骤

进入 MaxCompute Studio 插件页面 下载插件包。

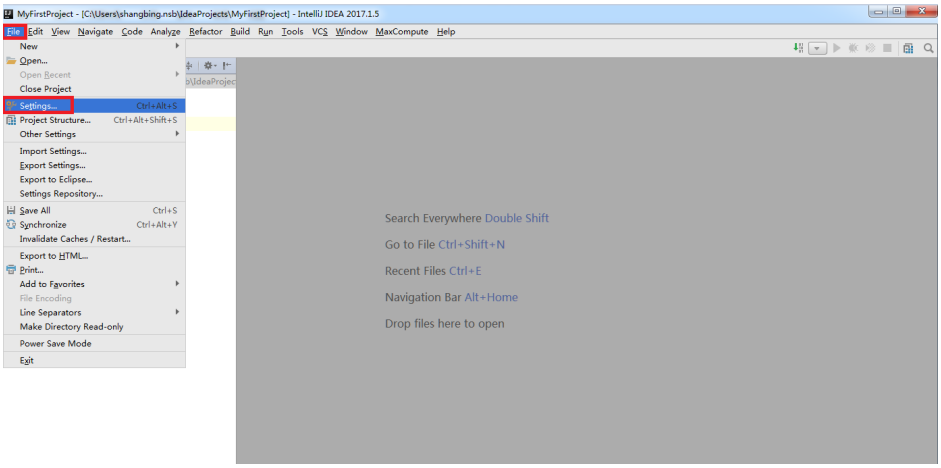
运行 IntelliJ IDEA。

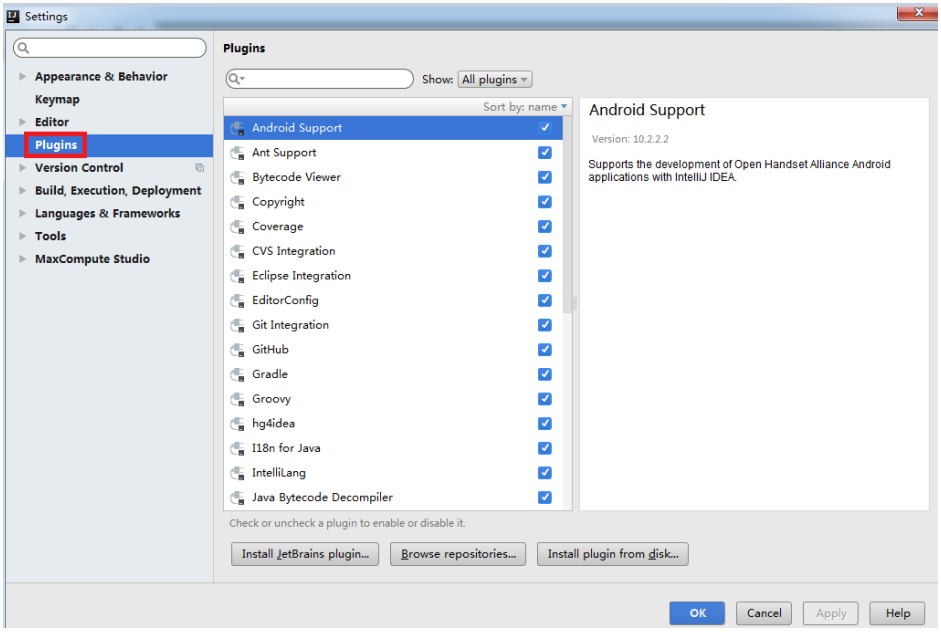
如果是第一次，会出现欢迎界面，单击欢迎界面中的 **configure**（配置），选中弹出菜单中

的 **Plugins**（插件），如下图所示：

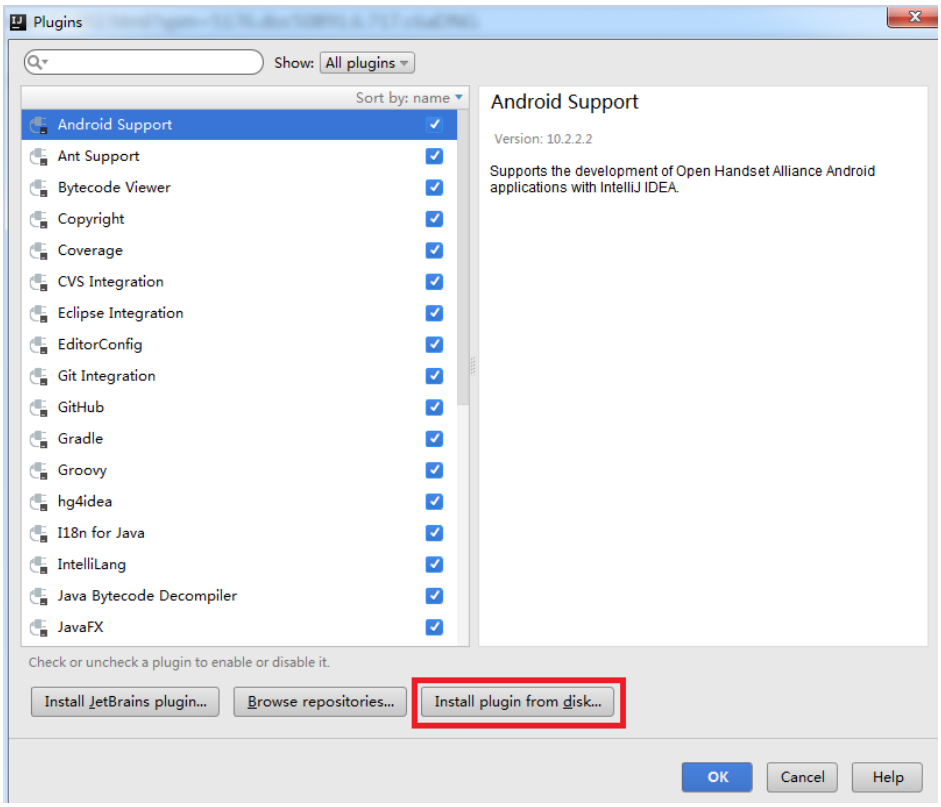


如果不是第一次运行，可以依次单击菜单 **File > Settings > Plugins** 进入相同的界面，如下图所示：

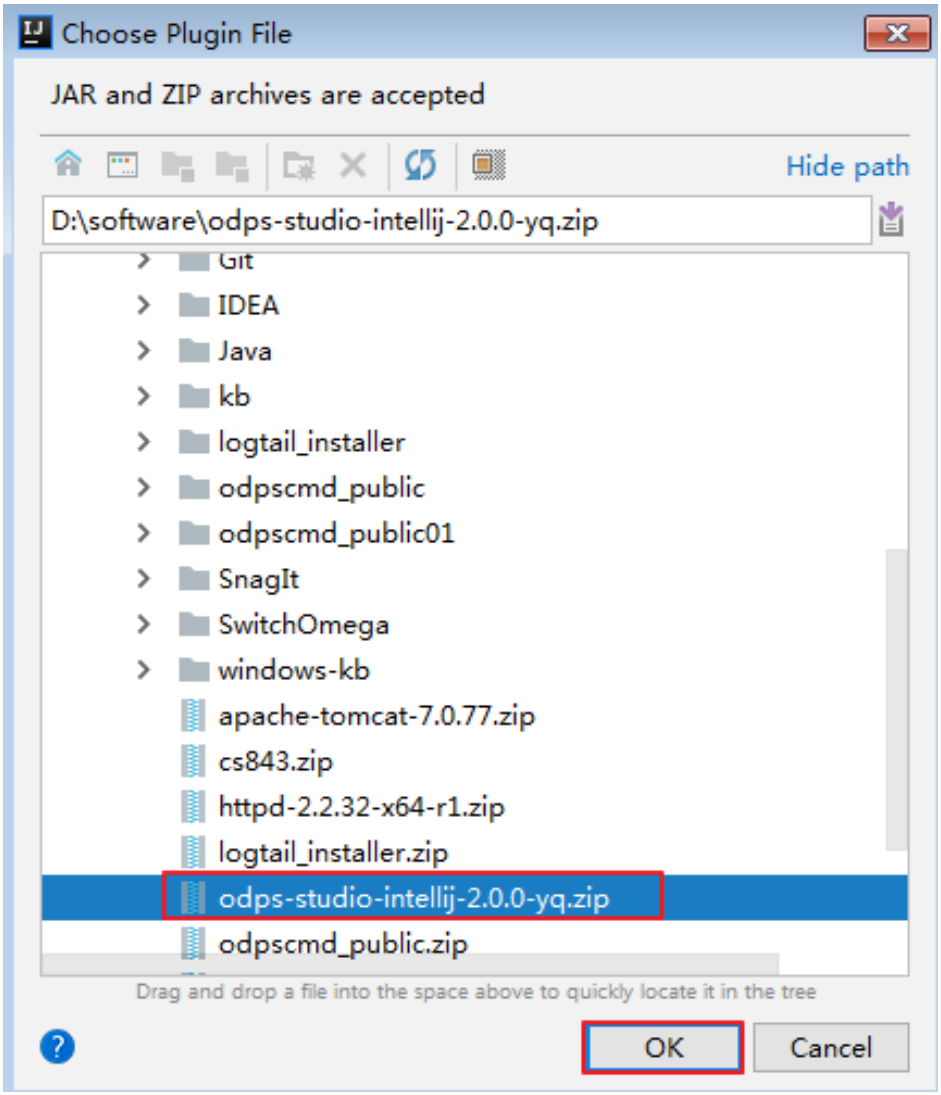




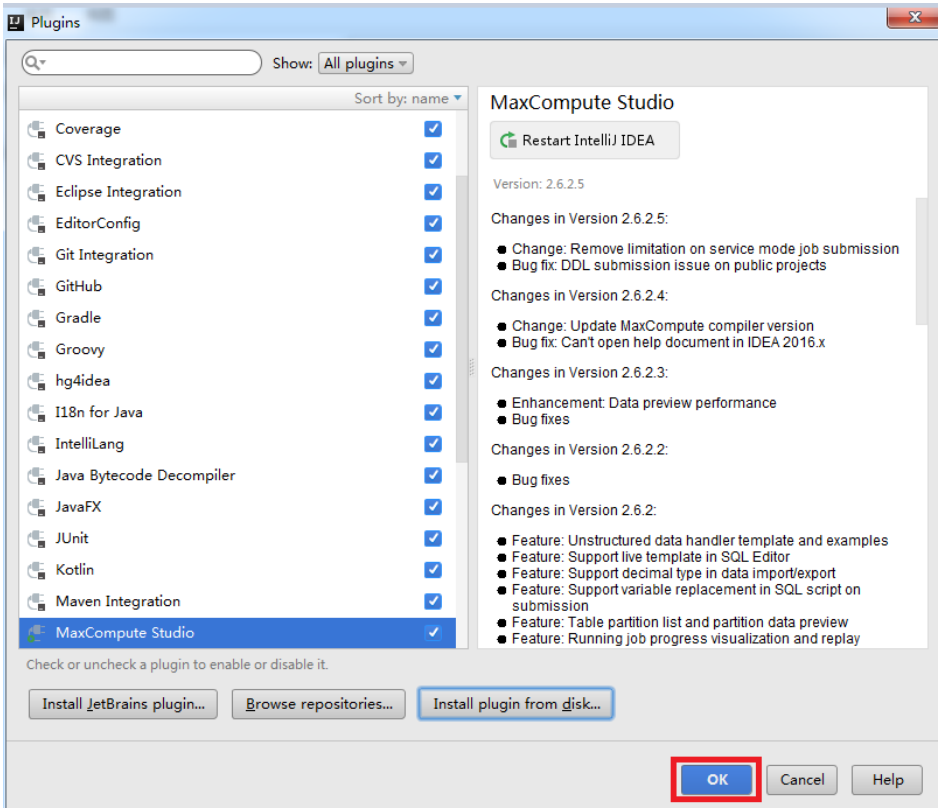
在插件页面，单击 **Install plugin from disk...**（从本地磁盘安装插件），如下图所示：



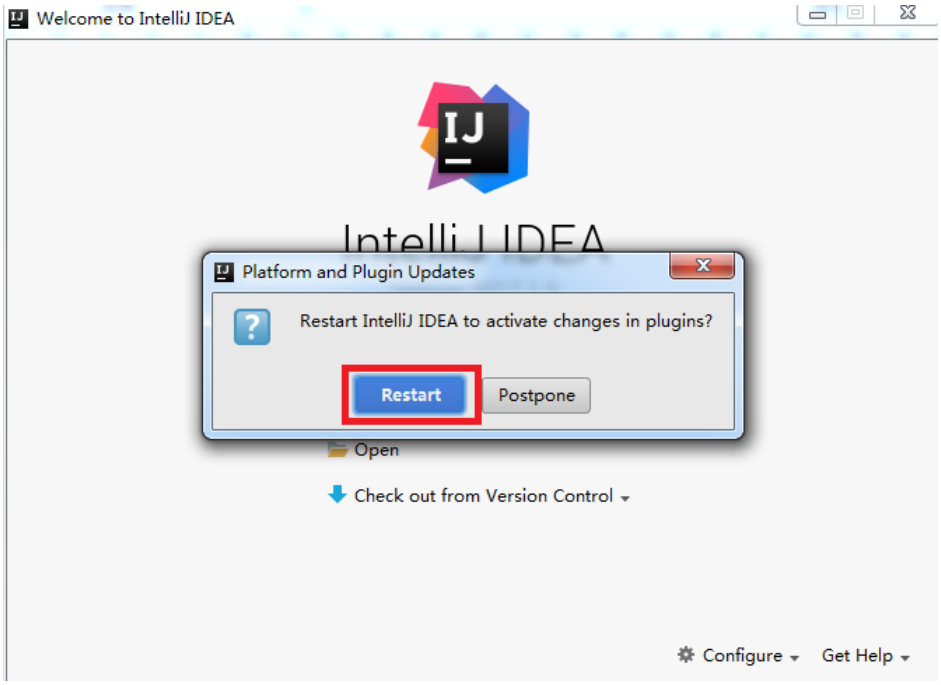
在弹出窗口中，通过单击目录名称前的灰色图标进行导航，找到插件文件并选中，单击 **ok**。



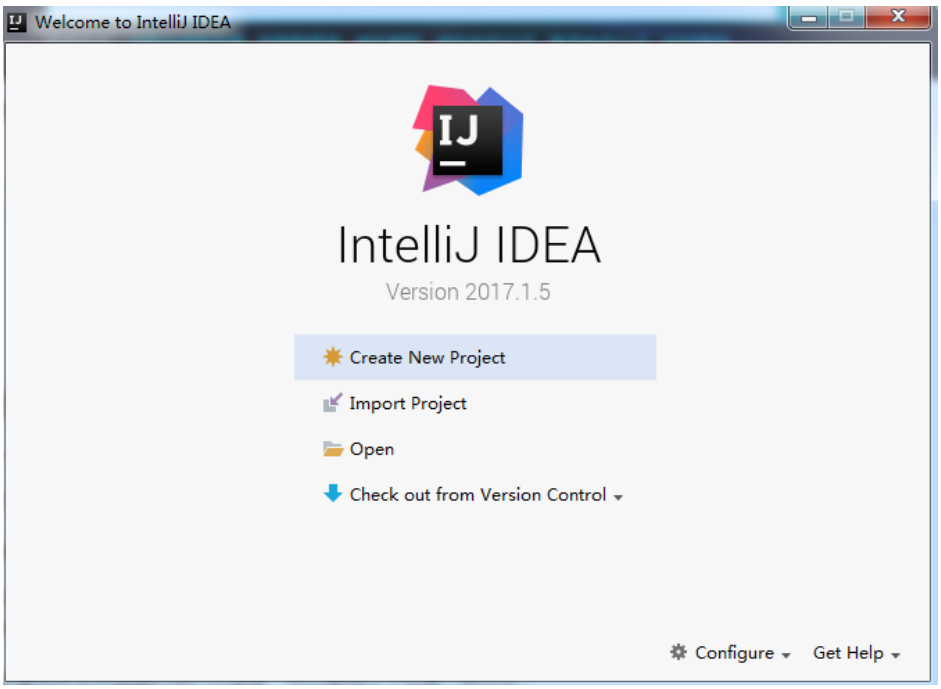
回到插件首页后，单击 OK，开始安装本地插件。



安装完成后，弹出重新启动的提示窗口，单击 **Restart**，重新启动 IntelliJ IDEA。



重新启动后，界面如下所示：



后续步骤

现在，您已经学习了如何安装 MaxCompute Studio 插件，您可以继续学习下一个教程。在该教程中您将学习如何配置 MaxCompute Project 连接管理数据和资源。详情请参见 新建 MaxCompute 项目空间连接。

查看 Studio 版本信息

通过以下步骤查看 Studio 的版本信息：

1. 打开 **Settings/Preferences** 页面（Windows Ctrl-Alt-S, Mac，）
2. 在对话框左侧边栏选择 **Plugins**，然后搜索 *MaxCompute Studio*
3. 可以看到 MaxCompute Studio 的版本号，以及版本的发布信息。

也可以在 Setting 页面左侧边栏选择 **MaxCompute Studio**， 在找到当前版本号。

检查新版本

缺省配置下，MaxCompute Studio 会自动检测新版本，当有新的可用版本时，会自动通知用户

收到更新提示后，用户可以选择：

- **安装**： 点击更新提示中的安装链接，将会自动下载并安装此新版本，安装完成后重启 IntelliJ IDEA
- **配置**： 点击更新提示中的配置链接，你可以配置是否自动检查新版本

如果关闭了自动更新功能，用户通过以下步骤可以检查 MaxCompute Studio 的版本更新并选择安装：

1. 打开 **Settings/Preferences** 页面 （ Windows Ctrl-Alt-S, Mac , ）
2. 在对话框左侧边栏选择 **MaxCompute Studio**
3. 在 Studio 配置页面上，点击按钮 **Check new versions**
4. 如果检测到新的可用版本，会提示用户新的版本号。 点击按钮 **Install new version** 可以安装，重新启动 IntelliJ 完成安装

可以通过 **Automatically checks for new versions** 复选框控制自动检查版本更新的开关。

下一步

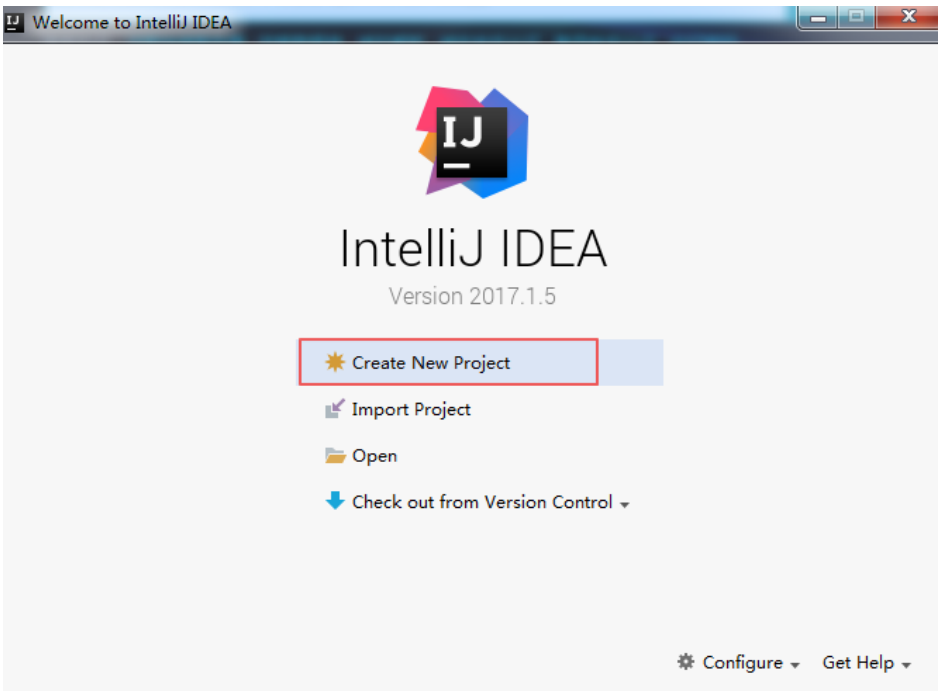
- 建立 MaxCompute 项目连接

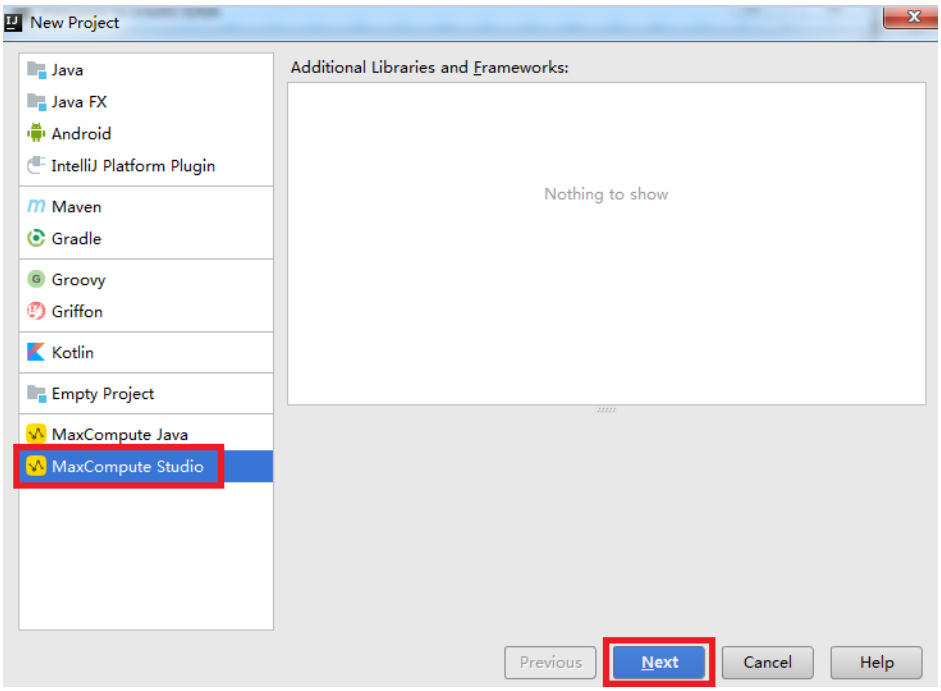
管理项目空间

MaxCompute Studio 的一大核心功能就是浏览 MaxCompute 项目空间（ Project ）的资源，包括 **Table**、**UDF**、**Resource** 等，要想实现这一功能，首先需要新建项目连接。

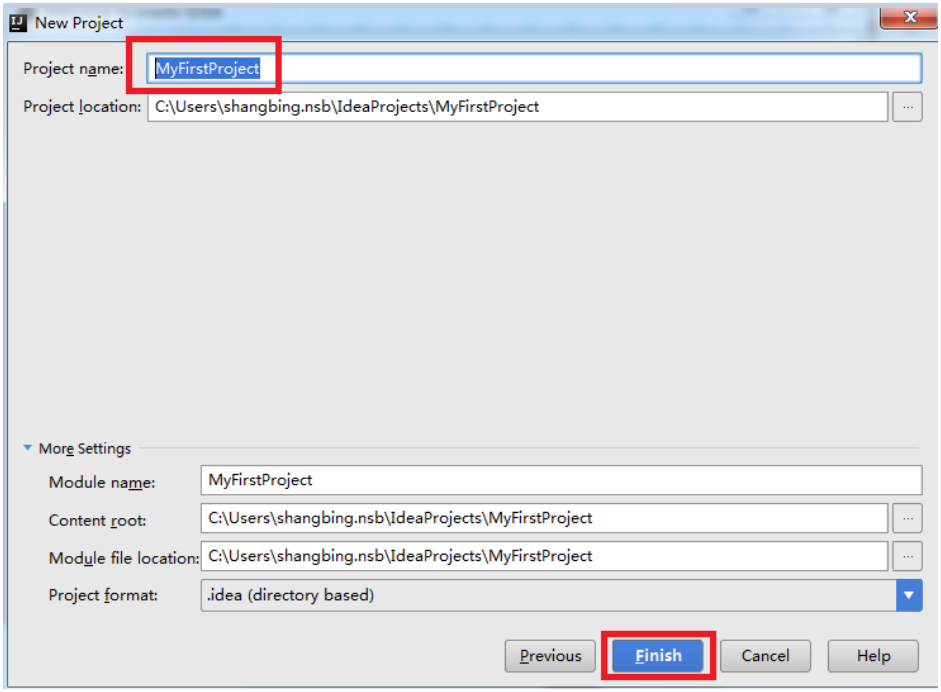
操作步骤

运行 IntelliJ IDEA 后，单击 **create new project**，选择弹出页面中的 **MaxCompute Studio**，单击 **Next**。如下图所示：

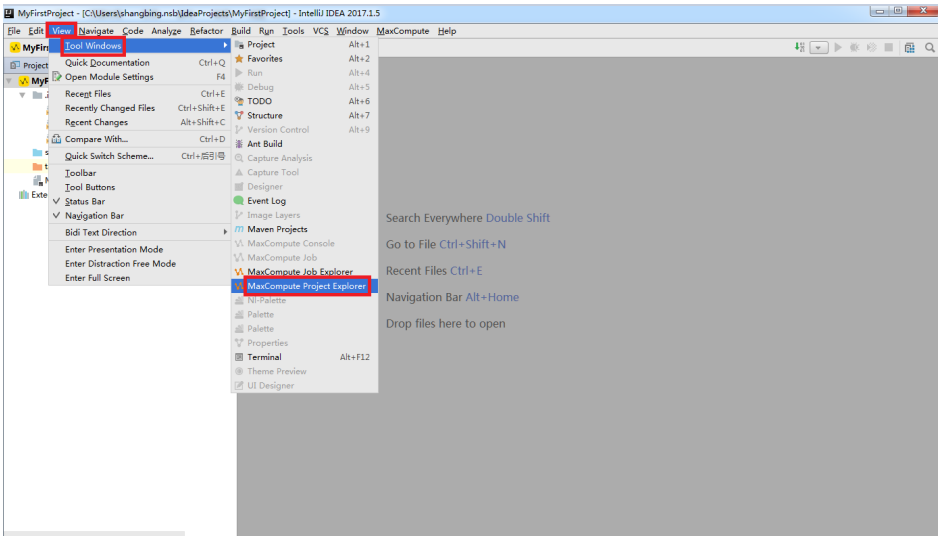




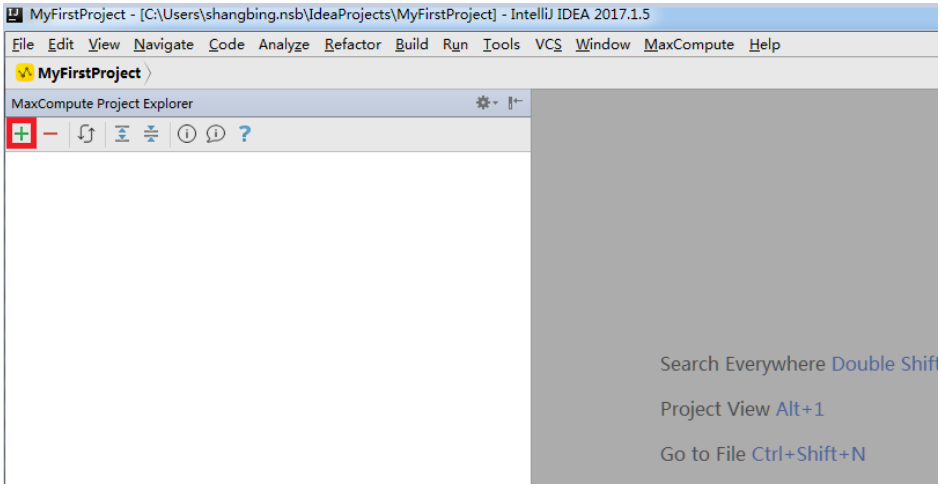
填写 Project name，单击 **Finish**。



单击菜单中的 **view** 选项，选择 **Tool Windows**，单击弹出页面中的 **MaxCompute Project Explorer**。如下图所示：

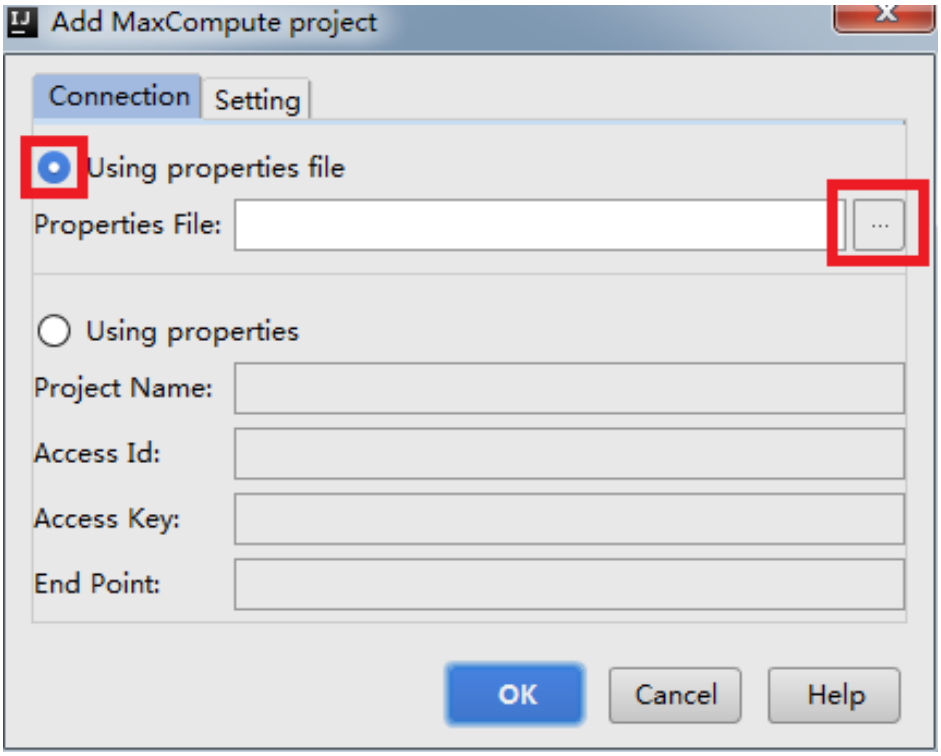


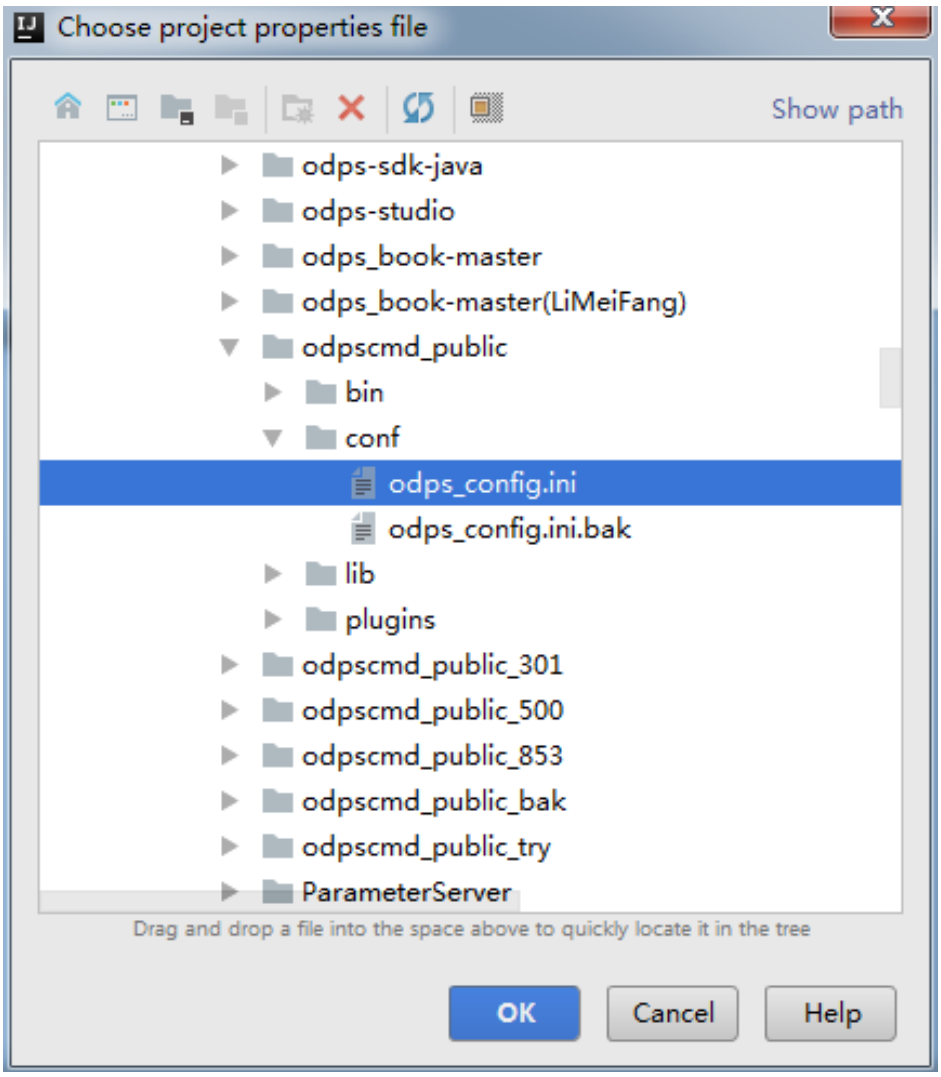
单击左上角的 **+**，添加一个 MaxCompute Project。如下图所示：



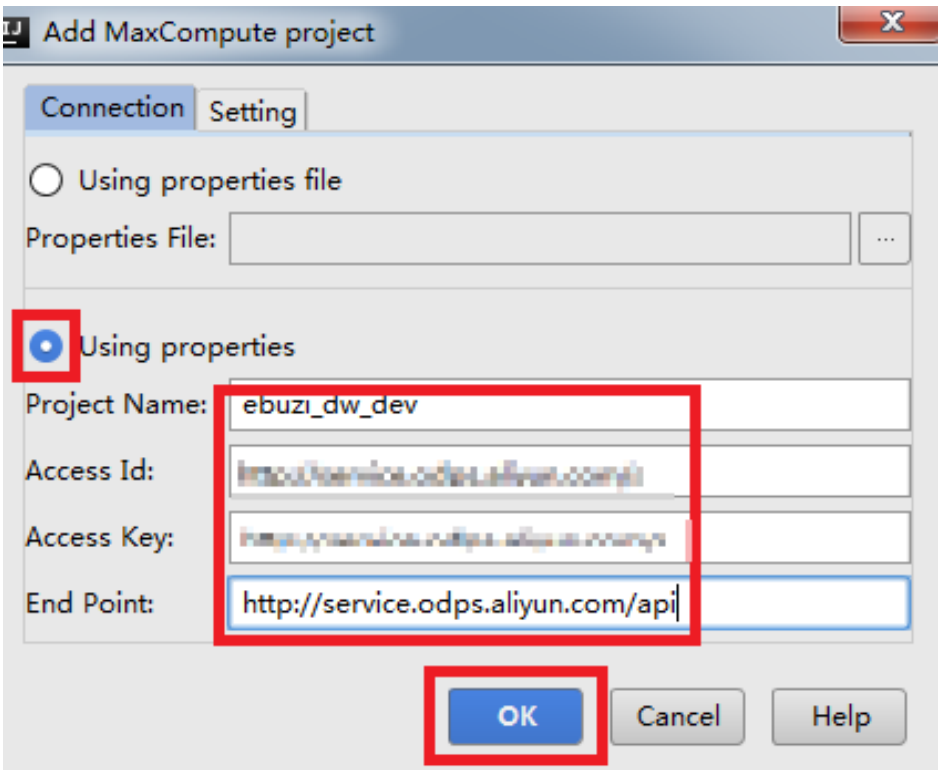
在 **Add MaxCompute Project** 对话框中，填入相关配置选项。有两种方式添加 MaxCompute 项目空间的连接：

Using properties file：如果当前机器上安装过 MaxCompute Console，可以选择 **Using properties file**，然后选择 MaxCompute Console 下的配置文件 **odps_config.ini**。





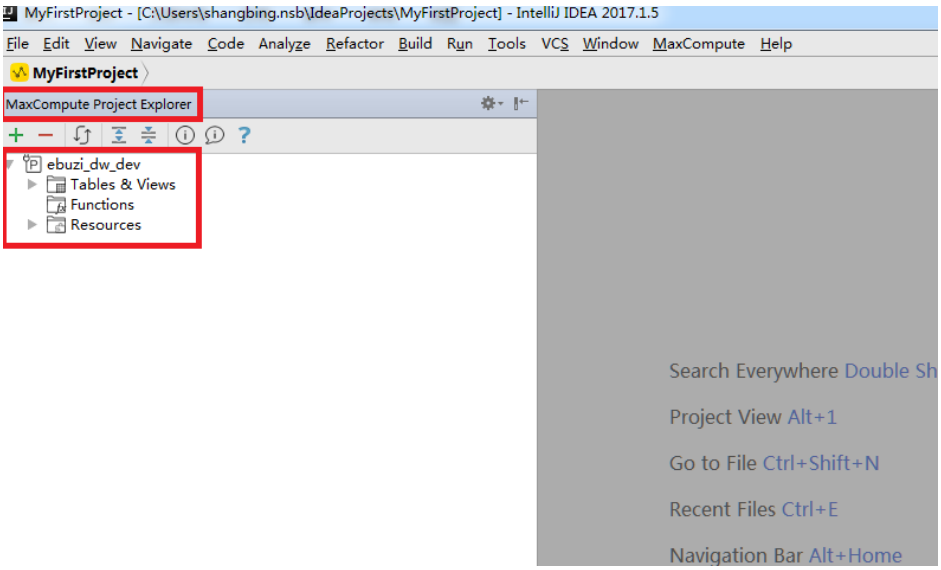
Using Properties：选择 **Using Properties** 后，填写 MaxCompute Project 的配置信息，包括 **Project Name**、**Access ID**、**Access Key** 以及 **Endpoint**。



注意：

实际上该操作以及配置的 MaxCompute Project 和之前新建的项目没有关系。MaxCompute Project 的配置需要进入 IntelliJ IDEA 界面后使用 Tool Windows 中的 MaxCompute Project Explorer，必须先打开一个IntelliJ Project，而这里的 IntelliJ Project 可以任意选择，它与 MaxCompute Project 没有任何关联。

配置完成后，左侧 MaxCompute Project Explorer 中会显示 MaxCompute Project 的信息，可以通过鼠标单击查看该 project 中的表、视图、函数以及资源等信息。



后续操作

现在，您已经学习了如何新建项目空间连接，您可以继续学习下一个教程。在该教程中您将学习如何进行元数据查询、清理数据、上传下载数据等操作，来管理数据和资源。详情请参见 管理数据和资源。

管理数据和资源

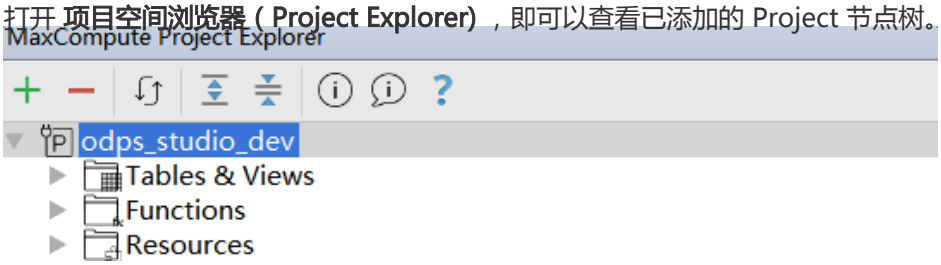
查看表及函数

在 **项目空间浏览器（Project Explorer)** 窗口中可以快速浏览已添加连接的表、函数、资源等。使用前提是 添加 MaxCompute 项目连接

浏览表和函数

要浏览项目空间的表和函数，使用以下步骤：

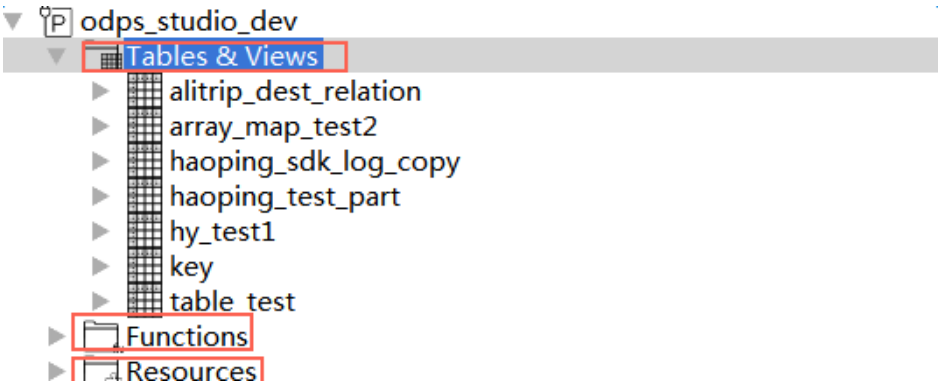
打开 **项目空间浏览器（Project Explorer)** ，即可以查看已添加的 Project 节点树。



节点树上方是工具栏，包括：

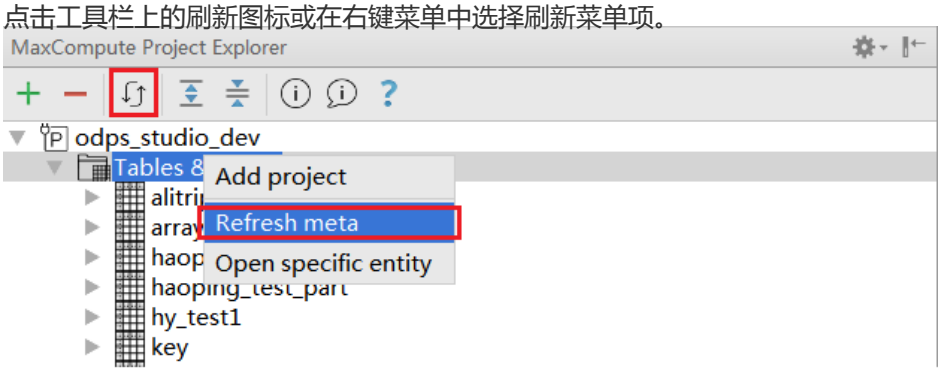
- **增加Project**：新增一个到 MaxCompute 项目空间的连接
- **删除Project**：删除一个项目空间浏览器中的项目连接，对服务器端项目空间无影响
- **刷新元数据**：从服务器端项目空间刷新元数据信息，刷新本地元数据缓存
- **展开节点**：展开全部树节点
- **折叠节点**：折叠全部树节点
- **用户反馈**：提交用户反馈
- **在线文档**：打开在线文档

双击或点击下拉箭头展开 **Tables** 节点，可列出该项目下的所有表（包括虚拟视图）。这里的表名列表与用户执行show tables命令相等，需要用户在project下有list table权限。函数（**Functions**）和资源（**Resources**）节点类似:



Studio 会将服务上的项目元数据下载到本地，当服务端元数据有更新时，如新增了一张表，需手动触发一次刷新，将变化的元数据重新加载到本地。 可以选择在项目（Project）或表（Table）级别做刷新，步骤如下：

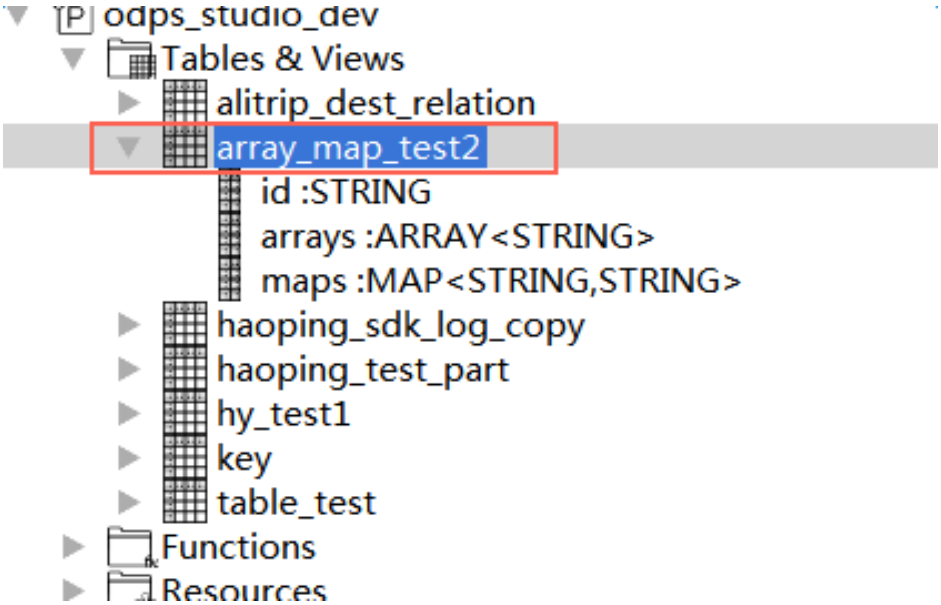
i. 选中相应的节点，
ii. 点击工具栏上的刷新图标或在右键菜单中选择刷新菜单项。



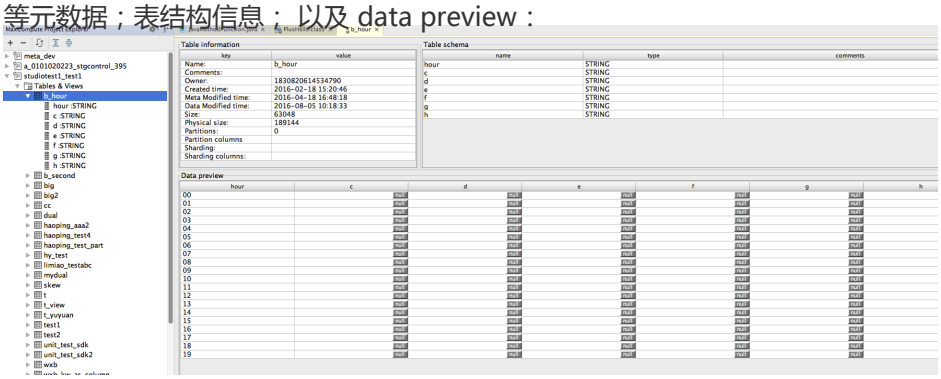
查看表详细信息

用户可以通过 Studio 的 **表详情视图（Table Details View）** 查看数据表相信信息

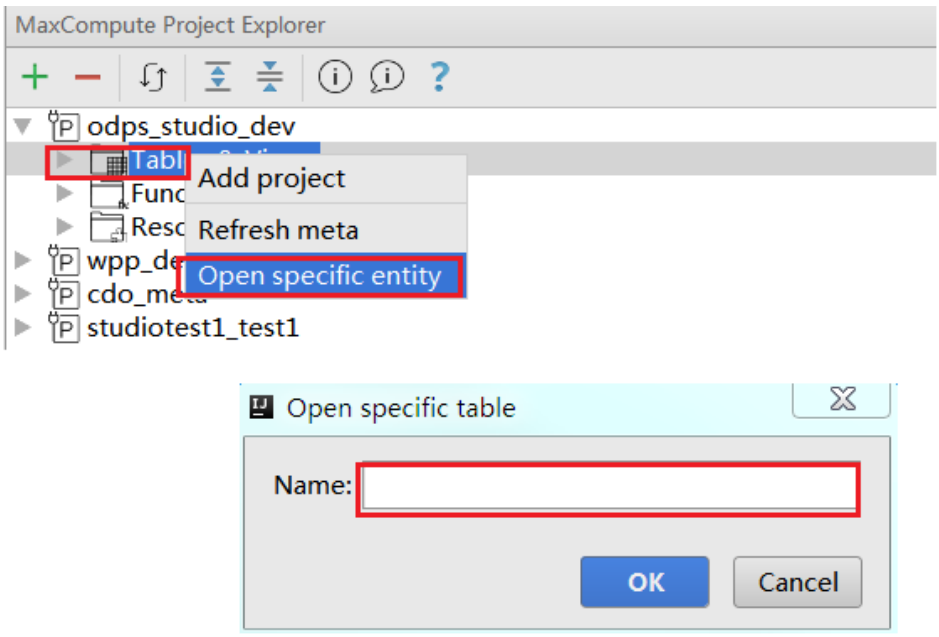
在节点树中，展开个表名节点，可快速查看列名和类型：



双击某个表或右键菜单 **Show Table Detail** 可以查看表的详细信息，包括 owner，size，column 等元数据；表结构信息； 以及 data preview：

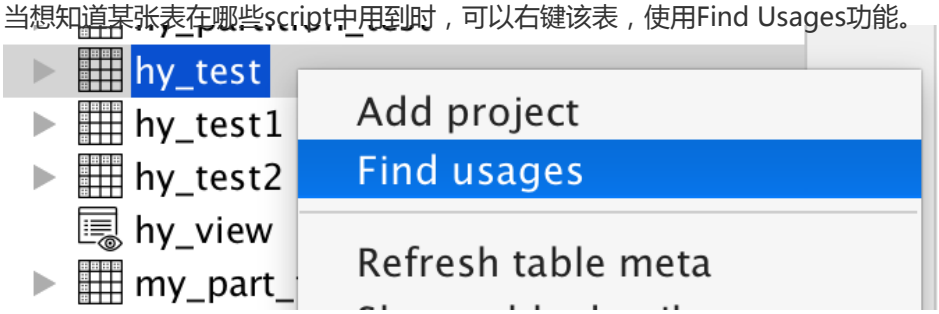


通过在Tables & Views右键菜单项 **Open specific entity** ，可以指定表名显示详情（注意要完整表名称）。另外如果用户没有project的list权限，而只有具体某张表的权限，也可以通过这种方式将该表抓取下来。函数（Functions）及资源（Resources）类似。



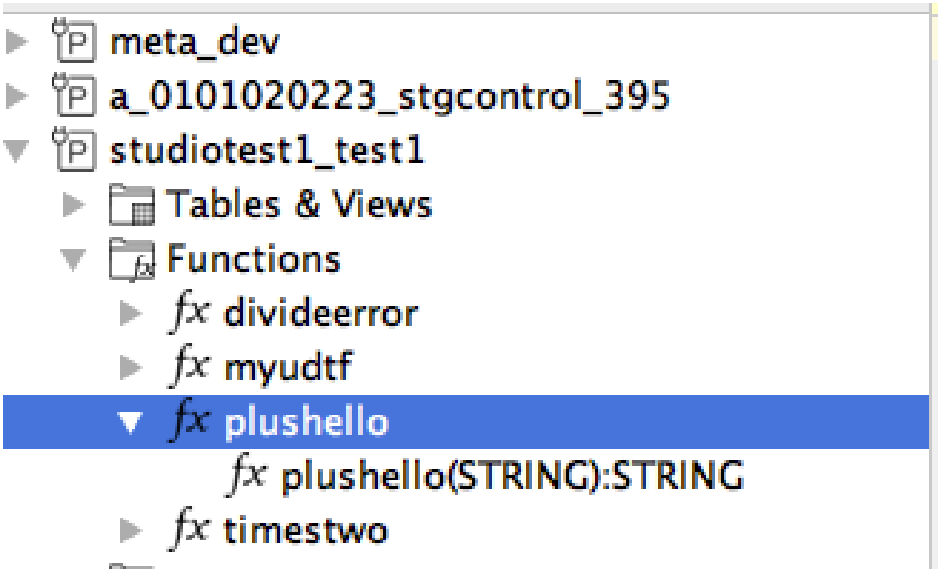
Intellij IDE 默认支持搜索，可展开表后直接敲击键盘模糊搜索。

当想知道某张表在哪些script中用到时，可以右键该表，使用Find Usages功能。



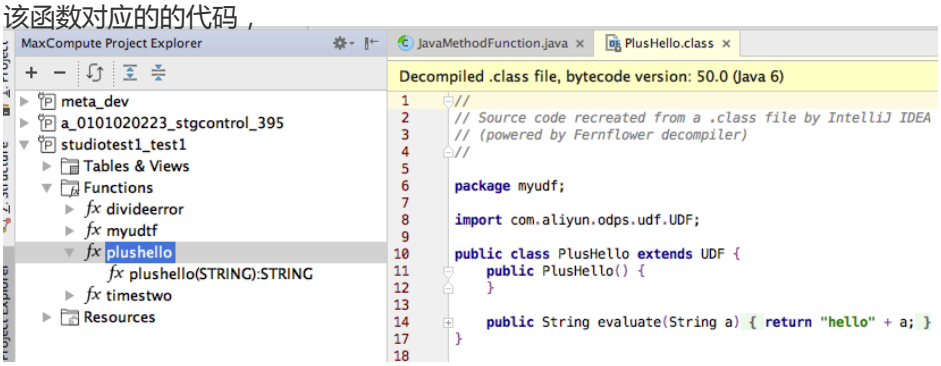
查看函数详细信息

在 **Functions** 树节点下可以展开某个函数节点，以显示该函数的方法签名



Python UDF解析签名需要安装pyodps(MaxCompute python sdk)，具体的先安装pip: sudo /usr/bin/python get-pip.py (请自行google下载get-pip.py)，然后安装pyodps: sudo /usr/bin/python -m pip install pyodps。需要注意的是mac系统自带一个python，位置在/usr/bin/python，请将pyodps安装在这个位置。

在 **Functions** 树节点下双击某个函数节点（或在 **Resources** 下双击改函数对应的源码资源）可打开该函数对应的的代码，



注意：Java代码通过反编译jar获取，并非源码。

导入导出表数据

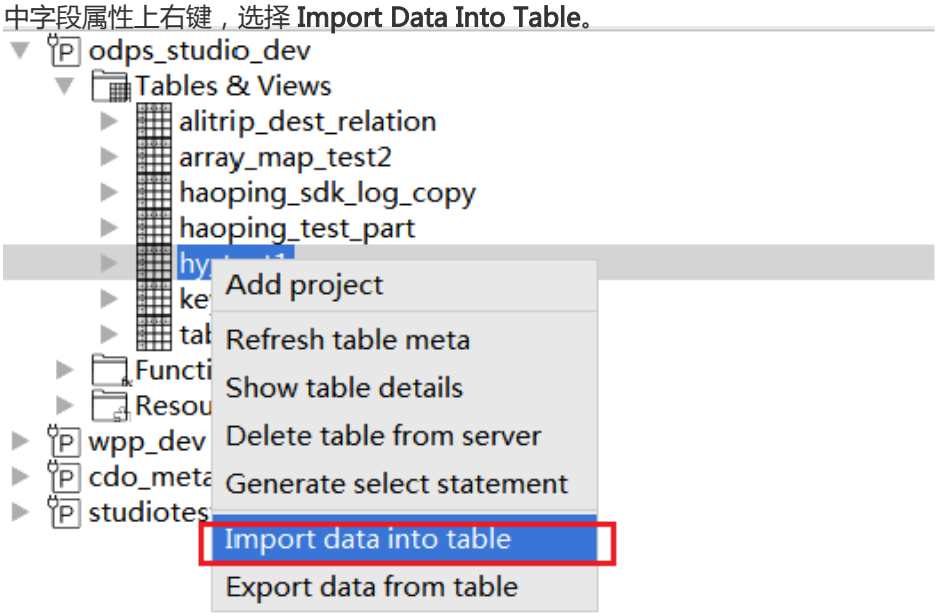
Studio 可以将 CSV，TSV 等格式的本地数据文件导入到 MaxCompute 表中，也可将 MaxCompute 中表数据导出数据到本地文件。Studio 是通过 MaxCompute 平台提供的**批量数据通道（Tunnel）**功能完成的

使用须知

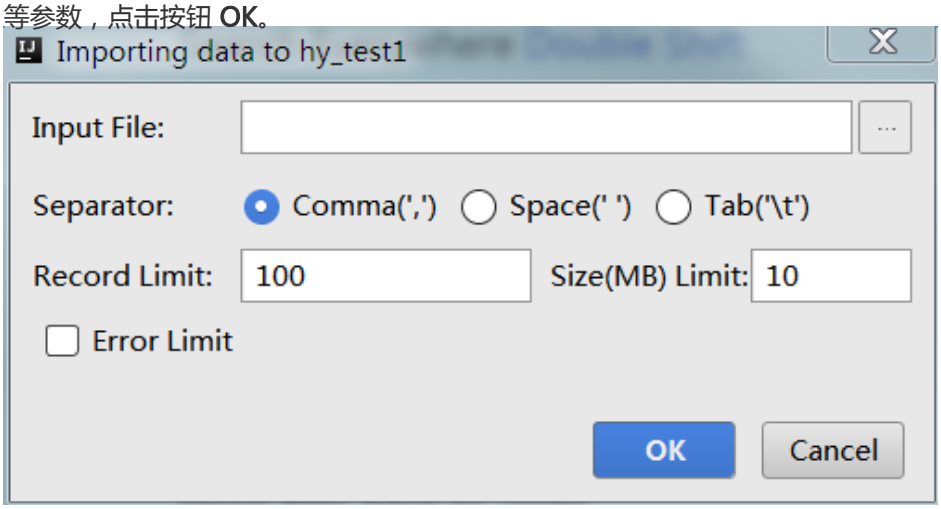
- *导入导出数据采用 MaxCompute Tunnel 服务，因此要求 Studio 中添加的 MaxCompute Project 必须开通或配置了 Tunnel服务。*
- *导入导出表必须具备相应权限。*

导入数据

打开 **项目空间浏览器（Project Explorer)** 窗口，在表名上点击右键或在表详细页面的 Data Preview 中字段属性上右键，选择 **Import Data Into Table**。



在弹出的 **Import Data** 对话框中，选择导入数据文件的路径，列分隔符，大小限制，错误容忍行数等参数，点击按钮 **OK**。

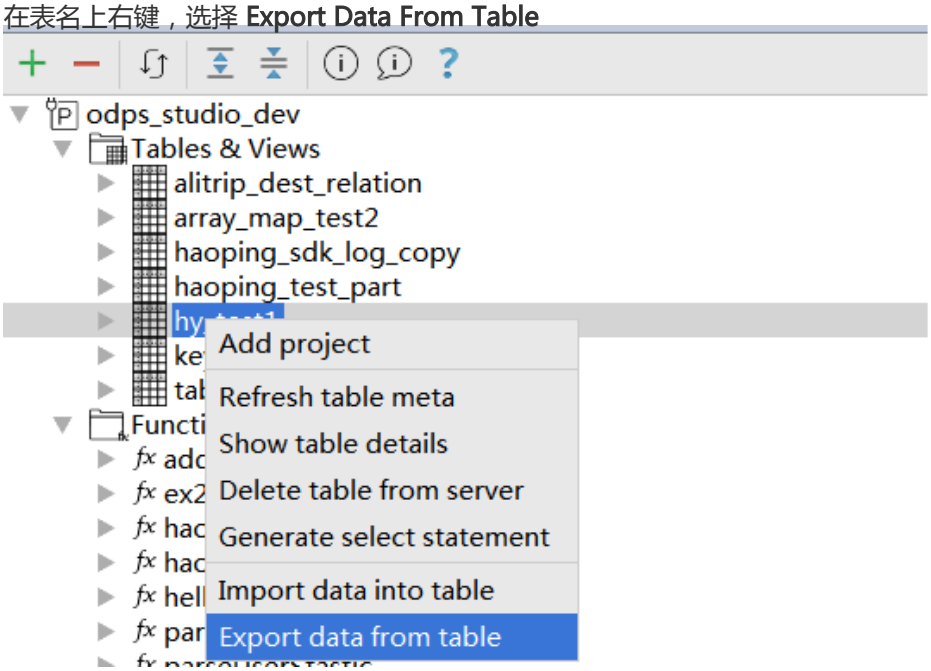


提示 **Import Data Success**，表示数据导入成功，可在表中查看导入的数据。

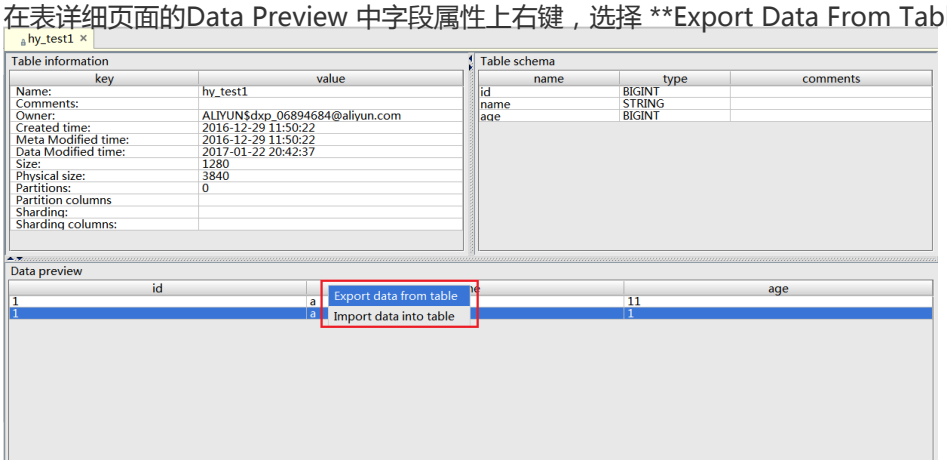
导出数据

1. 启动导出表数据有两种方式：

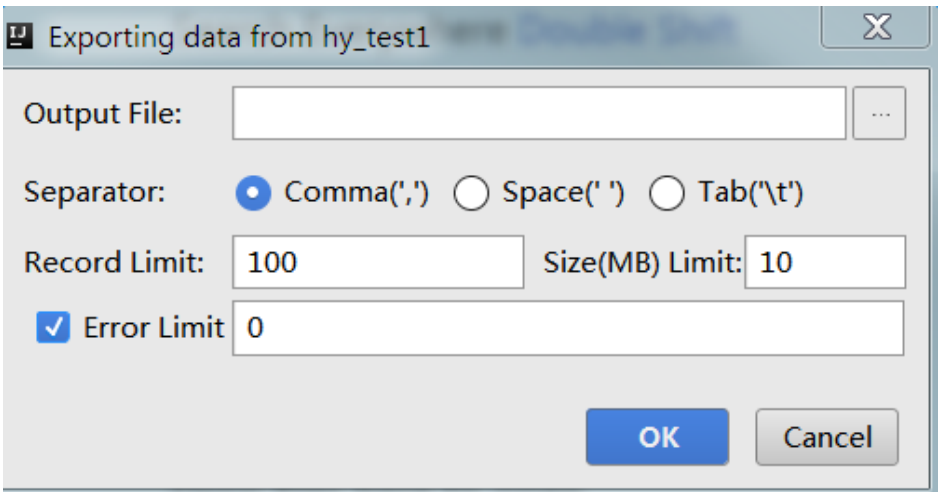
- 在表名上右键，选择 **Export Data From Table**



- 在表详细页面的Data Preview 中字段属性上右键，选择 **Export Data From Table

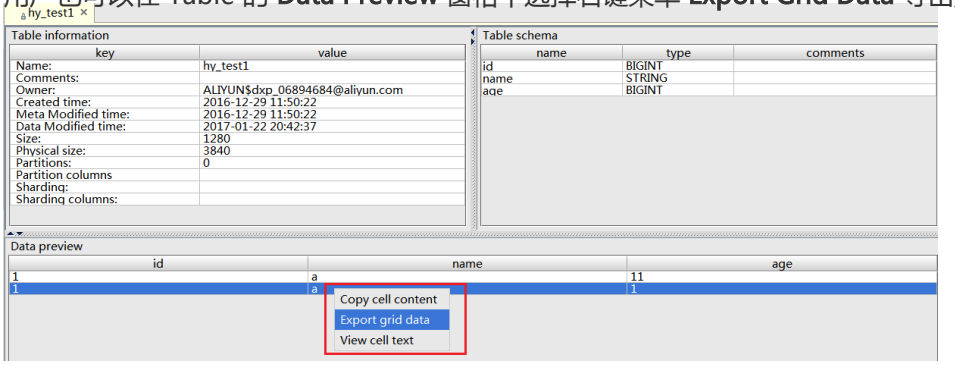


弹出Export Data对话框，选择导出数据文件的保存路径，列分隔符，大小限制，错误容忍行数等，填写完成后点击OK。



提示 **Export Data Success**，表示数据导出成功，可在目标文件中看到导出的数据。

用户也可以在 Table 的 **Data Preview** 窗格中选择右键菜单 **Export Grid Data** 导出数据。



> 注意：在 **Data Preview** 窗格的导出数据功能仅导出显示在数据示例（Data Sample），而不一定是表中的全部数据。

开发 SQL 程序

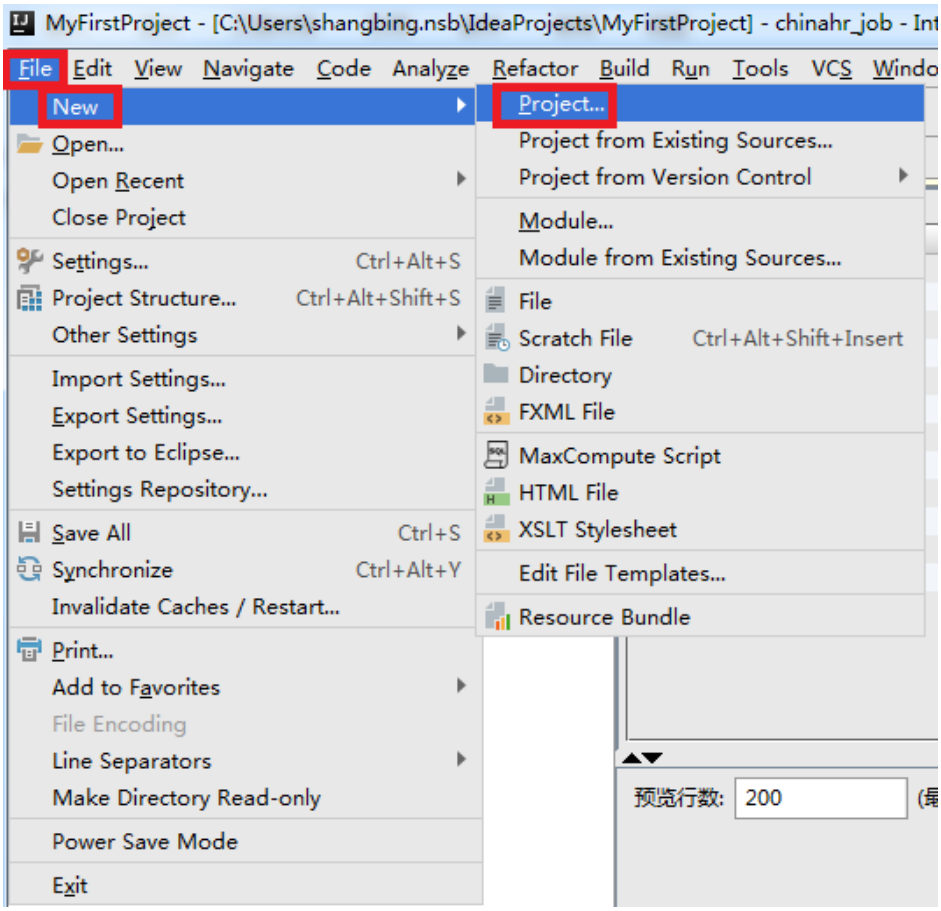
开发 MaxCompute Script 前，需要创建一个 MaxCompute Script Module，创建时存在以下两种情况：

本地没有 script 文件

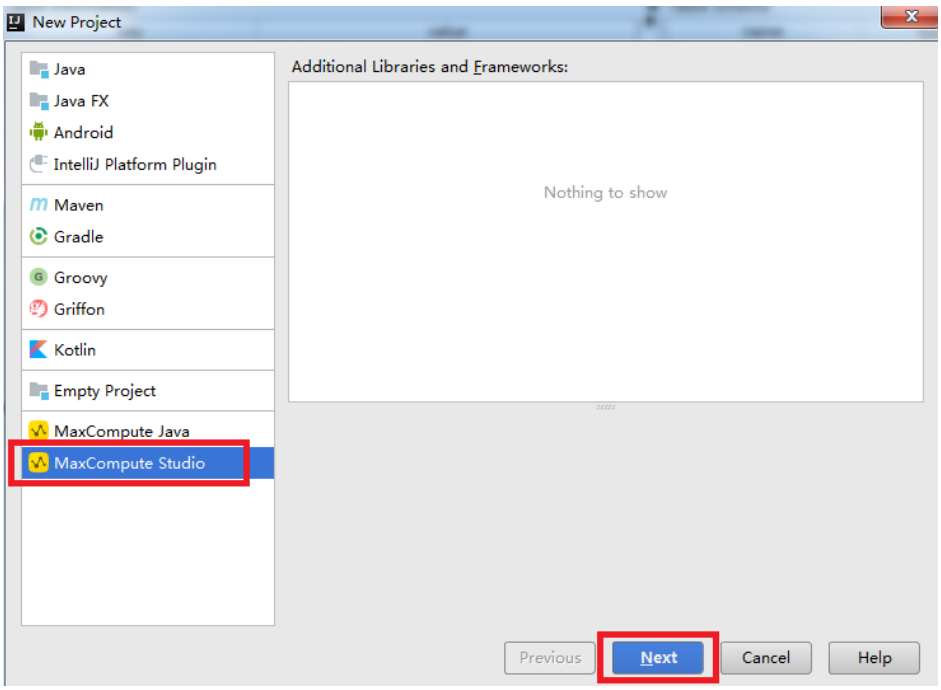
本地没有 script 文件时，可通过 Intellij 创建一个全新的 Module。

操作步骤

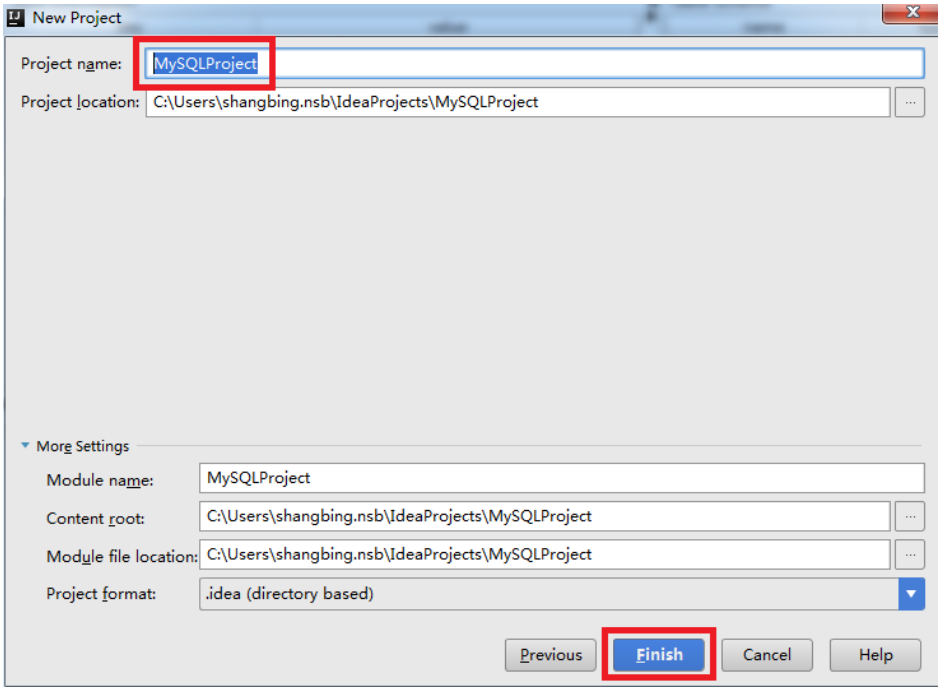
打开或者新建一个 MaxCompute Studio Project。本文以新建为例，单击菜单栏中的 **File**，导航至 **New > Project**。如下图所示：



选择左侧导航栏中的 **MaxCompute Studio**，单击 **Next**。

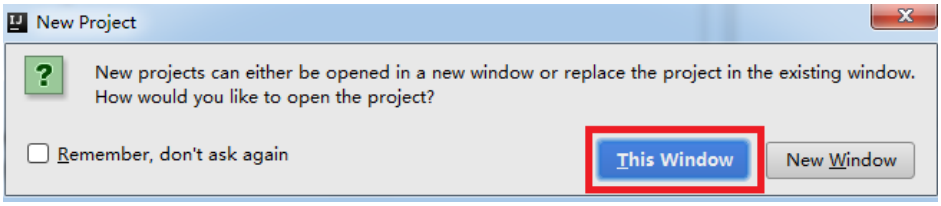


填写 Project Name，单击 **Finish**。

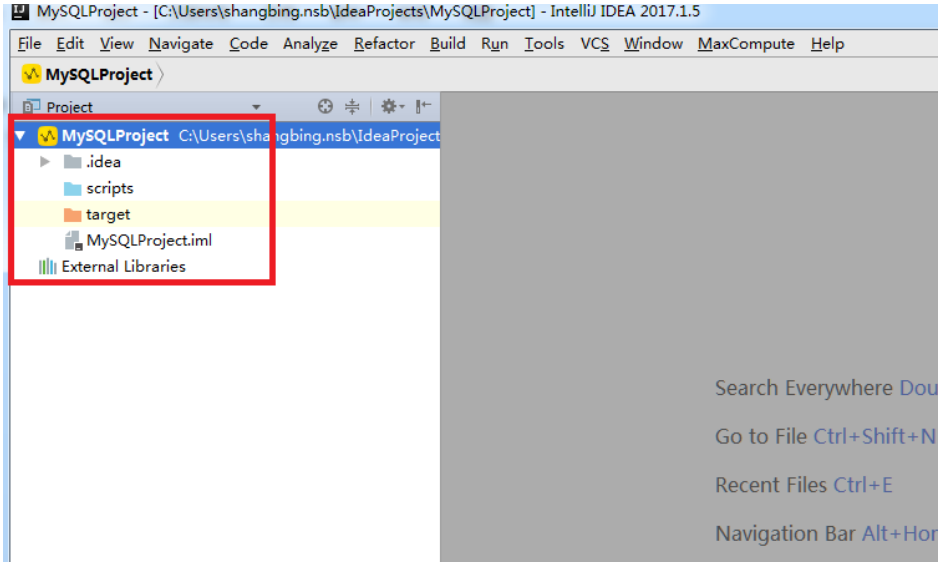


注意：

如果之前有打开的 Project，将会提示您是否在当前窗口中打开（即关掉之前的 Project），选择 **This Window**。



创建完成后，页面如下图所示，您即可在此项目中开发 SQL 脚本。



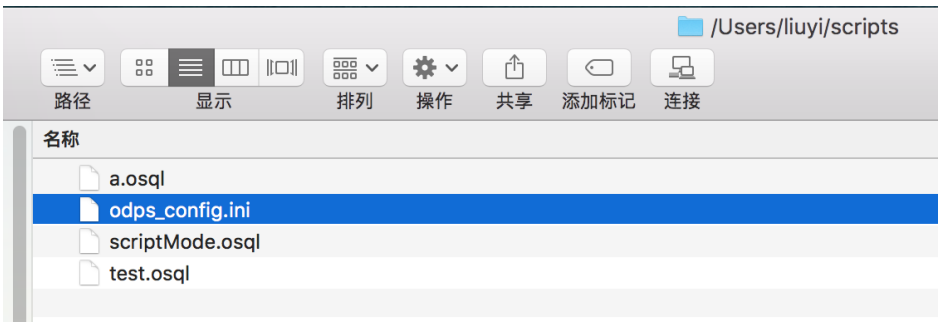
本地已有 script 文件

假如本地某个文件夹下已经有从服务器上下载下来的很多 script，此时需要用 MaxCompute studio 来编辑这些 script，您可直接打开一个 Module。

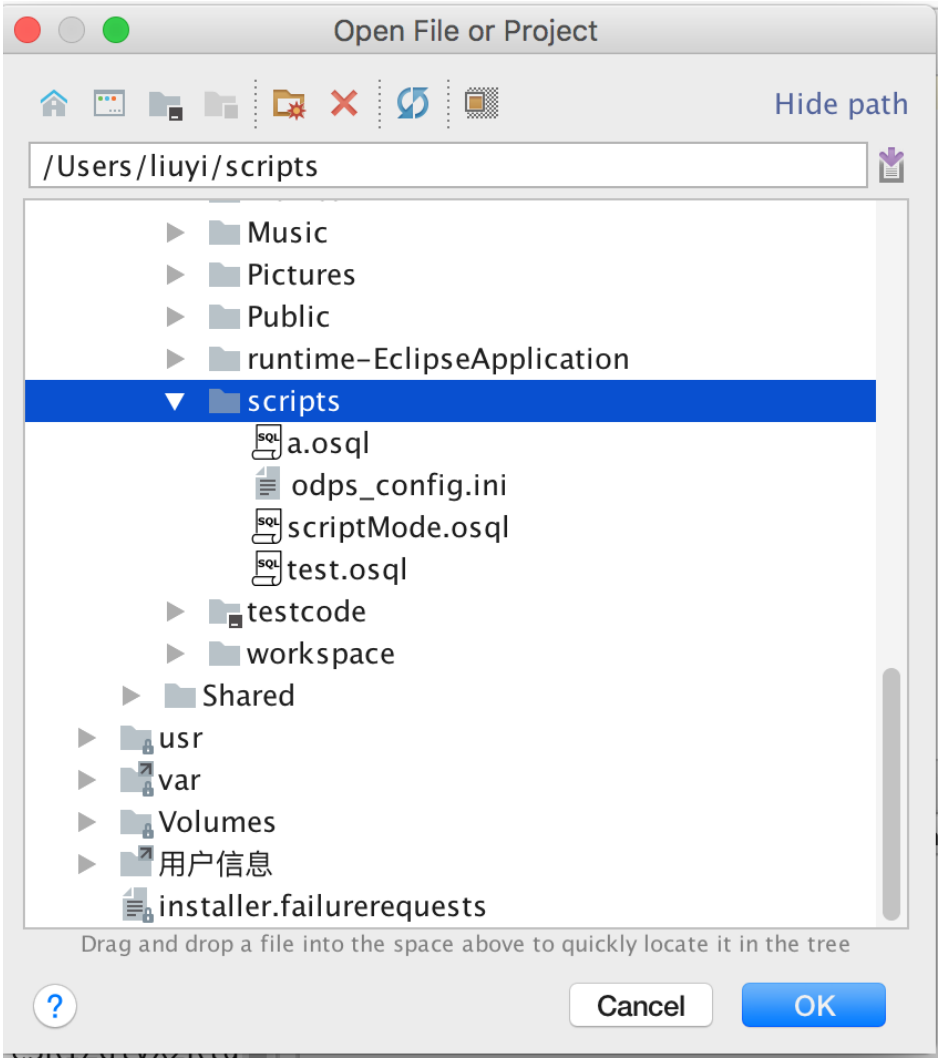
操作步骤

在 **scripts** 文件夹下创建一个 MaxCompute 的连接配置文件 odps_config.ini，在里面配置与 MaxCompute 连接的鉴权信息：

- project_name=xxxxxxxx
- access_id=xxxxxxxxxx
- access_key=xxxxxxxxx
- end_point=xxxxxxxxx



打开 IDEA，导航至 **File > open**， 选择您的 **scripts** 文件夹。

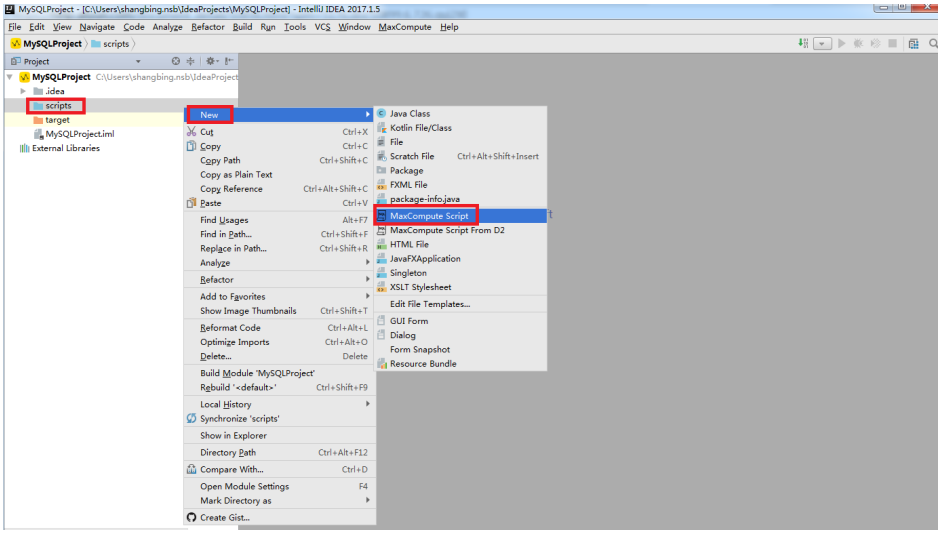


MaxCompute Studio 会探测该文件夹下是否存在 odps_config.ini 文件，根据这个文件里的配置信息抓取服务端的 meta，然后编译文件夹下的所有 script。

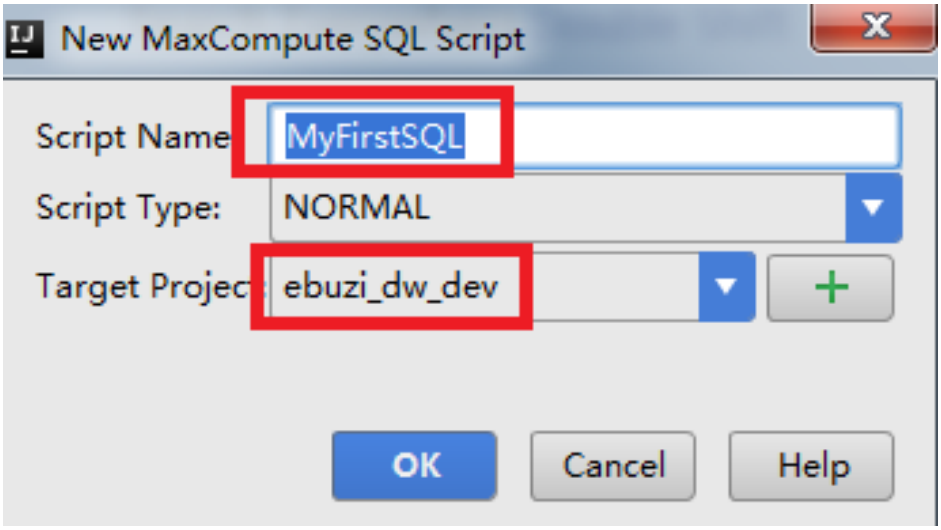
MaxCompute Studio 模块创建完成后，即可开始编写 MaxCompute SQL 脚本。

操作步骤

右击 scripts，导航至 New > MaxCompute Script。



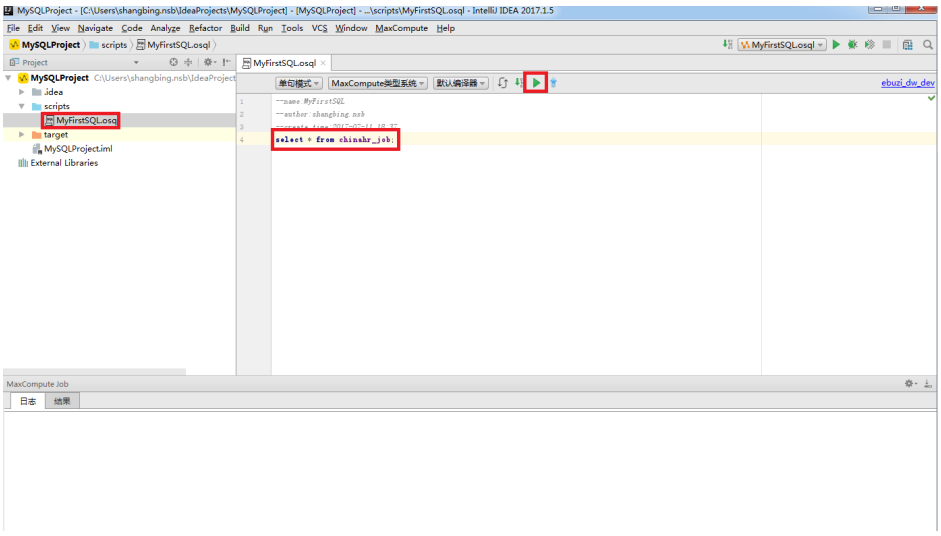
填写弹出框中的相关内容。



- Script Name：脚本名称。
- Script type：脚本类型。
- Target Project：目标 MaxCompute 项目。

上述窗口中，您也可以选择新建一个 MaxCompute Project 的配置，单击 **Target Project** 后面的 + 即可新建。配置详情请参见 新建项目空间连接。

在 SQL 文件编辑界面中，编写 SQL：select * from chinahr_job;。



注意：

实际 SQL 请根据自己的 MaxCompute Project 中的表进行编写。

MaxCompute Studio 功能

MaxCompute Studio 不仅提供语法高亮，智能提醒，错误提示等功能，还支持以下功能：

code folding： 可以将子查询等折叠起来，方便长 SQL 的阅读。

brace matching：鼠标单击高亮左括号，其匹配的右括号也会高亮，反之亦然。

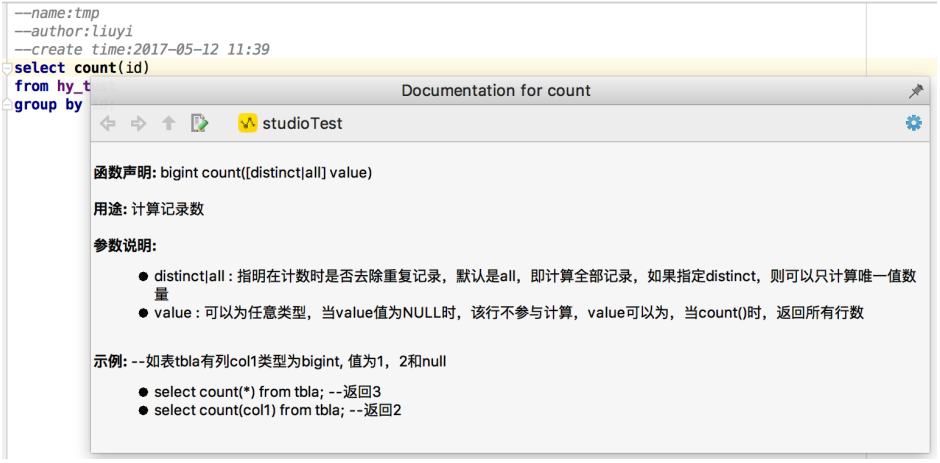
go to declaration：按住 Ctrl 键，单击 **table**，即可查看 table 详情。单击 **function**，即可显示其源码。

code formatting：支持对当前脚本格式化，快捷键（Ctrl + Alt + L）。

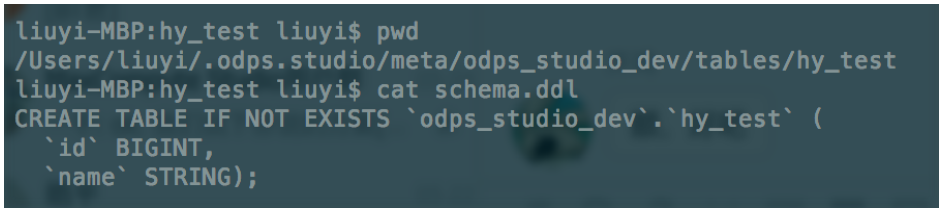
find usages：选中 editor 中的某张表（或函数），右键菜单选 **Find Usages**，则会在当前 IntelliJ project 下寻找所有使用该表的脚本。

live template：Studio 内置了一些 SQL live template，可以在编辑器中使用 Ctrl + J (Command + J on Mac OS X) 快捷键唤出（例如忘记了 insert into table 的语法，便可唤出 live template popup 后搜索 insert table）。

builtin documentation：支持在系统内置函数处通过 Ctrl + Q （Ctrl + J on Mac OS X）唤出帮助文档。



可单击 toolbar 右上角切换绑定的不同的 MaxCompute 项目，也支持跨 project 资源依赖。例如 script 绑定了 ProjectA，同时还会用到 ProjectB.table1，这时 Studio 会自动使用 ProjectA 的账号去抓取 ProjectB 的元数据。表的元数据 Studio 会保存在本机中类似下图的位置：



MaxCompute Studio 可直接将 MaxCompute SQL 提交到服务端运行，并显示查询结果，执行计划等详细信息。它在提交前会进行编译，能够有效避免提交到服务端后才发现编译错误。

前提条件

首先 创建 MaxCompute 项目连接，并绑定目标项目。

创建 MaxCompute Studio Module。

在提交前需根据自身需求进行相关设置。MaxCompute Studio 提供了丰富的设置功能，可在 Editor 编辑页面上方的 Tool Bar 工具栏中快速设置。设置主要分为以下三种：

编辑器模式：编译器模式设定包括两种模式，脚本模式和单步模式。

单步模式会将提交的脚本文件按；分隔，逐条提交到服务端执行。

脚本模式为最新开发模式，可将整条脚本一次提交到服务端，由服务端提供整体

优化，效率更高，推荐使用。

类型系统：类型系统主要解决 SQL 语句的兼容性问题，分为以下三种类型：

旧有类型系统：原有 MaxCompute 的类型系统。

MaxCompute 类型系统：MaxCompute 2.0 引入的新的类型系统。

Hive 类型系统：MaxCompute 2.0 引入的 Hive 兼容模式下的类型系统。

编译器版本：MaxCompute Studio 提供稳定版编译器和实验性编译器两种模式：

默认编译器：稳定版本。

实验性编译器：包含编译器最新特性。

ToolTips：脚本提交设置可采用 **全局设定**，导航至 **File -> Settings -> MaxCompute**，选择 **MaxCompute SQL**，其中 **Compiler -> Submit** 中可以设置以上属性。

提交 SQL 脚本

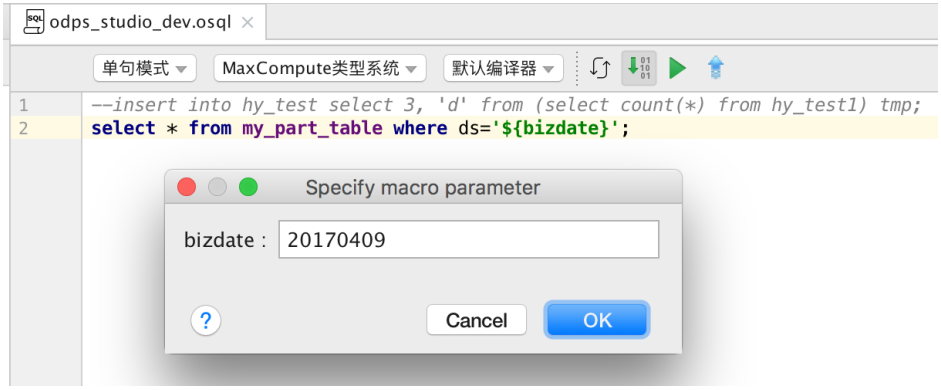
Editor 上方工具栏中提供同步，编译，提交功能。

* **同步功能**：更新 SQL 脚本中使用的元数据，包括表名、UDF 等。如果 Studio 提示表或函数找不到，而服务端又明确存在时，可尝试使用该功能更新元数据。

* **编译、提交**：按 MaxCompute SQL 预发规则编译或提交到服务端，编译错误会在 MaxCompute Compiler 窗口中显示详细信息。

具体操作

编写好 SQL 语句后，单击工具栏上的绿色运行图标，或者右击 Script Editor，选择 Run MaxCompute SQL Script，即可提交到服务端。当 SQL 中存在变量时（如下图的${bizdate}），会弹出对话框，提示您输入变量值。

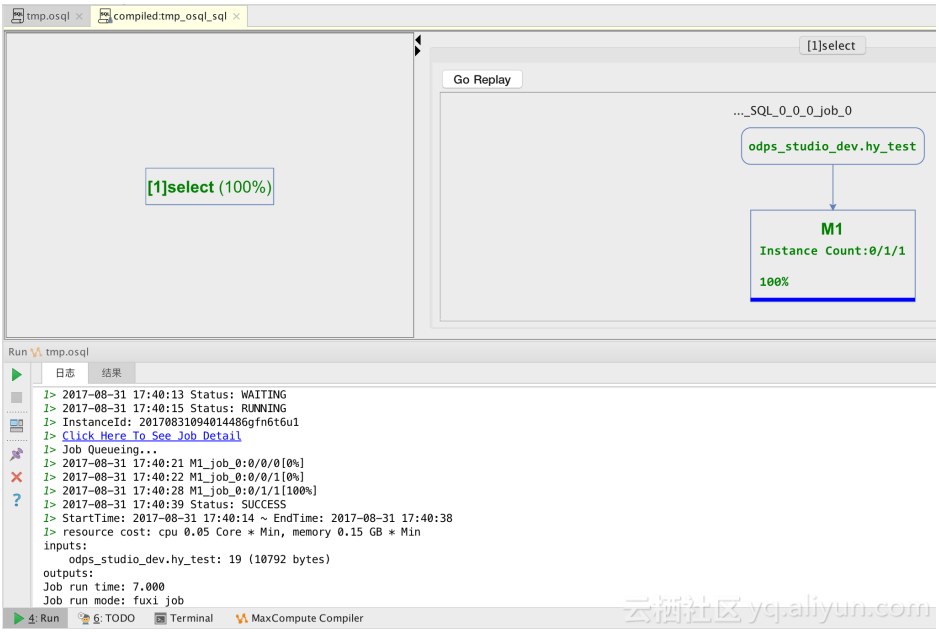


脚本会先被本地编译（依赖于您在 Project Explorer 窗口中添加的项目元数据），无编译错误后便会提交到服务端执行。Studio 提供了几个窗口来显示执行的详细信息：

MaxCompute Compiler 窗口：显示脚本的编译过程信息。

Compile Result 主窗口：显示脚本的 SQL statement 和 POT 图。当单句模式提交多条 SQL 语句时，会产生多个 Statement图。**右侧的 POT 图可双击某个TASK(如截图的M1)查看其详细执行计划。**

Run 窗口：日志标签页显示脚本在服务端执行过程中的进度信息；结果标签页显示脚本的执行结果。



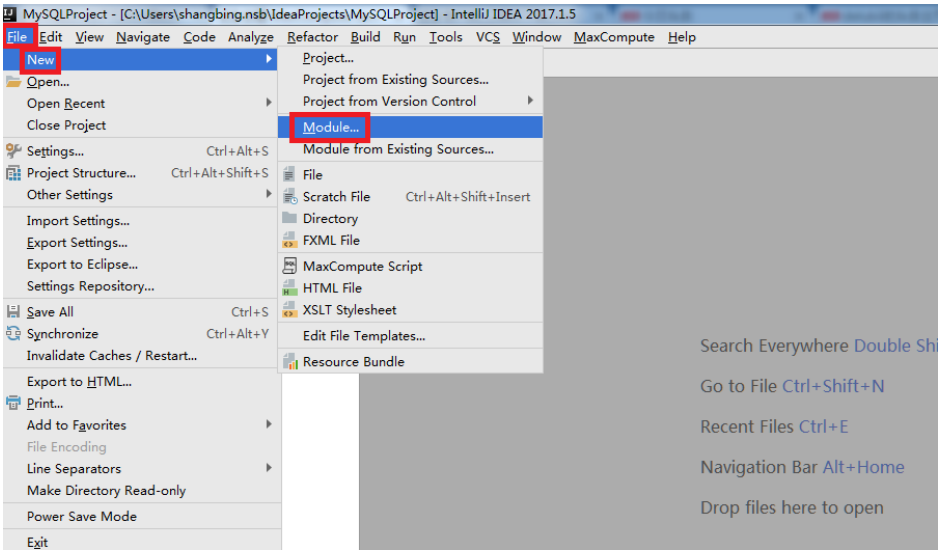
开发 Java 程序

MaxCompute Studio 支持开发 Java UDF 和 MR，首先需要新建一个 MaxCompute Java Module。

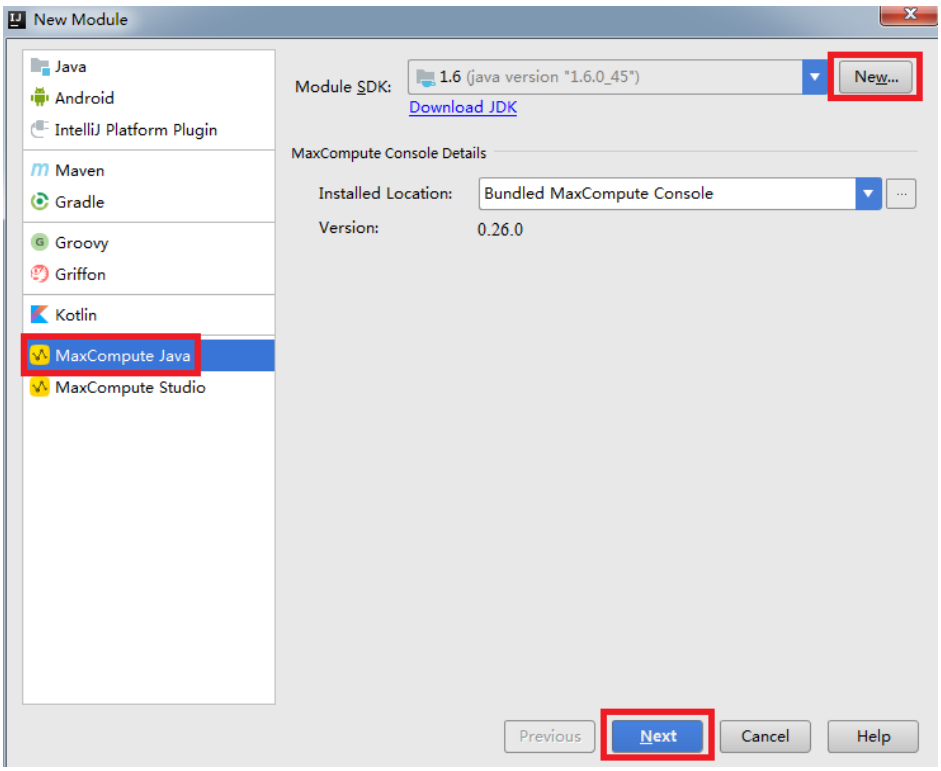
操作步骤

打开 IntelliJ IDEA，选择已创建的 MaxCompute Studio Project。

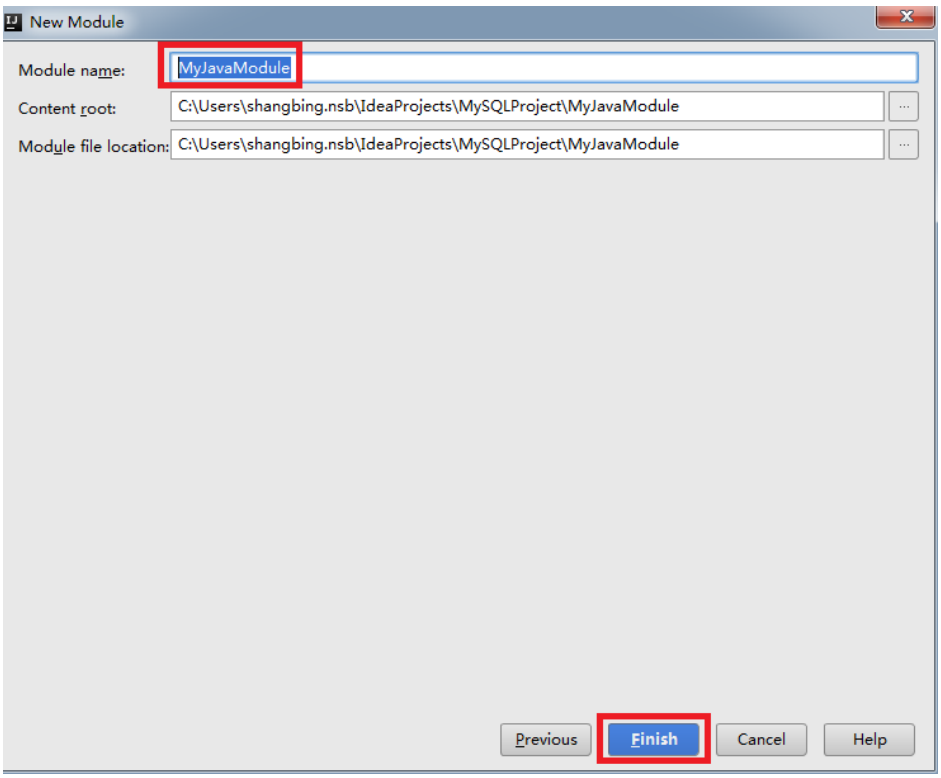
导航至 **File -> New**，单击 Module。如下图所示：



选择左侧导航栏中的 **MaxCompute java**，单击右侧页面中的 **New**，指定 java 目录（如果本地环境中未安装 java，请安装完成后再继续后面的步骤），单击 **Next**。



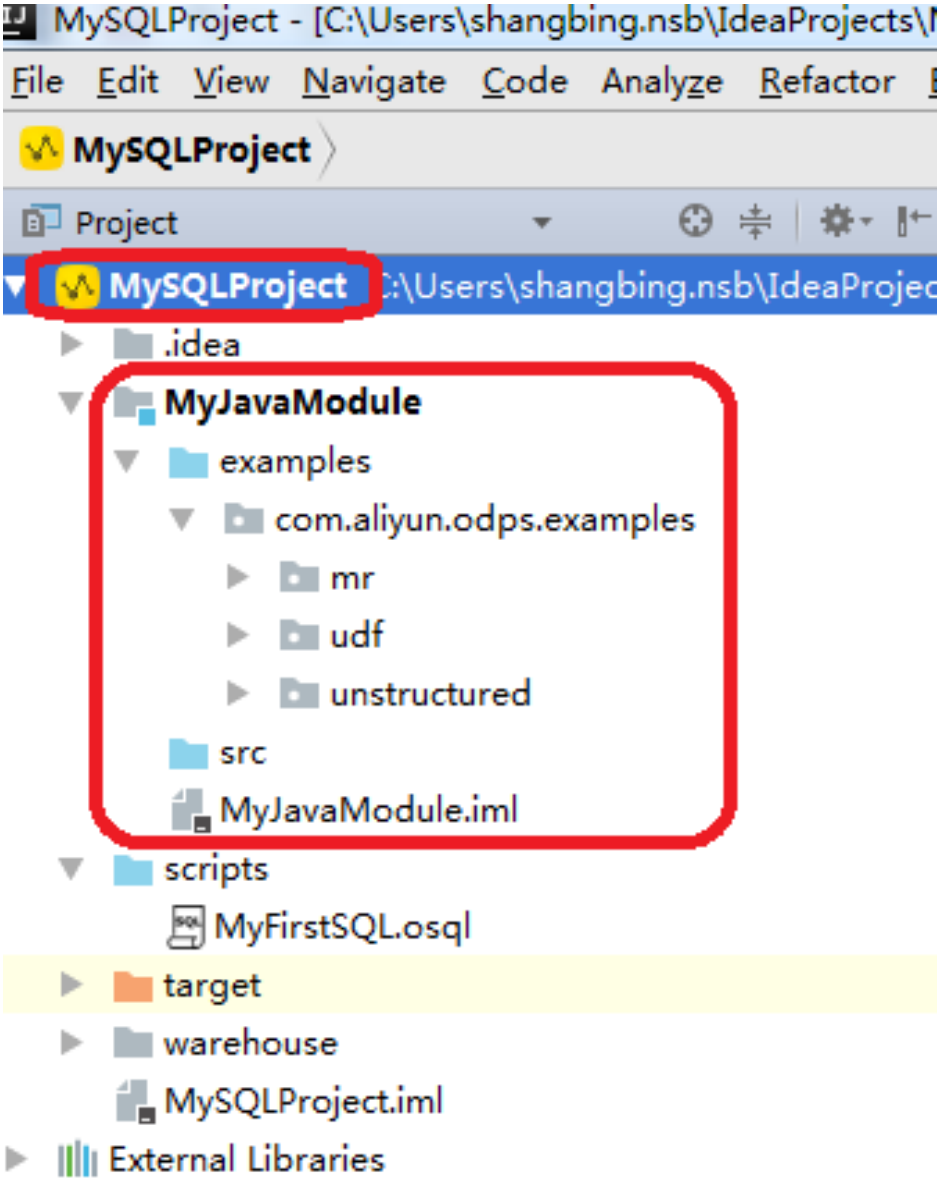
填写 New Module 页面的相关信息，单击 **Finish**。如下图所示：



完成以上操作，即成功创建了 MaxCompute Java Module。

Module 结构说明

MaxCompute Java Module 的结构如下图所示：



结构说明：

src/main/java：用户开发 java 程序源码。

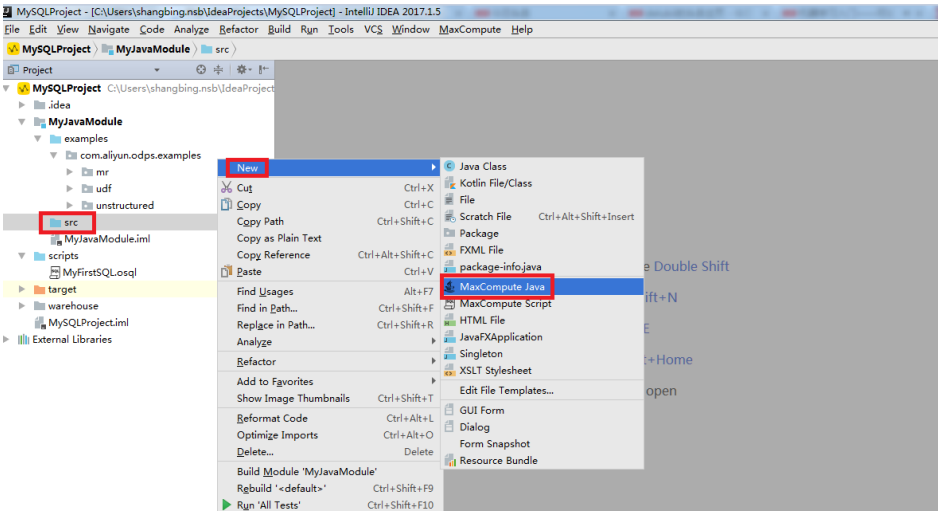
examples：示例代码，包括单测示例，您可参考这里的例子开发或编写UT。

warehouse：本地运行时需要的 schema 和 data。

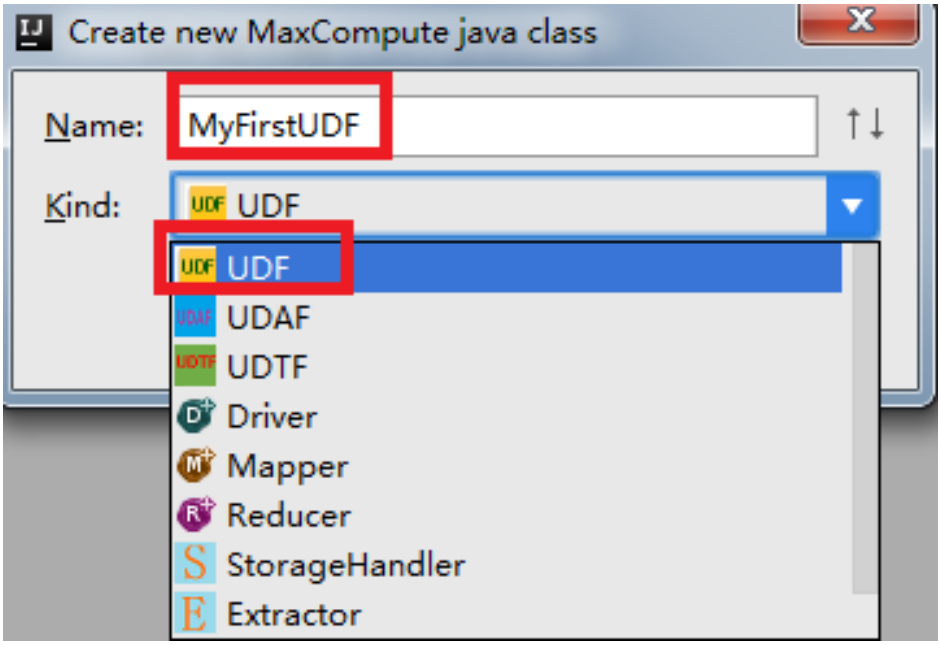
创建完成 MaxCompute Java Module 后，即可开发 UDF。

操作步骤

展开已创建的 MaxCompute Java Module 目录，导航至 **src -> new**，单击 MaxCompute Java。如下图所示：

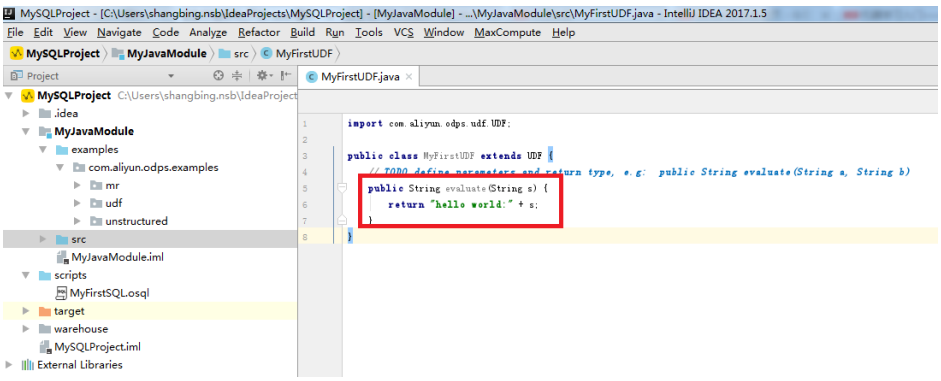


填写弹出框中的相关内容，单击 OK。如下图所示：



- Name：填写创建的 MaxCompute Java Class 名称。
- Kind：选择类型。目前支持的类型有：自定义函数（UDF/UDAF/UDTF）、MapReduce（Driver/Mapper/Reducer）、非结构化开发（StorageHandler/Extractor）等。

创建成功后，即可在 IntelliJ IDEA 中开发、编辑、测试 Java 程序。

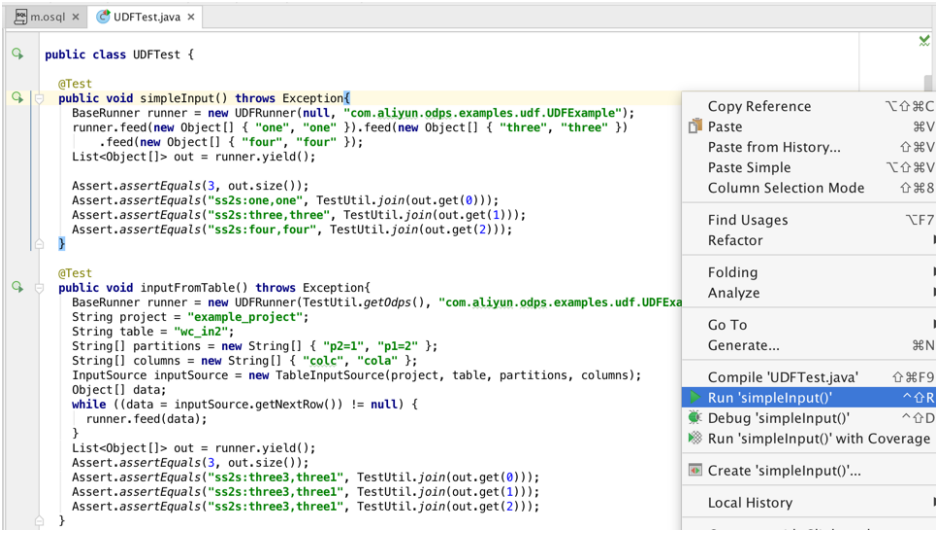


调试 UDF

开发 UDF 完成后，即可通过单元测试和本地运行两种方式进行测试，看是否符合预期。

单元测试

在 examples 目录下有各种类型的单元测试示例，可参考示例编写自己的 Unit Test。



本地运行

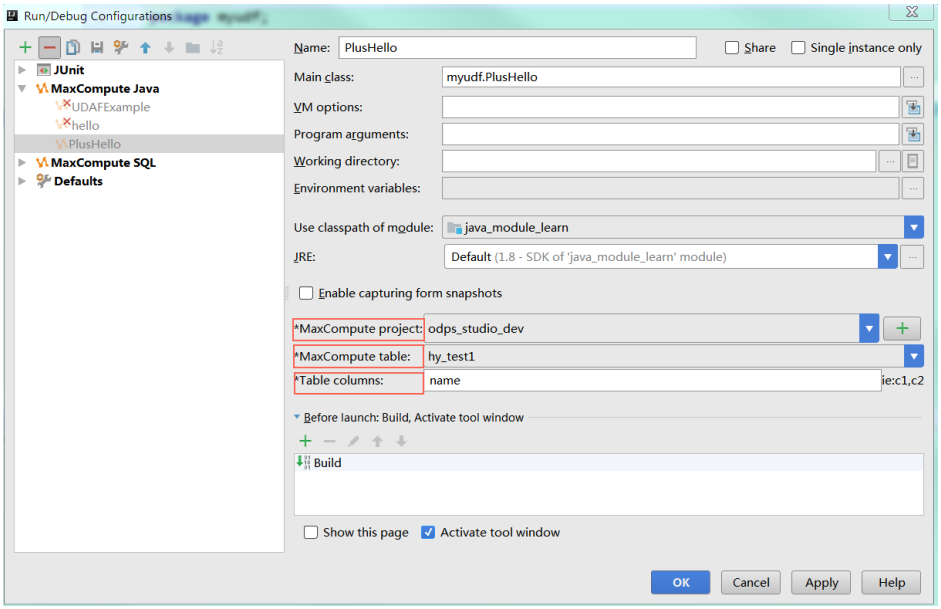
本地运行 UDF 时，需要指定运行数据源，有以下两种方式设定测试数据源：

MaxCompute Studio 通过 Tunnel 服务自动下载指定项目下的表数据到 warehouse 目录下。

提供 Mock 项目及表数据，您可参考 warehouse 下的 example_project 自行设置。

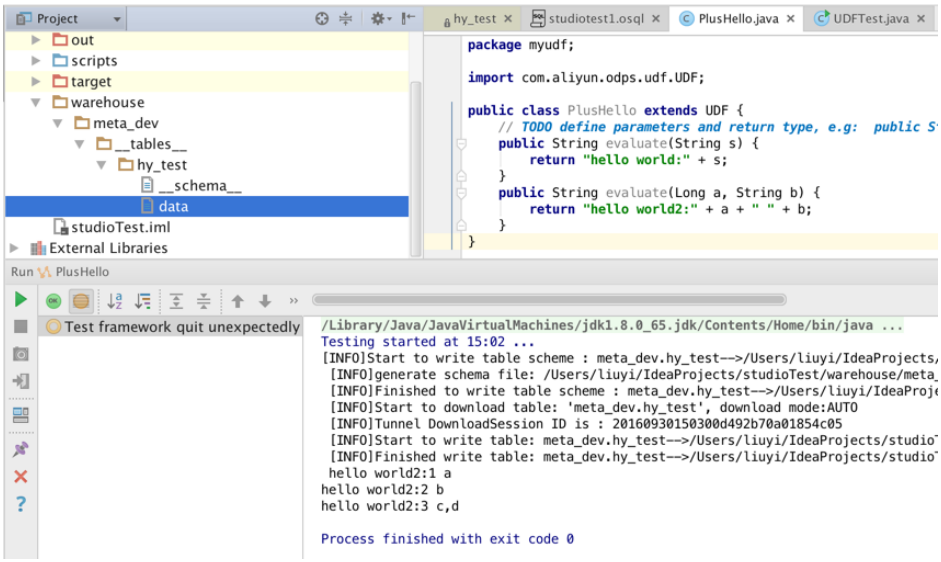
操作步骤

右击 UDF 类，单击 **运行**，弹出 run configuration 对话框。



UDF/UDAF/UDTF 一般作用于 select 子句中表的某些列，需要配置 MaxCompute project，table 和 column（**元数据来源于 project explorer 和 warehouse 下的 Mock 项目**）。

单击 OK。

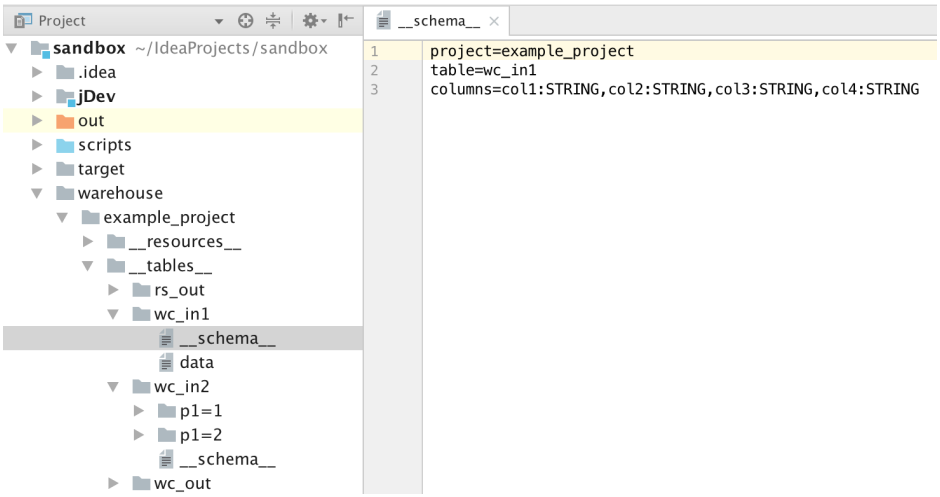


注意：

- 如果指定项目下的表数据未被下载到 warehourse 中，需要先下载数据。默认下载100条，如需更多数据，请自行通过console的tunnel命令或studio的表下载功能。
- 如果采用 Mock 项目或已下载数据，则直接运行。
- UDF 的 local run 框架会将 warehouse 中指定列的数据作为 UDF 的输入，开始本地运行 UDF，您可以在控制台看到日志输出和结果打印。

本地 warehouse 目录

本地 warehouse 用于存储表（包括 meta 和数据）或资源，用于本地执行 UDF 或 MR。warehouse 目录如下图所示：



注意：

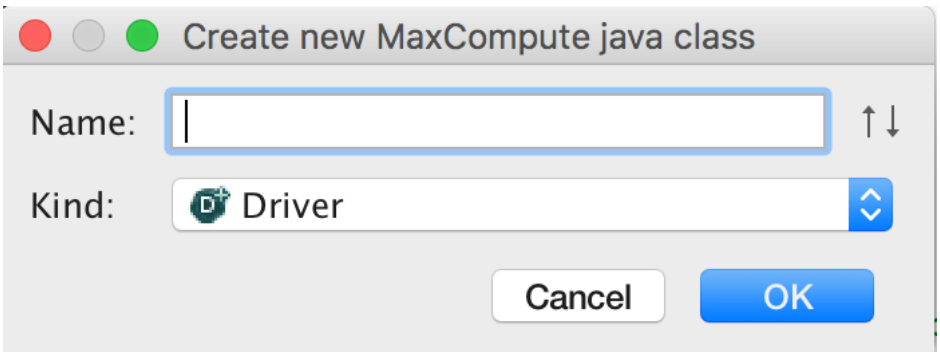
- warehouse 目录下依次是项目名，tables，表名，表 schema 和 sample data。
- schema 文件依次配置项目名，表名，以及列名和类型（冒号分隔），分区表还需配置分区列（非分区表参考 wc_in1，分区表参考 wc_in2）。
- data 文件采用标准 csv 格式存储表的 sample 数据：
  - 特殊字符为逗号，双引号和换行（\n 或 \r\n）
  - 列分隔符为逗号，行分隔符为 \n 或 \r\n
  - 如果列内容里包含特殊字符，需要在该列内容前后加上双引号，例如：3,No -> “3, No”
  - 如果列内容包含双引号，则每个双引号转义成两个双引号，例如：a”b”c -> “a””b””c”
  - \N 表示该列为 null，如果该列内容（string 类型）就是 \N，需要转换为 “”””\N””“”
  - 文件字符编码为UTF-8

创建完成 MaxCompute Java Module后，即可以开始开发MR了。

开发MR

1.在module的源码目录即src->main上右键new，选择MaxCompute Java。

2.分别创建Driver，Mapper，Reducer。



3.模板已自动填充框架代码，只需要设置输入/输出表，Mapper/Reducer类等即可。

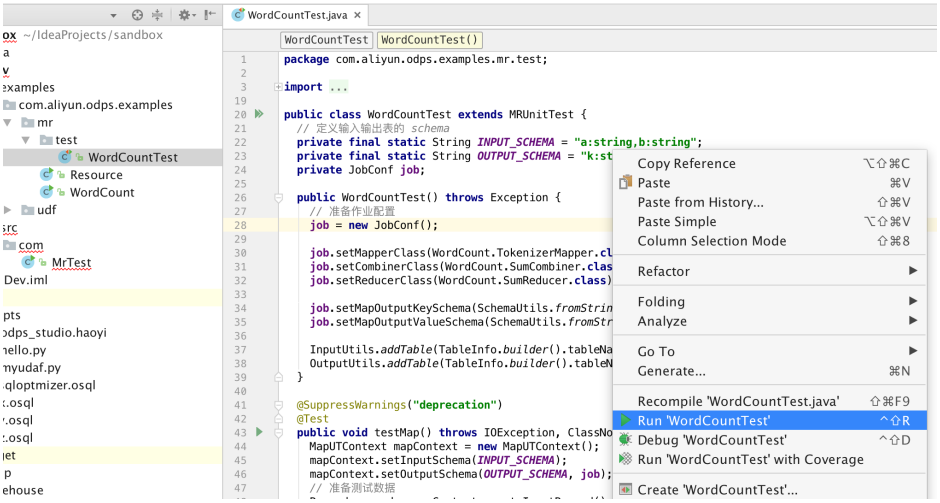
HelloDriver.java

```
HelloDriver
1  package mymr.myudf;
2
3  import ...
11
12 public class HelloDriver {
13
14     public static void main(String[] args) throws OdpsException {
15
16         JobConf job = new JobConf();
17
18         // TODO: specify map output types
19         job.setMapOutputKeySchema(SchemaUtils.fromString(_?));
20         job.setMapOutputValueSchema(SchemaUtils.fromString(_?));
21
22         // TODO: specify input and output tables
23         InputUtils.addTable(TableInfo.builder().tableName(_?).build(), job);
24         OutputUtils.addTable(TableInfo.builder().tableName(_?).build(), job);
25
26         // TODO: specify a mapper
27         job.setMapperClass(_?);
28         // TODO: specify a reducer
29         job.setReducerClass(_?);
30
31         RunningJob rj = JobClient.runJob(job);
32         rj.waitForCompletion();
33
34     }
35
36 }
```

调试MR

MR开发好后，下一步就是要测试自己的代码，看是否符合预期，我们支持两种方式:

单元测试：在examples目录下有WordCount的单测实例，可参考例子编写自己的UT。

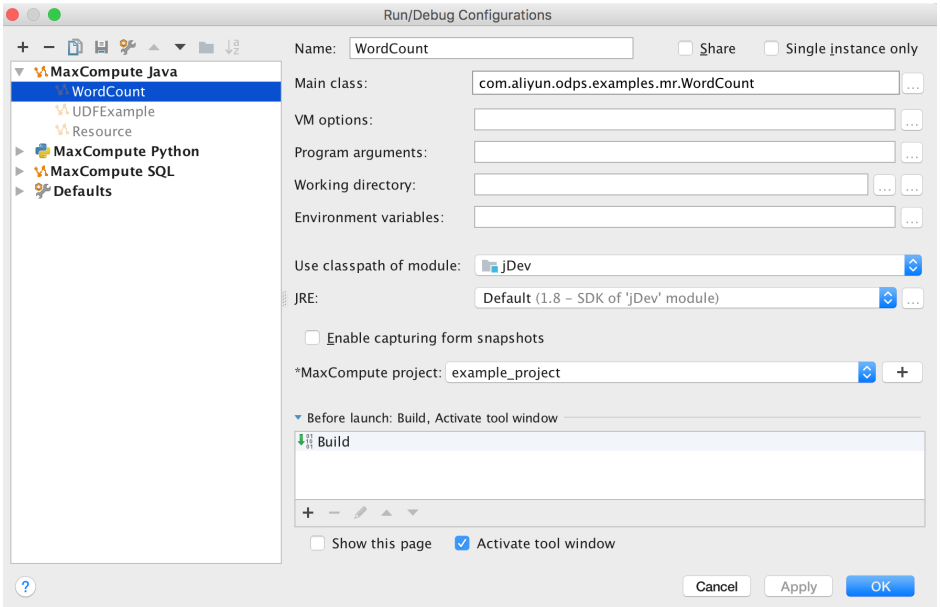


本地运行MR：本地运行时，需要指定运行数据源，有两种方式设定测试数据源：

studio通过tunnel服务自动下载指定MaxCompute project的表数据到warehouse目录下。默认下载100条，如需更多数据测试，请自行使用console的tunnel命令或者studio的表下载功能。

提供mock项目(example_project)及表数据，用户可参考warehouse下example_project自行设置。

1.运行MR: 在Driver类上右键，点击”运行”菜单，弹出run configuration对话框，配置MR需要在哪个MaxCompute Project上运行即可。



2.点击ok，如果指定MaxCompute project的表数据未被下载到warehourse中，则首先下载数据；如果采用mock项目或已被下载则跳过。接下来，MR local run框架会读取warehouse中指定表的数据作为MR的输入，开始本地运行MR，用户可以在控制台看到日志输出和结果打印。

生产运行MR

本地调试通过后，接下来就可以把MR发布到服务端，在MaxCompute分布式环境下运行了:

1.首先，你得将自己的MR程序打成jar包，并发布到服务端。如何打包发布?

2.通过studio无缝集成的MaxCompute console（具体的，在Project Explorer Window的project上右键，选择Open in Console），在console命令行中输入类似如下的 jar命令：

```
jar -libjars wordcount.jar -classpath D:\odps\clt\wordcount.jar com.aliyun.odps.examples.mr.WordCount wc_in
wc_out;
```

MaxCompute2.0新增了一套非结构化数据处理框架，支持通过外部表的方式直接访问OSS,OTS等。Studio对此提供了一些代码模板支持，方便用户快速开发。

编写StorageHandler/Extractor/Outputter

1. 创建MaxCompute Java Module(在examples目录下的unstructured文件夹有示例代码供参考)。
2. 在module的源码目录即src->main上右键new，选择MaxCompute Java。
3. 输入包名.类名，如myun.MyExtractor，选择类型Extractor，点击OK。

4. 模板已自动填充框架代码，只需要编写自己的逻辑代码即可。
5. 类似上述步骤可分别完成Outputter和StorageHandler的编写。

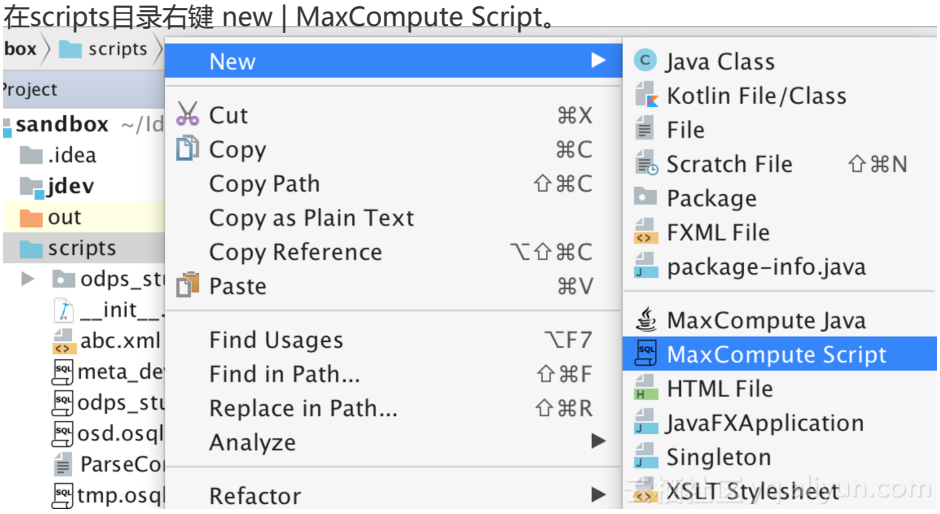
打包上传

StorageHandler/Extractor/Outputter写好后，可以参考打包发布将已写好的java程序打成jar包，并作为

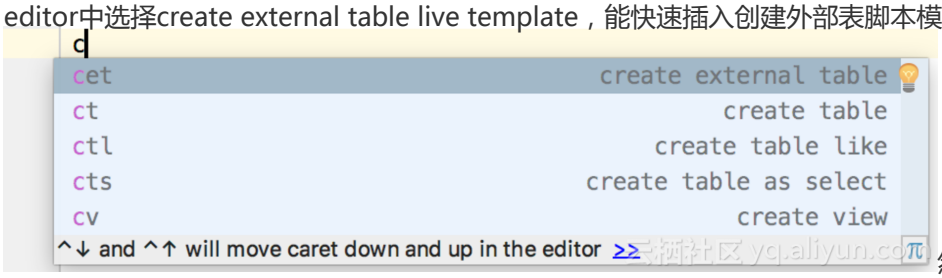
resource上传到服务端。

创建外部表

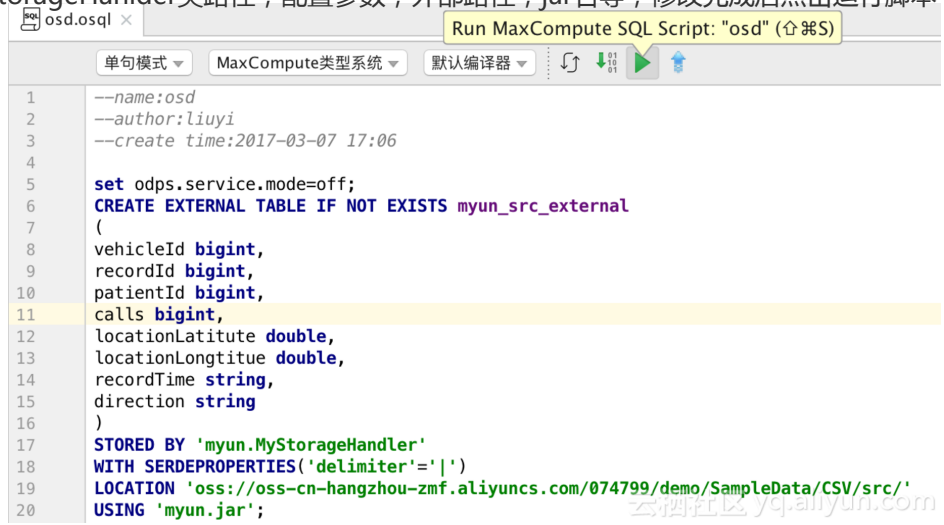
1. 在scripts目录右键 new | MaxCompute Script。



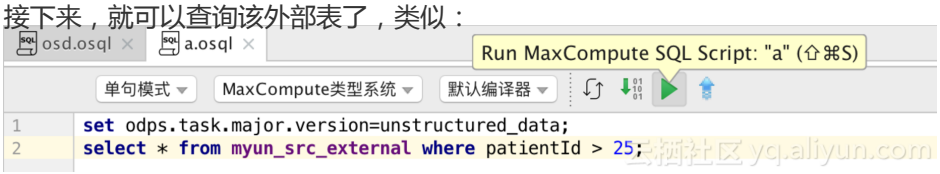
2. 输入sql脚本名，Target Project选择脚本将要在哪个MaxCompute project下执行，点击OK。
3. editor中选择create external table live template，能快速插入创建外部表脚本模板：



然后修改外部表名称，列及类型，StorageHanlder类路径，配置参数，外部路径，jar名等，修改完成后点击运行脚本，创建该外部表。



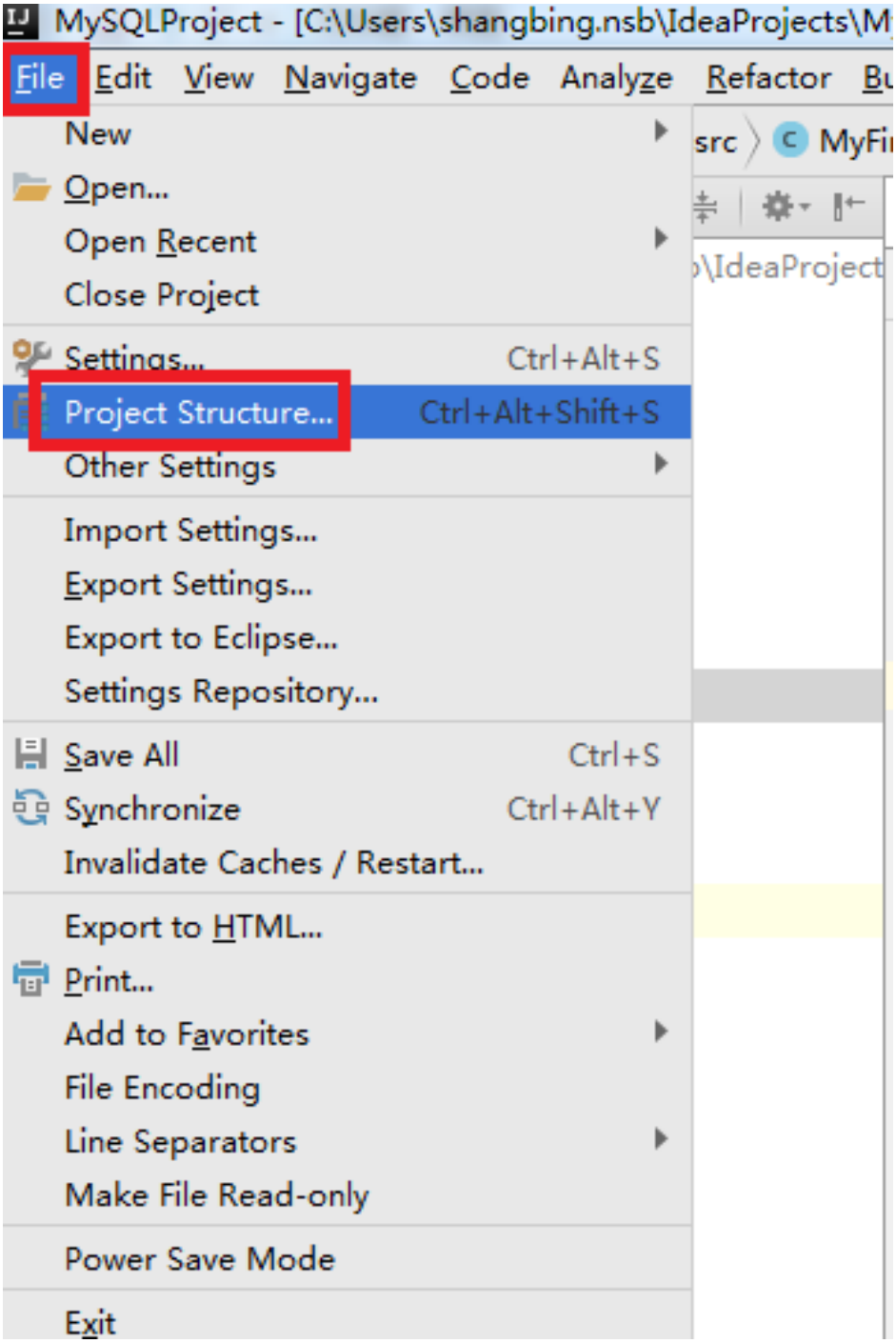
4. 接下来，就可以查询该外部表了，类似：



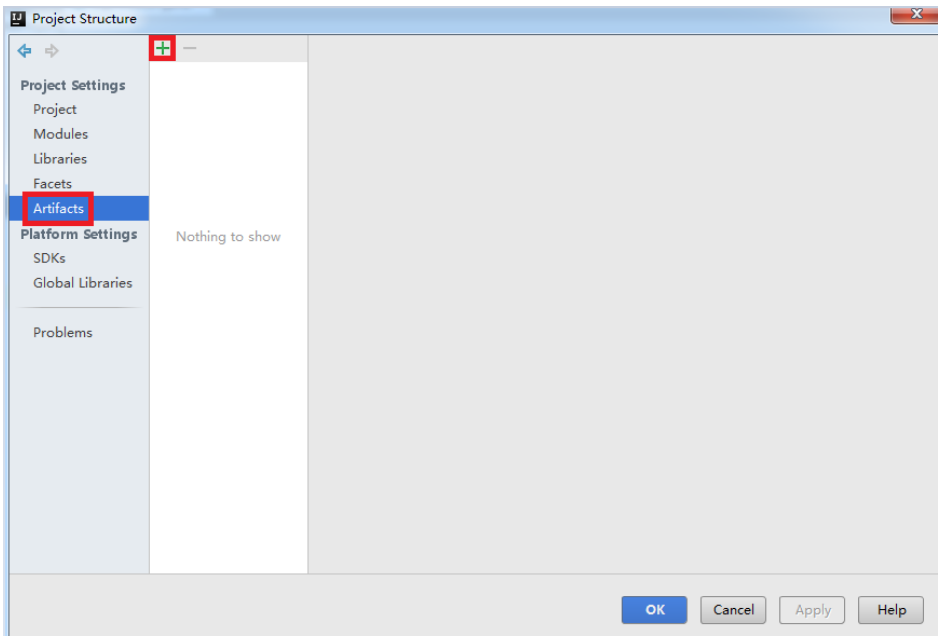
完成 UDF 或 MR 开发后，需要打包发布到 MaxCompute 系统。

打包 UDF 或 MR

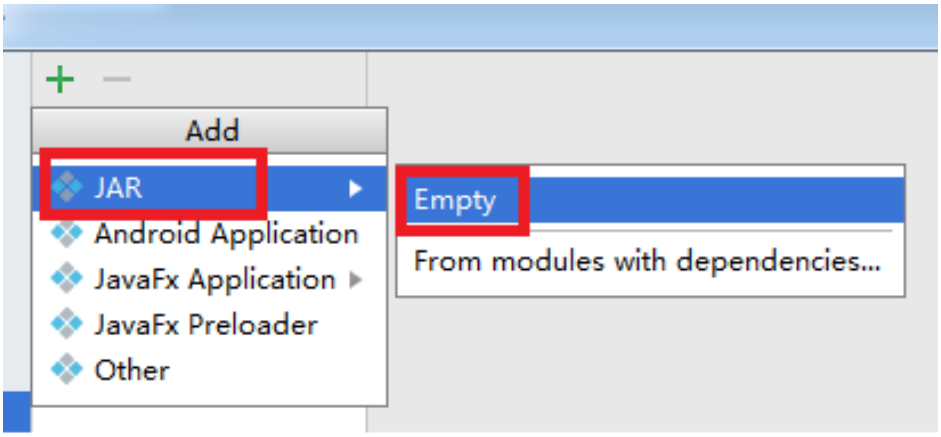
单击 File 下的 Project Structure。如下图所示：



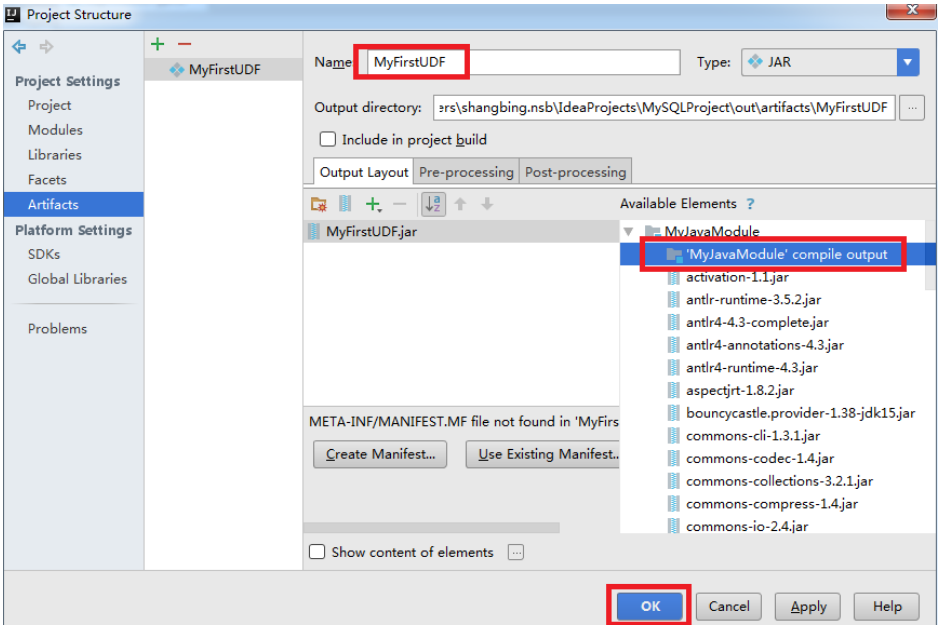
选中左侧导航栏中的 Artifacts，单击中间栏上方的加号。如下图所示：



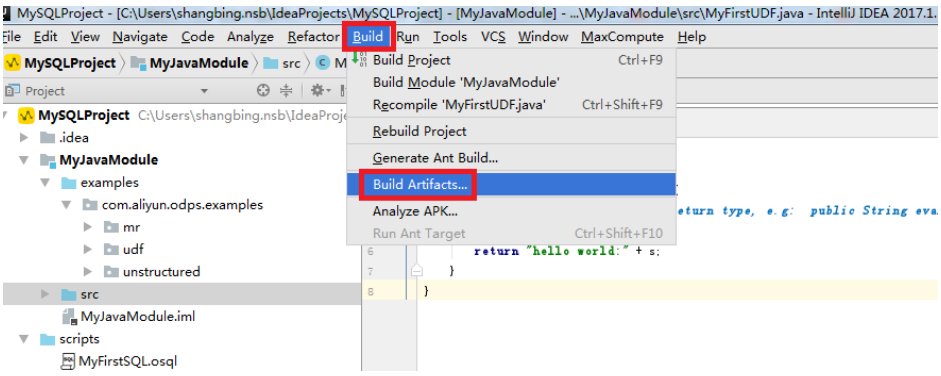
单击 JAR 下的 Empty，即添加一个空的 JAR 包。



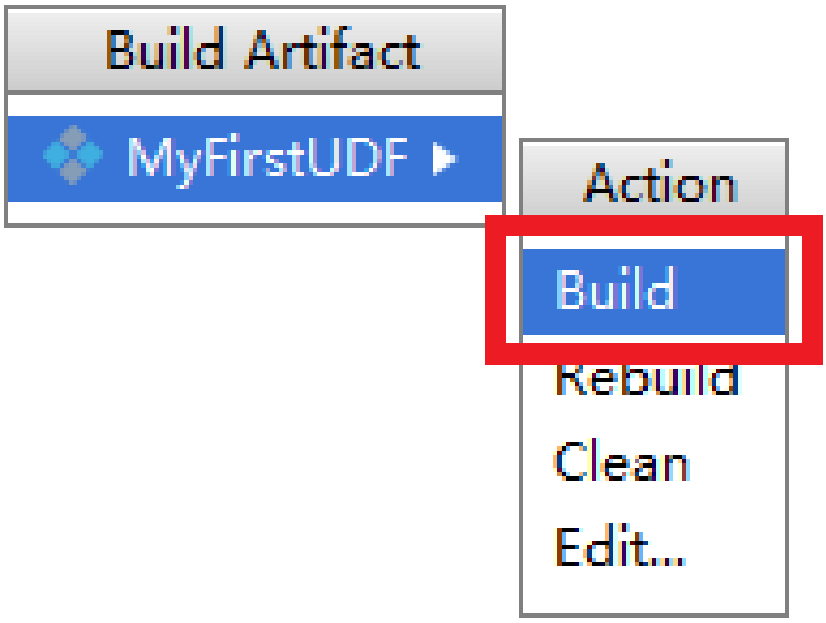
指定 Name 后，添加内容到该 JAR 包中，即双击右侧 Available Elements 中相应 Name 下的 MyJavaModule complie output，单击 OK，完成 JAR 包内容的设置。如下图所示：



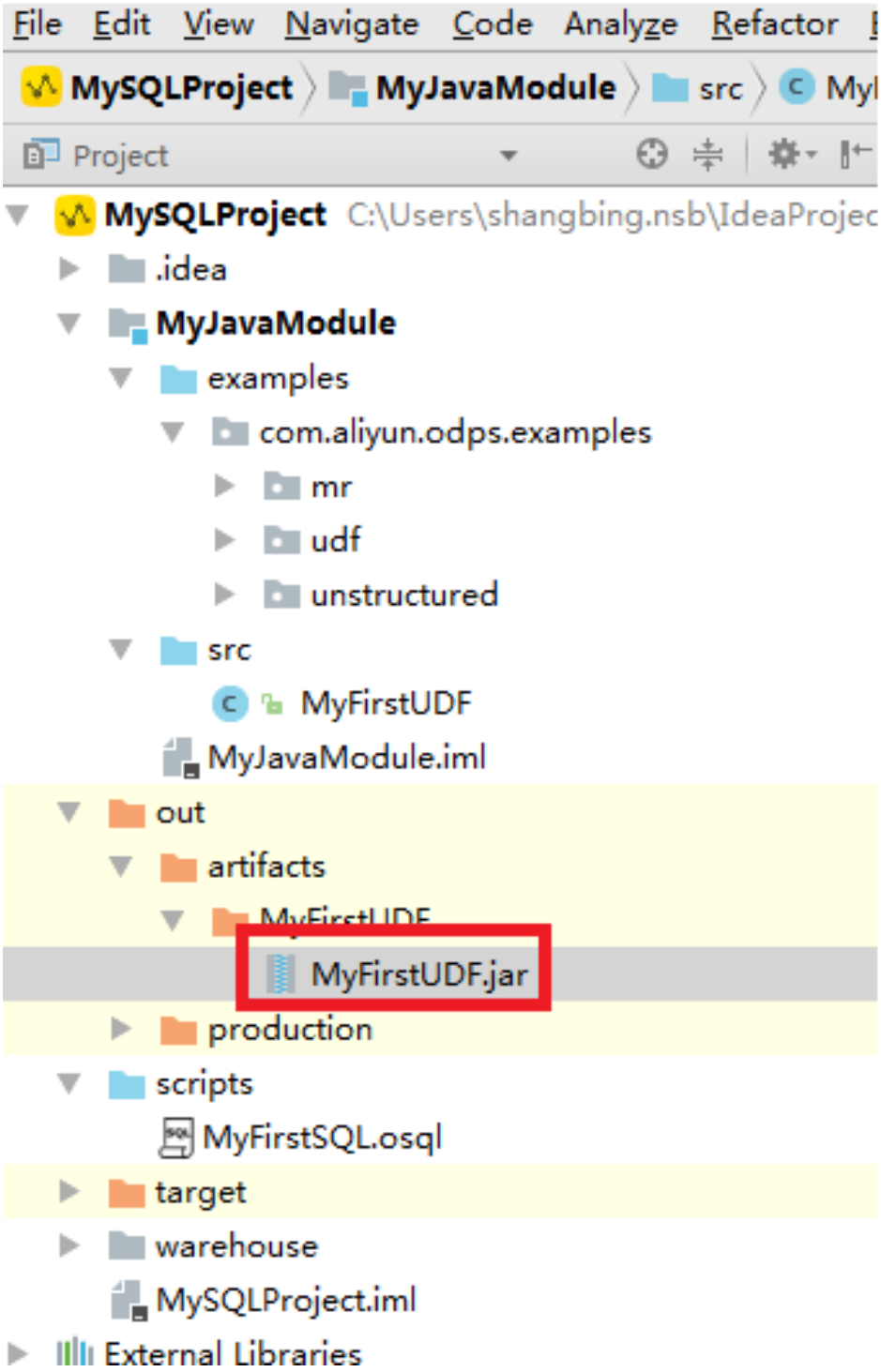
选择菜单中的 Build，单击 Build Artifacts。



单击弹出框中的 **Build**。



Build 完成后，即可在 MySQLProject 下的 out 目录中找到编译完成的 JAR 包。

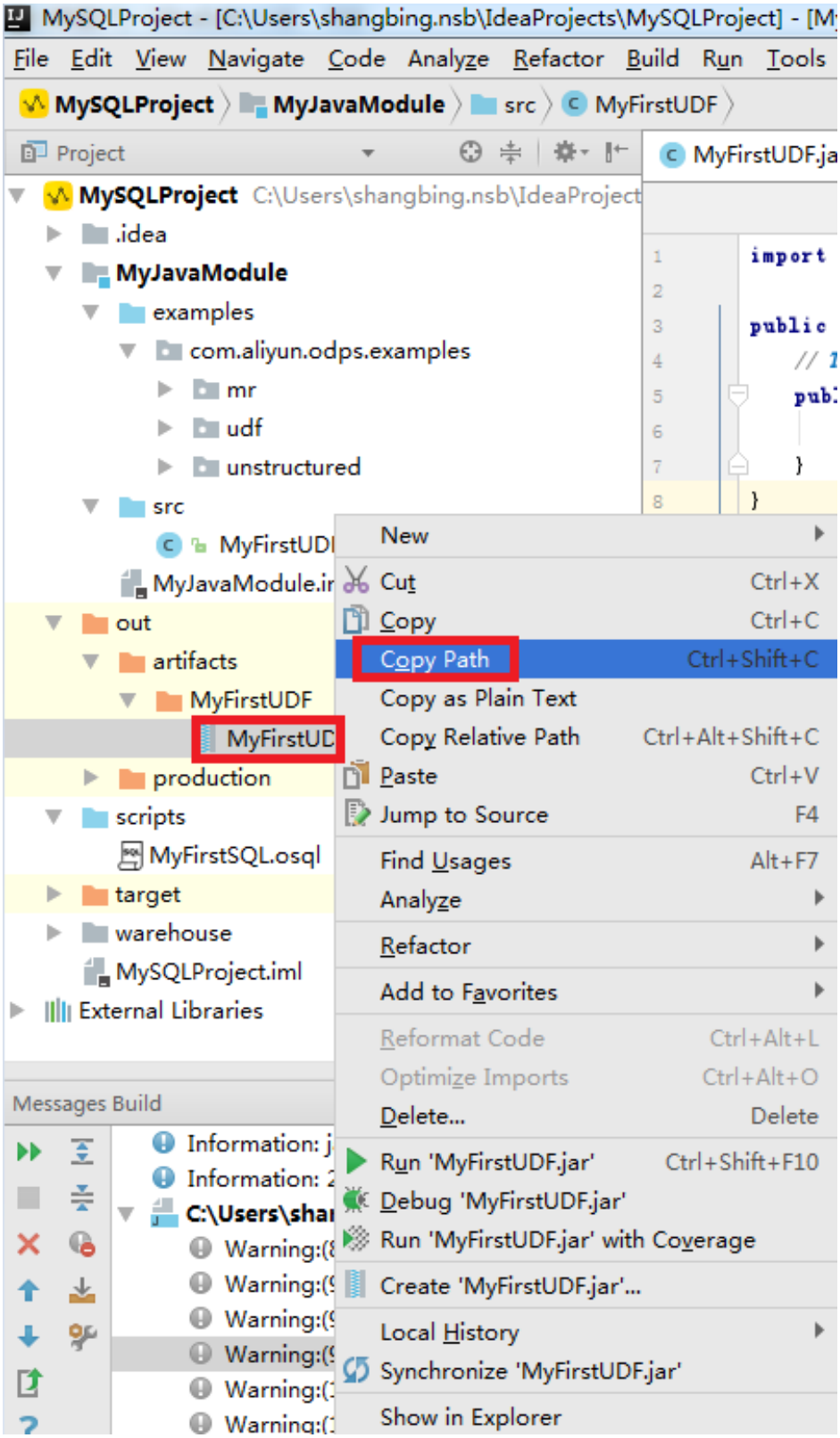


上传 JAR 包

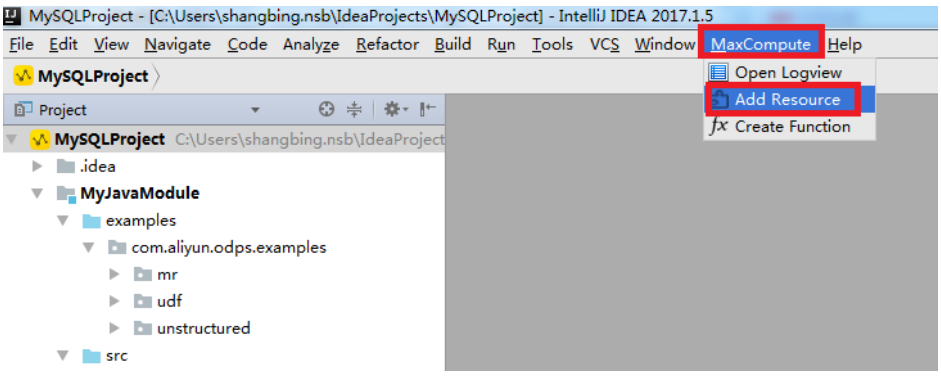
打包成功后，即可将该 JAR 包上传到 MaxComptute 服务端。

操作步骤

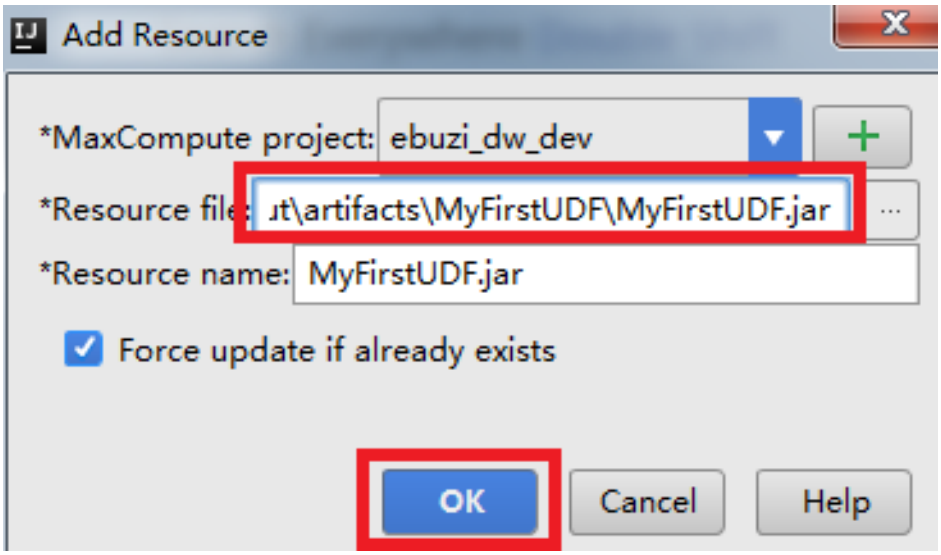
右键单击打包好的 JAR 包，选择菜单中的 Copy Path，复制 JAR 包在本地磁盘中的存储路径。



单击菜单中的 **MaxCompute -> Add Resource**，将该 JAR 包以资源的方式上传到 MaxCompute Project 中。



将复制的 JAR 包路径粘贴到 Resource file 中，单击 OK。



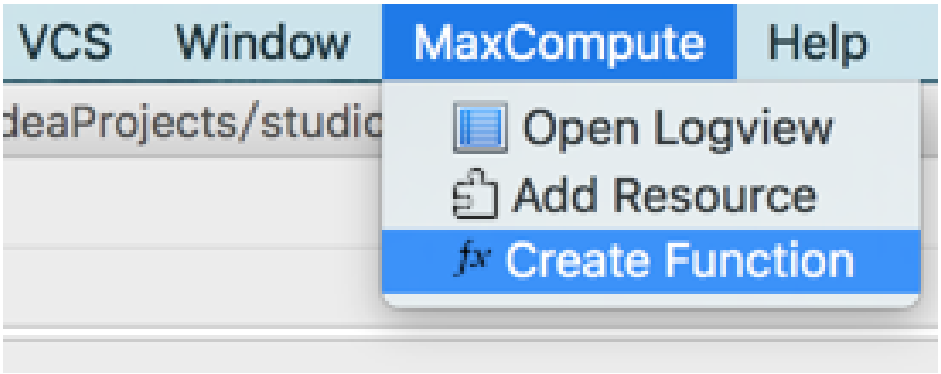
上传成功后，您即可在 project explorer 窗口的 resources 节点下看到该资源。

注册 UDF

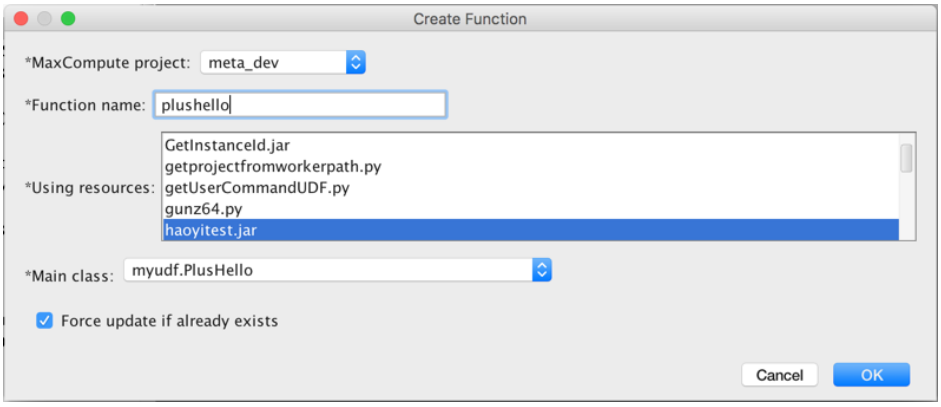
JAR 包上传完成后，即可注册 UDF 函数。

操作步骤

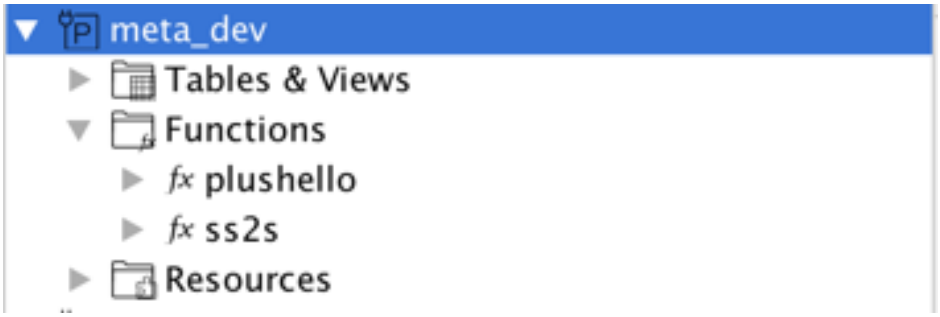
在 MaxCompute 菜单选择 **Create Function**。



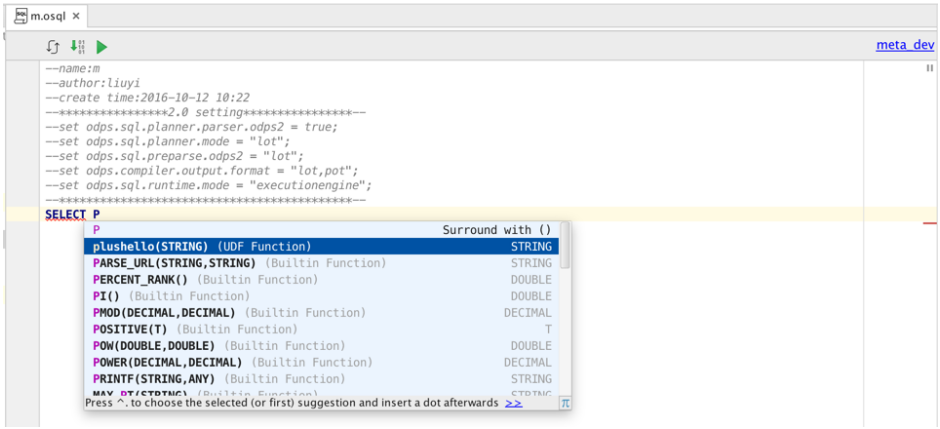
选择需要使用的资源 JAR 和 JAR 的主类，输入函数名，然后单击 OK。



注册成功后，即可在 project explorer 窗口的 functions 节点下看到该函数。



完成上述操作，即可在 SQL 中使用新编写的 UDF 完成后续开发。

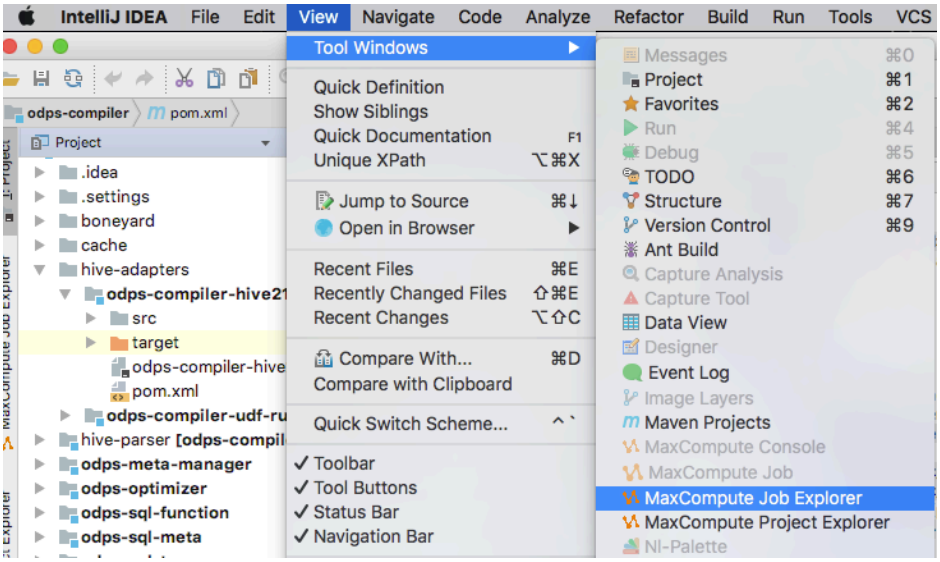


管理 MaxCompute 作业

通过Studio可以方便查看**当前用户**提交的MaxCompute的运行实例，包括运行状态、作业类型、起止时间等。

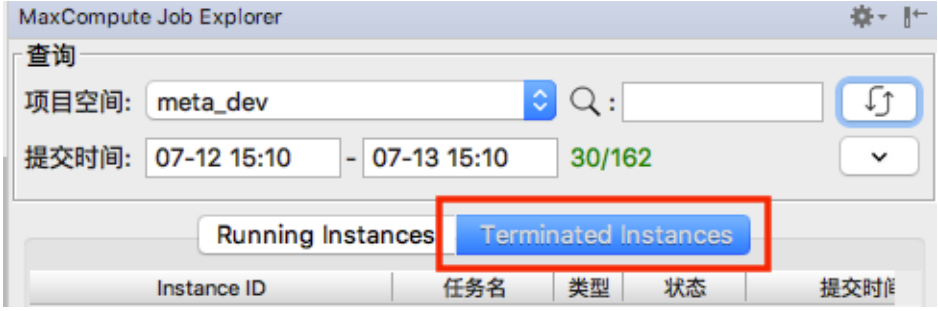
打开Job Explorer

如果Job Explorer View没有在左侧Dock上显示，可以通过View|ToolWindows|MaxCompute Job Explorer 打开：

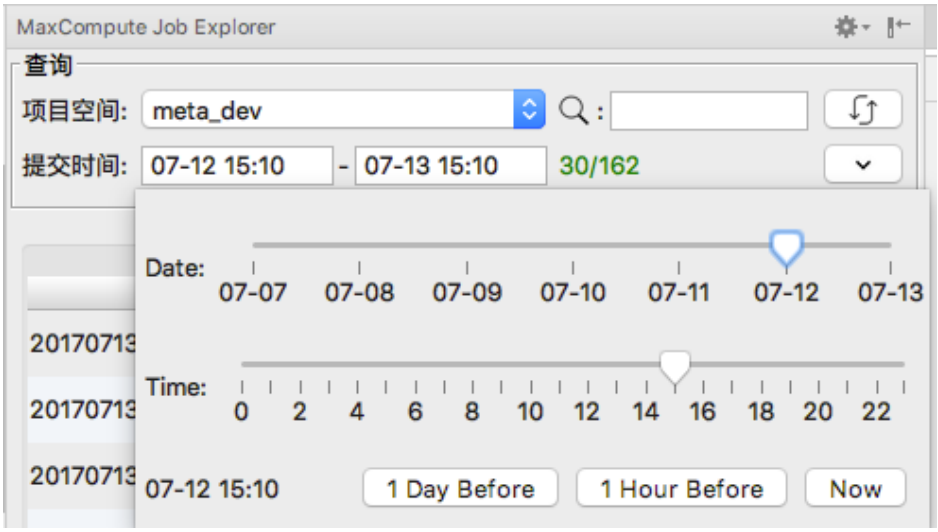


查看项目下所有作业实例

Job Explorer 支持按照状态查询提交的作业列表，如下图所示，切换到Terminated Instances tab表示查询已经结束的作业：

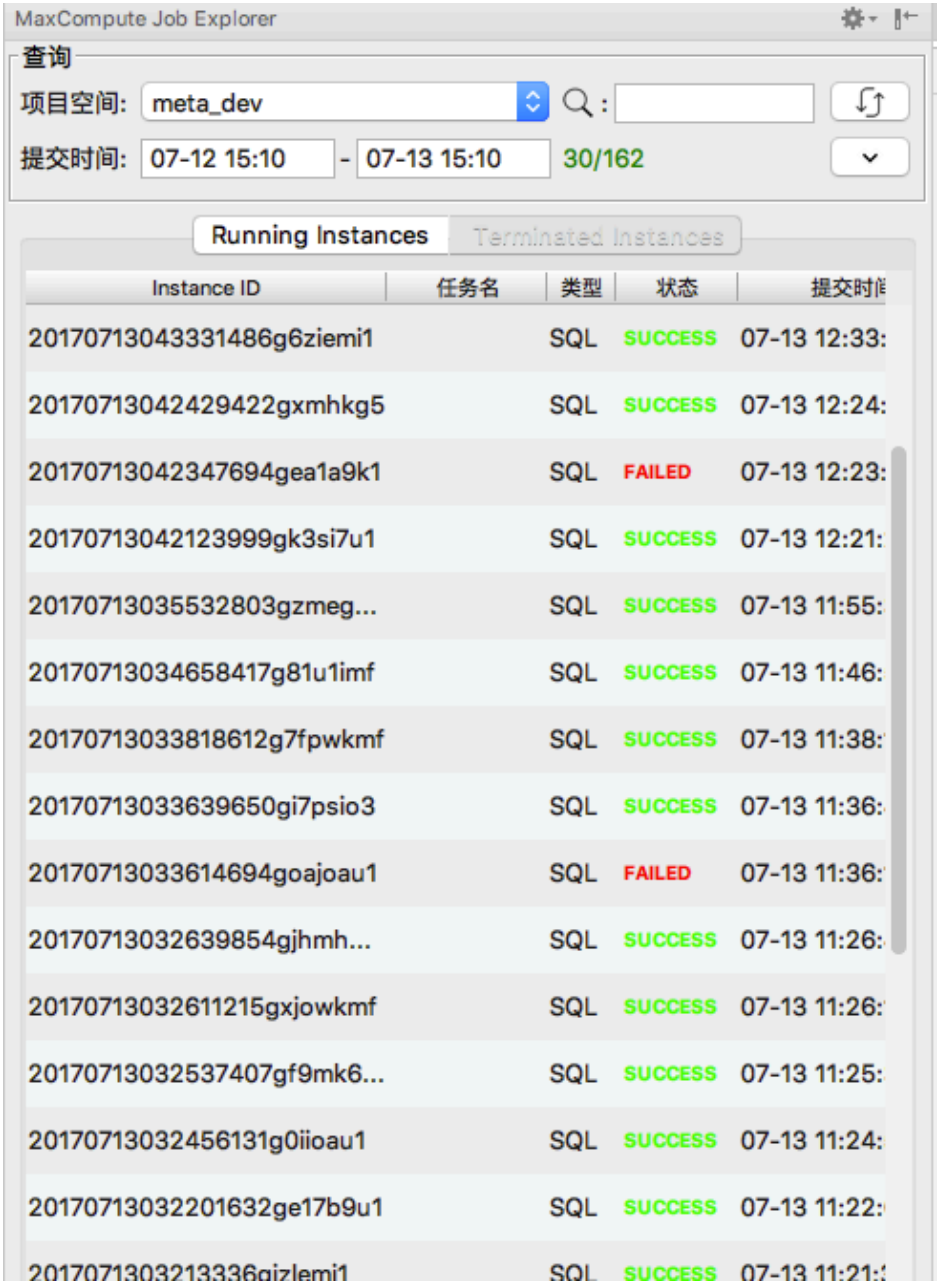


选择MaxCompute项目，通过Time Picker 设置查询的起止时间：



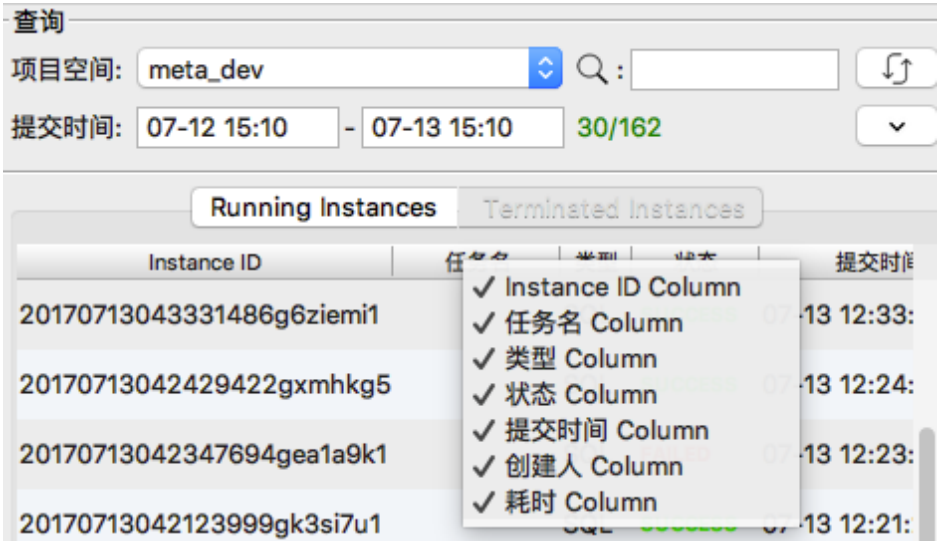
点击刷新按钮获取作业列表。其中**30/162**表示**当前显示实例个数与总实例个数。**

当右侧 Scroll Bar 滑动到底部时会自动追加更多实例，默认追加20条。



作业列表Column配置

可以通过Table header的popup menu 关闭/打开 作业列表中的任意列：



跳转到作业排队队列

Running 状态作业如果正在排队队列中等待调度，作业列表中会展示当前排队的位置和全局优先级，可以通过点击跳转到排队队列详情页面：

Tips: 在Running Instances Tab下的作业状态、队列位置等会自动更新，作业结束后会从列表中移除

搜索功能

可以根据作业实例 ID 作为查询条件，精确搜索指定实例。

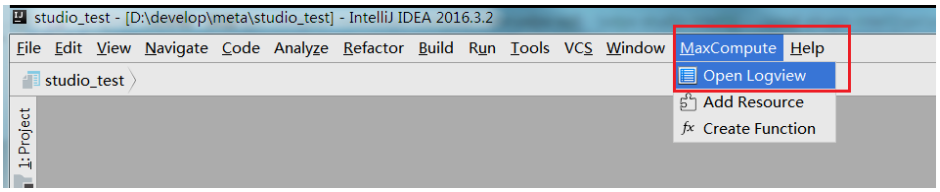
Tips:在作业实例列表中右键可直接Copy **InstanceId**及**Logview URL**

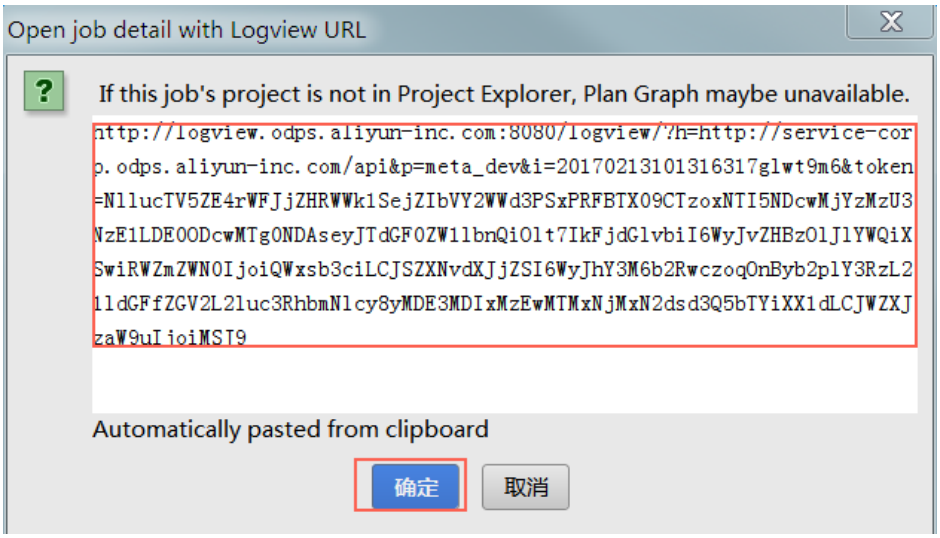
查看作业实例

Studio支持两种方式查看MaxCompute作业实例：

1.通过 Logview URL 以只读方式打开作业详情。使用 Logview 来查看一个作业的详细信息是 MaxCompute 用户熟悉的方式。使用 Logview 还有一个便利之处是可以查看其他用户在其他项目空间中提交的任务状态。在 Studio 中我们也提供了通过输入一个有效的 Logview URL 打开任意一个作业详情的功能。

在菜单栏中，打开MaxCompute 菜单下Open Logview，即可自动将粘贴板中有效Logview URL地址复制到弹出窗口中。





2.作业浏览信息中，选择某个MaxCompute实例，双击或右键Open，可查看该实例详细信息。

作业详情视图

作业详情页面包含主要包含五个视图和两个Tool Window窗口：

可视化视图: 该视图以图形化方式显示作业整体信息，可查看子任务依赖关系，子任务实例详细信息等。

概要(JSON)视图：以JSON格式显示作业运行详细信息。

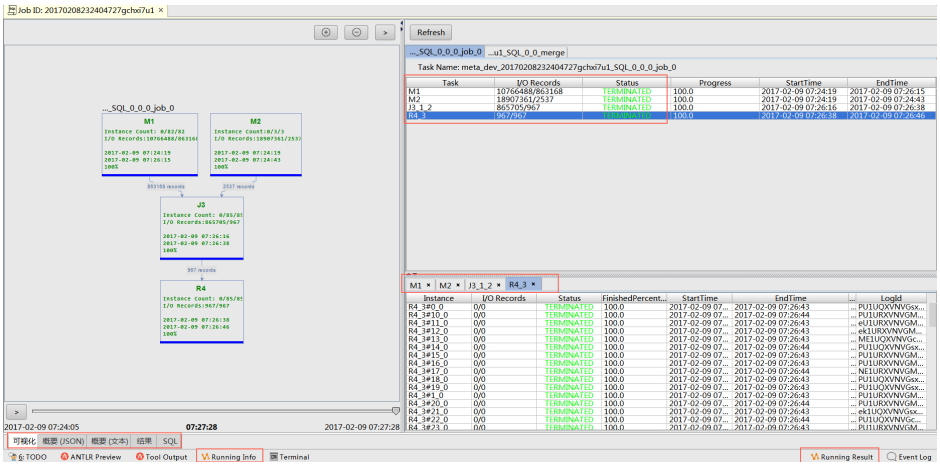
概要(文本)视图：以文本方式显示子任务运行信息。

结果视图：显示该作业运行结果。

SQL视图：显示该作业提交时所对应的SQL语句。

Running Info：Tool Window，显示运行的logview地址，点击URL地址即可跳转到Logview中。

Running Result：Tool Window，显示运行结果，通结果视图一致。

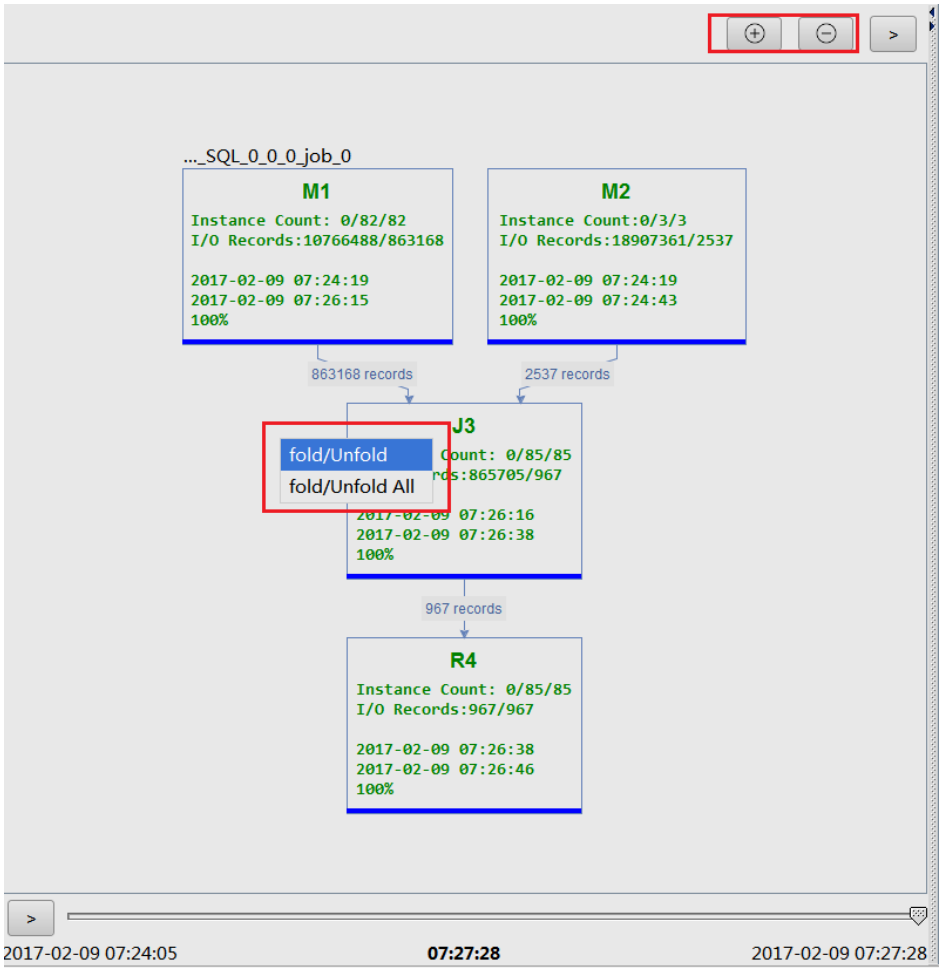


可视化视图详情

可视化视图作为日常主要使用工具，进行详细介绍，其他视图可点击相应名称进行切换。视图左边展示作业的执行关系图，右边展示子任务列表及各个子任务对应的Worker列表。可点击**Refresh**按钮刷新作业，重新获取作业详情。

关系图：子任务之间的执行逻辑图，双击或右键展开可显示各个子任务的**POT图**。用户可点击+ 或 -对视图进行放大缩小。

Tips:放大缩小也可按 **Ctrl + 鼠标滚轮**。



关系图信息说明：

Instance Count：a/b/c, 指某一时刻正在运行子任务实例个数为a，已结束任务实例个数b，总任务实例个数c。

I/O records：同理为某一时刻的input records和output records。（**注意这里没有总records，因为如果是running状态不能获取总records数**）。

百分比与蓝色进度条：表示该任务运行情况，该比例根据子任务运行实例分析得出。

子任务间连线上显示的是输出records数量。箭头表示数据流动方向。

子任务列表：以列表形式呈现当前作业所启动的子任务的信息，包括状态、I/O量，执行完成度，起止时间等。

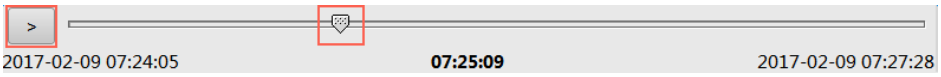
Worker列表：以列表形式呈现各个子任务对应进程信息。

作业回放

Studio支持作业回放功能，作业回放就像播放媒体文件一样，可在12s内回顾该Job执行的历史轨迹。该功能主要用于帮助用户了解MaxCompute实例在不同时刻运行状态，快速判断子任务级运行顺序及消耗时间，掌握Job执行关键路径，从而针对运行较慢的子任务进行优化。

点击 > 按钮即可开始播放，再次点击则暂停。用户也可手动拖动进度条。

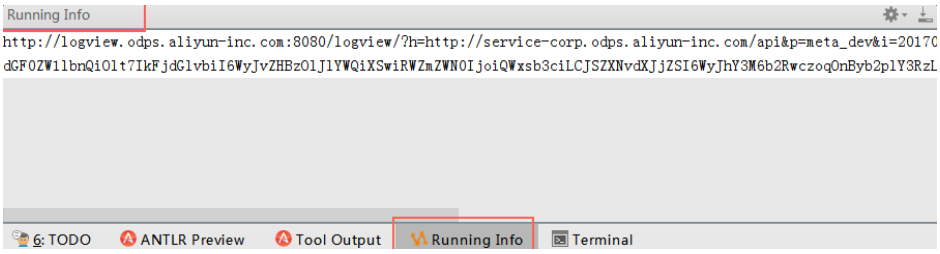
进度条左边为作业开始时间，中间为播放时间，右边为结束时间。如果是running状态的job，则右边为当前时间。



注意：回放功能仅通过时间估算某一个时刻IO数据量，从而确定完成进度，并不能代表该时刻真实IO数据量。

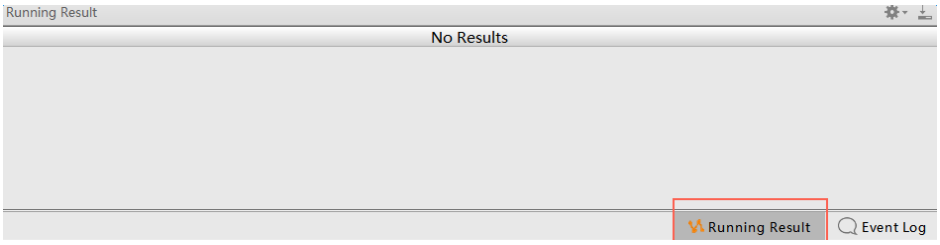
Running Info

展示作业运行输出信息、点击Logview URL可以再浏览器中打开对应Logview



Running Result

展示作业运行的数据结果（并不是所有的作业都有数据输出）

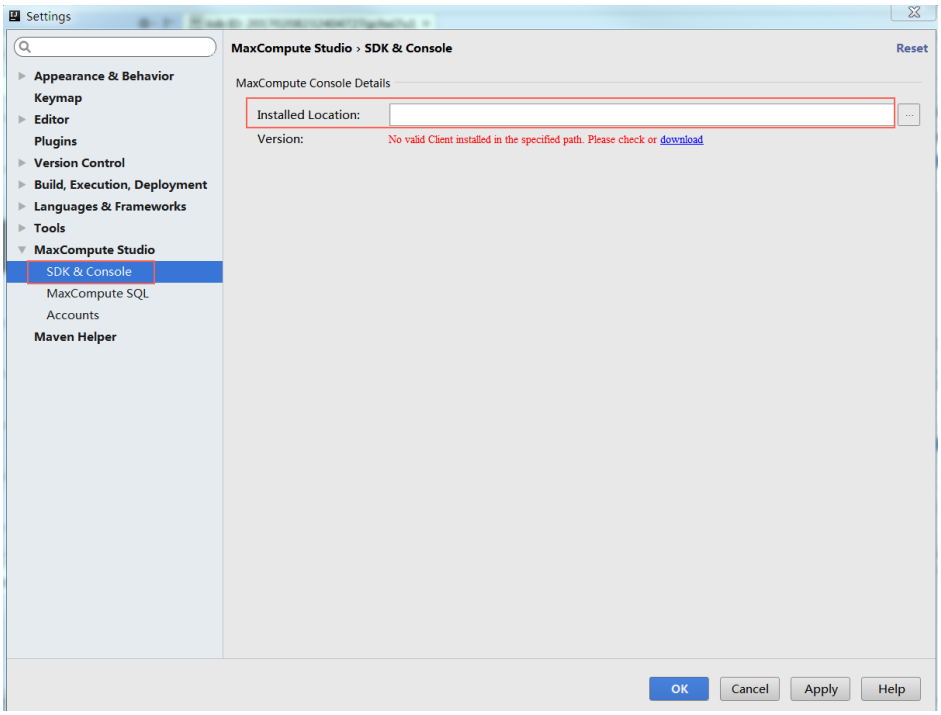


工具集成

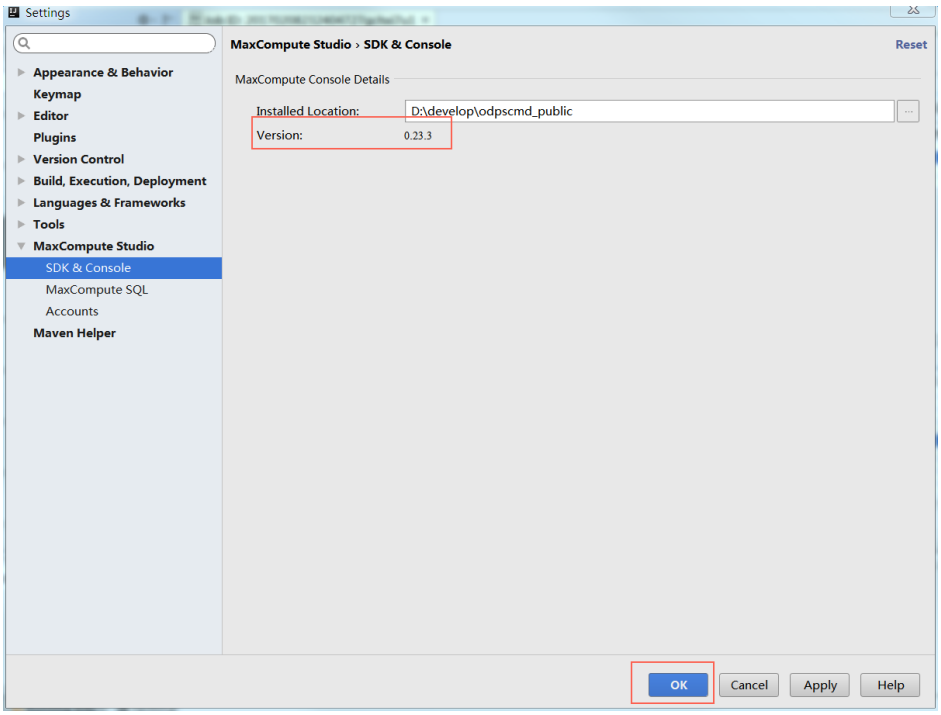
MaxCompute Studio集成了MaxCompute客户端程序，可在Studio中直接打开客户端。

配置客户端安装路径

Studio中已包含最新版MaxCompute客户端程序，并指定为默认客户端。用户也可自行指定其他版本客户端程序，设置位置为：Intellij Settings | MaxCompute Studio | SDK & Console 中添加客户端程序跟路径。Console下载地址



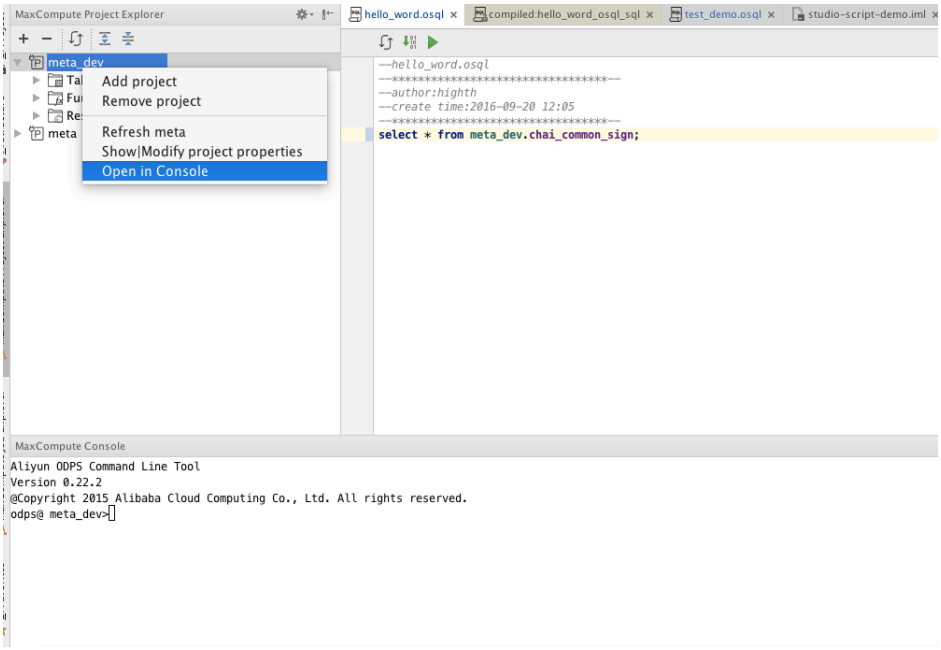
设置成功后会显示MaxCompute客户端版本信息。



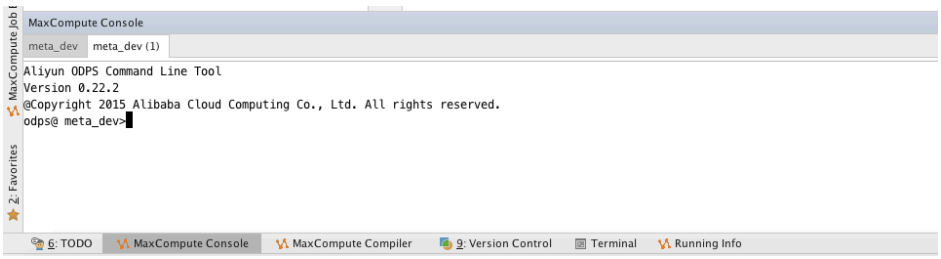
打开MaxCompute客户端

设置MaxCompute客户端安装路径成功后，就可以在Studio中打开客户端程序。

在项目浏览列表中，选中要打开的项目，右键菜单中点击Open in Console。



用户可以打开多个客户端程序，操作步骤同上。



配置选项

安装 MaxCompute Studio 插件后，可以在 IntelliJ IDEA 的 Setting 页面的左边栏找到 MaxCompute Studio 的配置项。打开 IntelliJ 配置页面的方法请参考 IntelliJ 的文档

MaxCompute Studio 配置选项页

MaxCompute Studio 配置选项页提供以下配置项：

本地元数据仓库存储路径
指定在本地存储 MaxCompute 项目空间元数据的路径。Studio 的缺省设置是本地用户目录下的 .odps.studio/meta 这个隐含目录。

版本更新选项

- 复选框 **Automatically checks for new version** 可以控制 Studio 是否自动检查新版本更新
- 按钮 **Check new versions** 用于手动检查新版本。 点击检查新版本按钮后，如果有新版本可以更新，将显示 **Install new version** 按钮，点击可以安装，安装完成后需要重启 IntelliJ IDEA。

SDK & Console 配置选项页

SDK & Console 配置选项页面提供以下配置项：

MaxCompute 客户端安装路径
指定本地安装的 MaxCompute 客户端的安装路径。 Studio 会检测路径中安装的 MaxCompute 客户端的版本，如果检测失败，会提示错误信息。

> Studio 2.6.1 之后的版本自带了最新的 MaxCompute 客户端，用户不用特别指定。 如果用户希望使用自己特定版本的 MaxCompute客户端，可以指定路径。

MaxCompute SQL 配置选项页

MaxCompute SQL 配置选项页面提供以下配置项：

启动语法高亮
勾选 **Enable syntax coloring** 选项，启动语法高亮功能。

启动代码自动补全
勾选 **Enable code completion** 选项，启动代码自动补全功能。

启动代码格式化
勾选 **Enable code formatting** 选项，启动代码格式化功能。

编译器选项
在这里这是全局缺省的编译器选项。 以下选项还可以在 SQL 编辑器的工具栏上为每个文件单独设置。

- **编译器模式（Compiler Mode）**
  - 单句模式（Statement Mode）：在该模式下，编译器对 SQL 文件单条语句作为单元进行编译、提交
  - 脚本模式（Script Mode）：在该模式下，编译器对整个 SQL 文件作为单元进行编译、提交。 *（注：脚本模式有利于编译器和优化器更大程度地优化执行计划，提高整体执行效率，目前处于测试阶段）*
- **类型系统**
  - 旧有类型系统（Legacy TypeSystem）：原有 MaxCompute 的类型系统
  - MaxCompute 类型系统（MaxCompute TypeSystem）：MaxCompute 2.0 引入的新的类型系统
  - Hive 类型系统（Hive Compatible TypeSystem）：MaxCompute 2.0 引入的 Hive 兼容模式下的类型系统
- **编译器版本**
  - 默认编译器（Default Version）：默认版本的编译器，
  - 实验性编译器（Flighting Version）：实验性的版本的编译器，包含正在测试中的编译器的新特性

Account 配置选项页

Account 配置选项页面提供添加和管理用户用于访问 MaxCompute 所用账户，参考 用户认证：

Studio 需要通过用户指定的账号访问 MaxCompute 的项目空间和执行提交作业等操作，目前 Studio 支持的账号类型有：

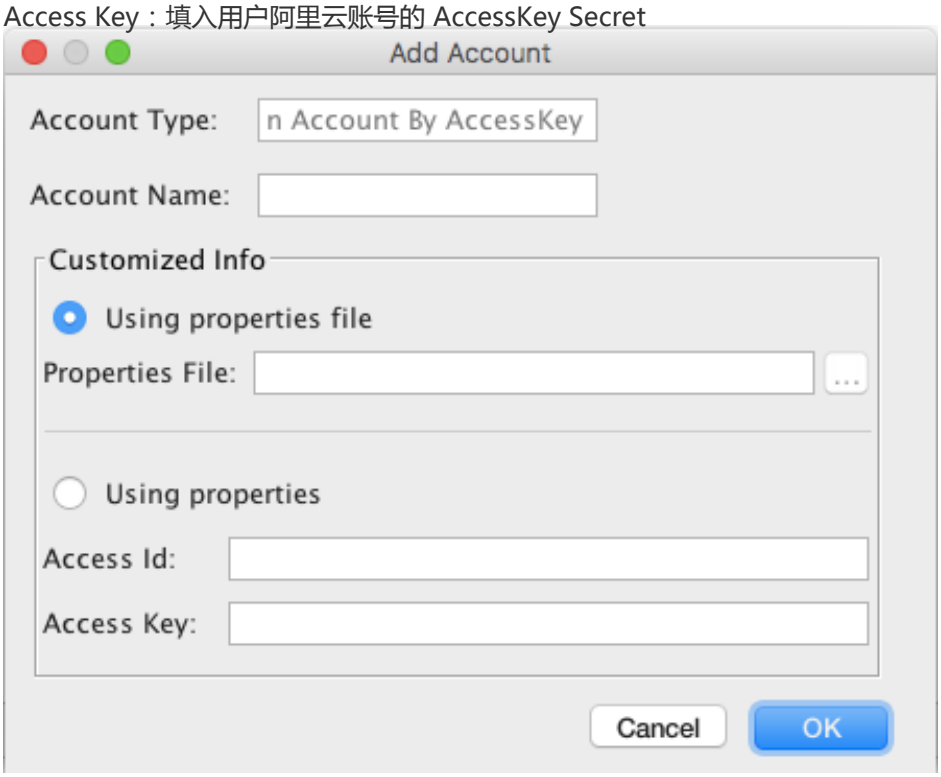
- 阿里云账号（AccessKey）

添加账户

在 Account 配置选项页面，执行以下步骤：

1. 点击按钮 + 或者快捷键 Ctrl-N
2. 选择账户类型 **Aliyun Account by AccessKey**
3. 在弹出的 **Add Account** 窗口中填入:

- Account Name： 改账户在 Studio 中的标识名称
- Using properties file: 从配置文件中读取 AccessKey ID 和 AccessKey Secret
    - 选择一个在 **用户认证** 中 conf/odps_config.ini 示例的配置文件
- Using properties： 手工填入 AccessKey ID 和 AccessKey Secret
    - Access Id： 填入用户阿里云账号的 AccessKey ID
    - Access Key：填入用户阿里云账号的 AccessKey Secret



1. 点击按钮 **OK** 完成添加。 添加完成后账号会出现在 Account 配置选项页面的 Account 列表里。

删除账户

在 Account 配置选项页面，执行以下步骤（该操作仅在 Studio 配置中删除账户配置，对用户账户本身不产生影响）：

1. 在 Account 列表中选择要删除的账户名称
2. 点击按钮 -
3. 在弹出的的确认对话框中，选择 **OK**

修改账户 AccessKey

在 Account 配置选项页面，执行以下步骤：

1. 在 Account 列表中选择要删除的账户名称
2. 点击”铅笔”图标进行编辑
3. 在弹出的 **Edit Account** 窗口中编辑 Account 信息，内容与上面的 **Add Account** 窗口类似

常见问题（FAQ）

如何通过studio来开发MaxCompute Java UDF?

- **1.新增一个module:** 参考创建MaxCompute java module，我们的UDF代码将放置在此module中。
- **2.开发UDF:** 参考UDF开发 studio提供了UDF开发模板，你可以依据模板完成UDF开发。
- **3.测试:** studio提供了UDF本地调试的机制，你可以参考UDF test模板编写自己的测试用例。
- **4.打包:** 你可以依赖IDE本身提供的打包功能，将自己的UDF源码打包成jar。
- **5.注册发布:** 在打好jar包后，需要先添加资源，随后注册函数。注册成功后就可以在MaxCompute project explorer窗口的functions节点下查看该UDF，同时也可以在script editor里使用该UDF。

MaxCompute用户在日常使用过程中，需要浏览管理所在project的meta（包括表，函数和资源），我们来看看如何通过Studio来方便的管理操作。

- **1.如何新增一个project连接:** 参考新增连接 通过MaxCompute project explorer新建一个MaxCompute project连接，连接连通性测试成功后，就可以看到刚才新增的project节点树。
- **2.如何查看table列表及schema:** 参考浏览meta 展开project下的tables & views能看到table和view列表，展开具体的某个table能看到列和类型。
- **3.如何查看function代码:** 展开functions能看到function列表，展开具体的某个function能看到方法签名，双击该function即可以看到反编译的源码。
- **4.如何查看下载表的sample数据:** 参考导入导出表数据 双击某个table就能看到详细的schema，还能预览sample数据，在data preview窗格中右键“Export grid data”。你也可以直接在节点树的表名上右键export。
- **5.如何更新节点的meta:** 譬如某个表新增了一列，你可以在选中该表后右键”refresh meta”或点击工具栏上的refresh按钮，即可以看到新增的这一列了。又譬如project下新建了一些表，选中tables & view后点击工具栏上的refresh按钮即可以看到新建的这些表了。
- **6.如何对表做DDL操作:** 删除是可以直接批量操作的，选中表后，右键”Delete table from server”即可。新增和编辑还不支持直接通过界面操作，你可以通过在editor里编写相应的DDL语句来完成。

Eclipse开发插件

为了方便用户使用 MapReduce及UDF的Java SDK进行开发工作，ODPS提供了Eclipse开发插件。该插件能够模拟MapReduce及UDF的运行过程，为用户提供本地调试手段，并提供了简单的模板生成功能。

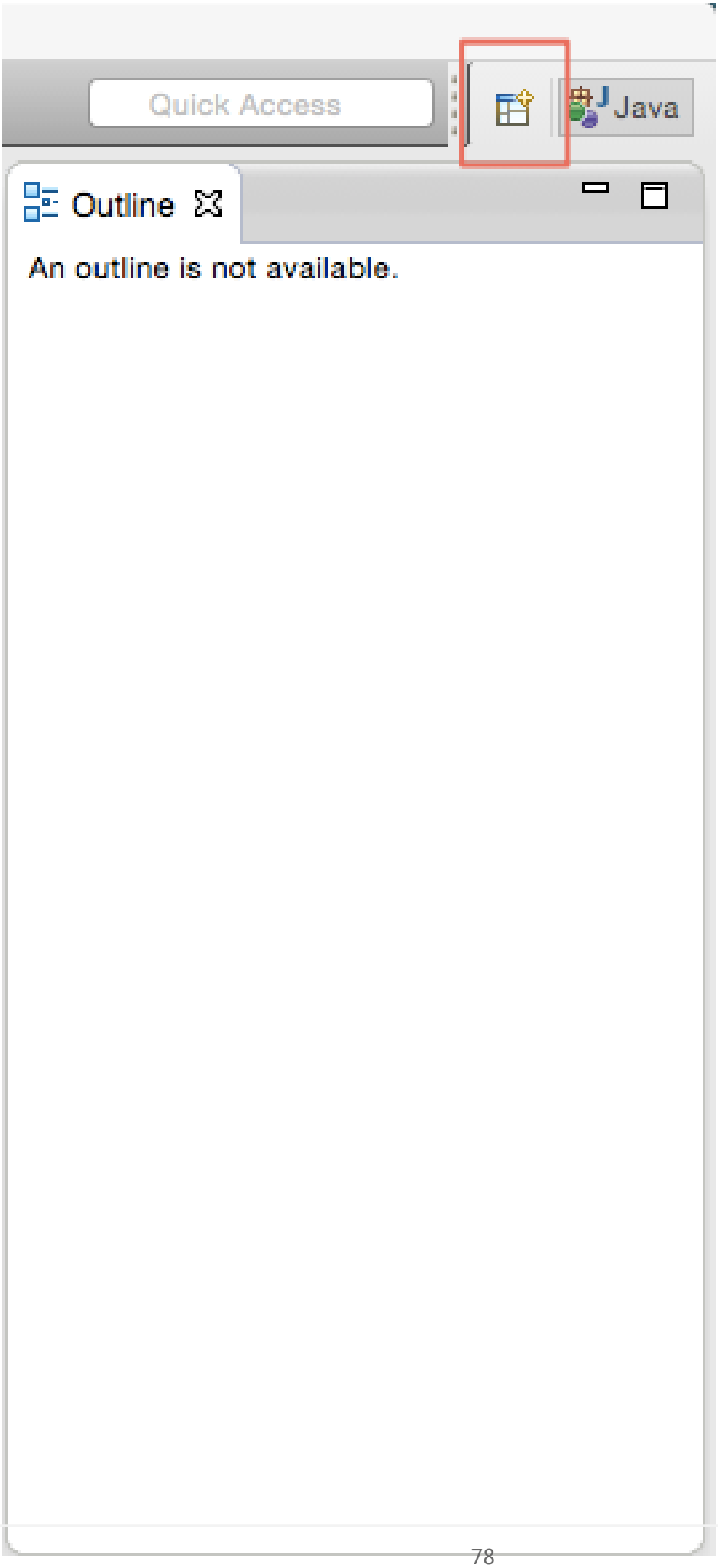
备注：

- 目前高版本的Eclipse Neon有可能会导致插件加载失败，请使用Eclipse Luna版本。
- 下载此插件请点击这里。
- 与MapReduce提供的本地运行模式不同，Eclipse插件不能够与ODPS同步数据。用户使用的数据需要手动拷贝到Eclipse插件的warehouse目录下。

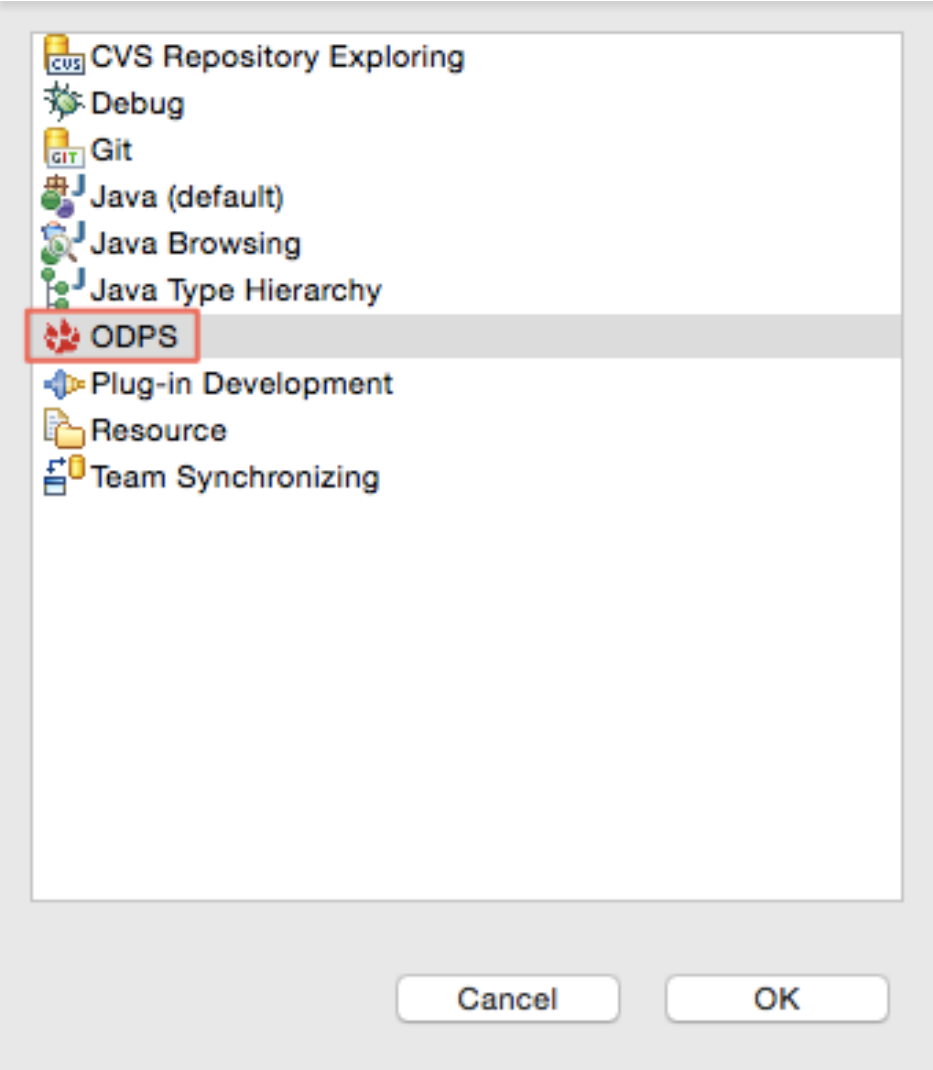
下载Eclipse插件后，将软件包解压，会看到如下jar内容：

```
odps-eclipse-plugin-bundle-0.16.0.jar
```

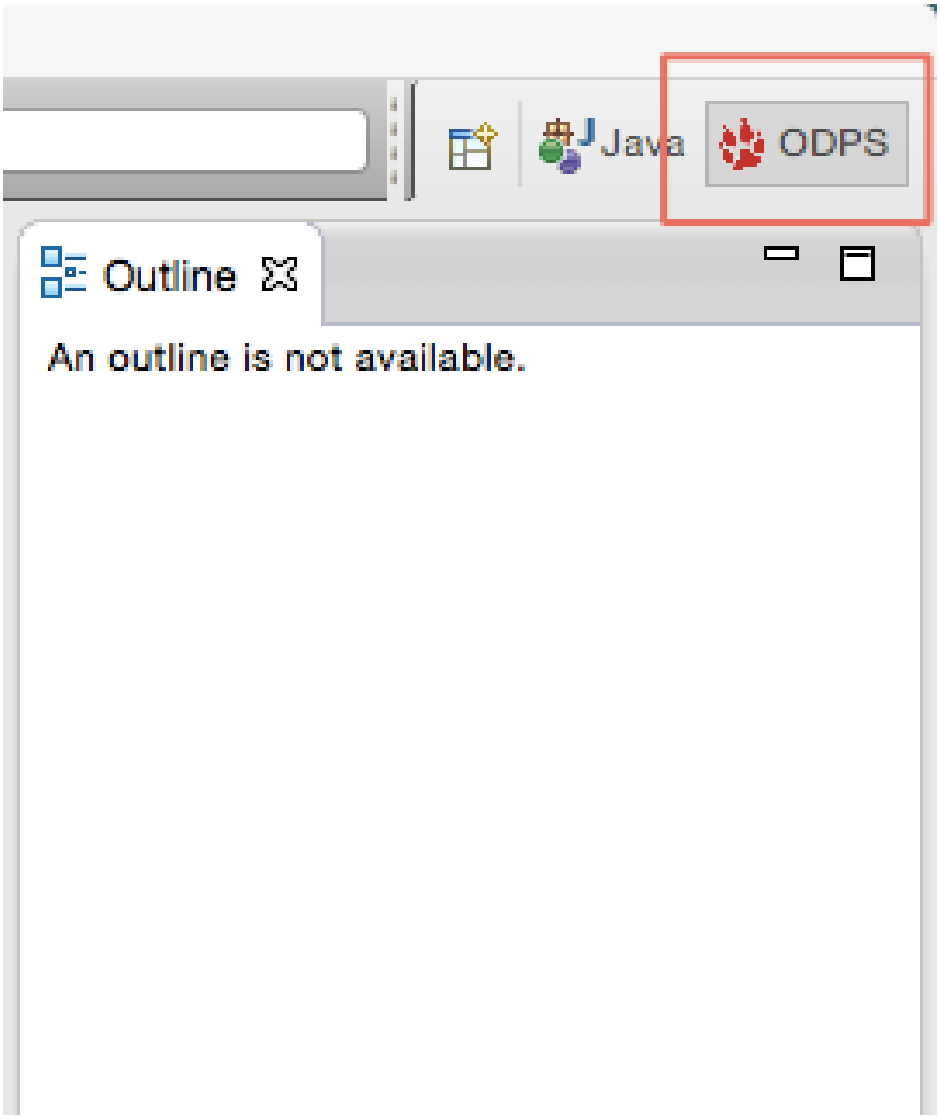
将插件放置在Eclipse安装目录的plugins子目录下。打开Eclipse，点击右上角的打开透视图(Open Perspective)：


Quick Access
Java
Outline
An outline is not available.

点击后出现下面的对话框：



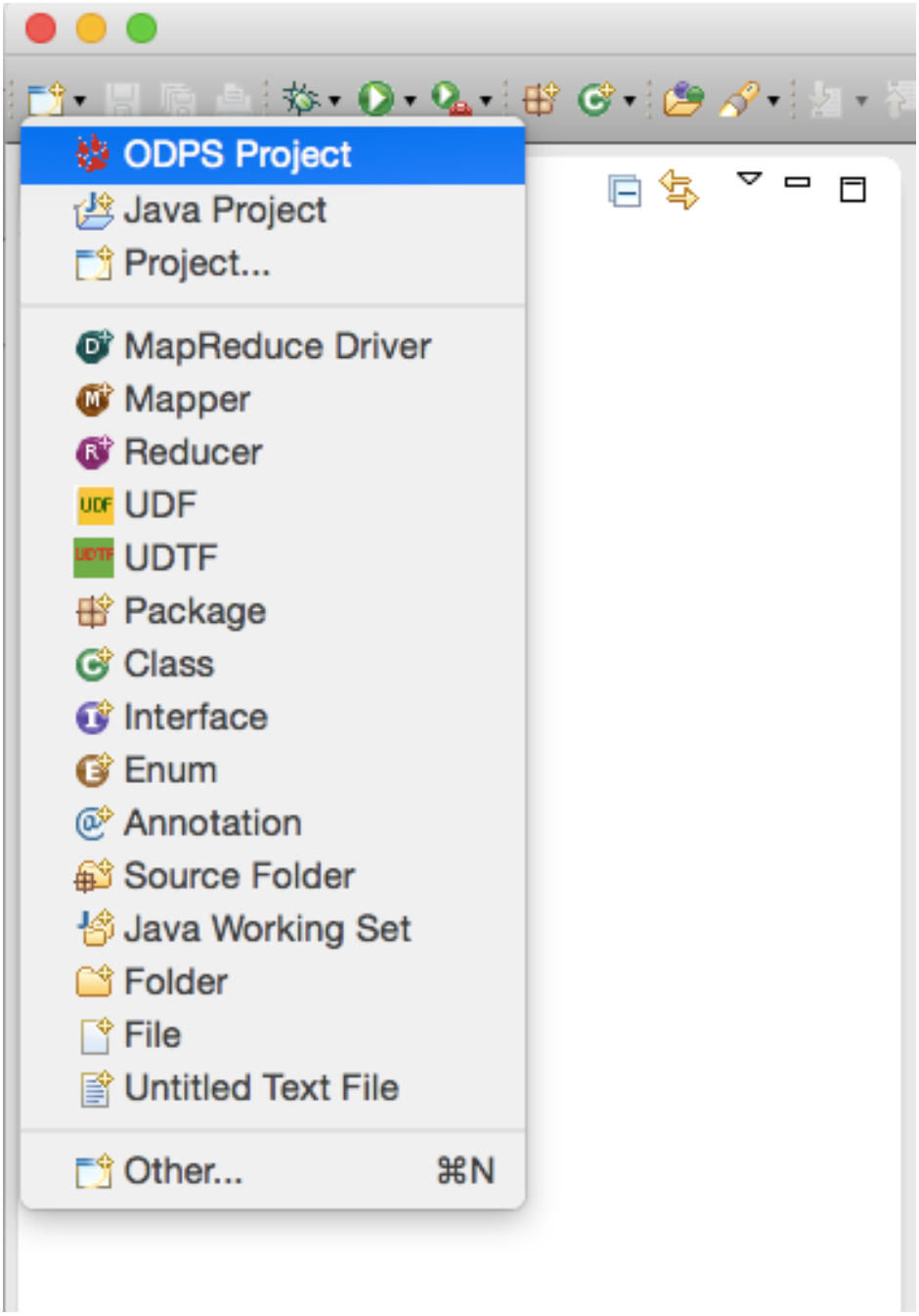
选择ODPS，随后点击OK键。同样在右上角会出现ODPS图标，表示插件生效：



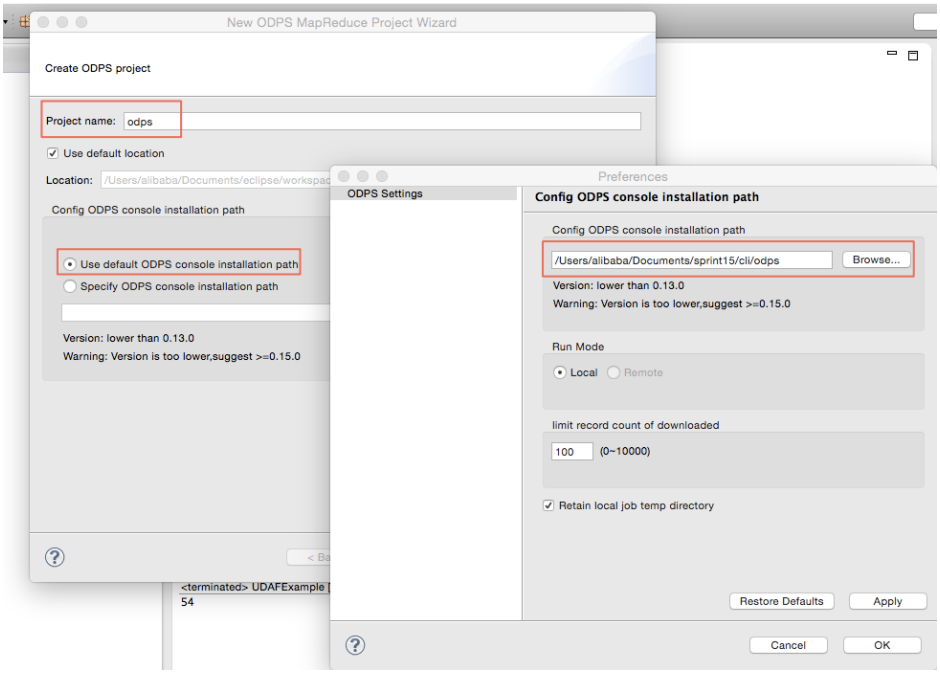
创建 MaxCompute 工程有两种方式。

方式一

在左上角选择文件(File) -> 新建(New)->Project->ODPS->ODPS Project，创建工程(示例中使用ODPS作为工程名)：



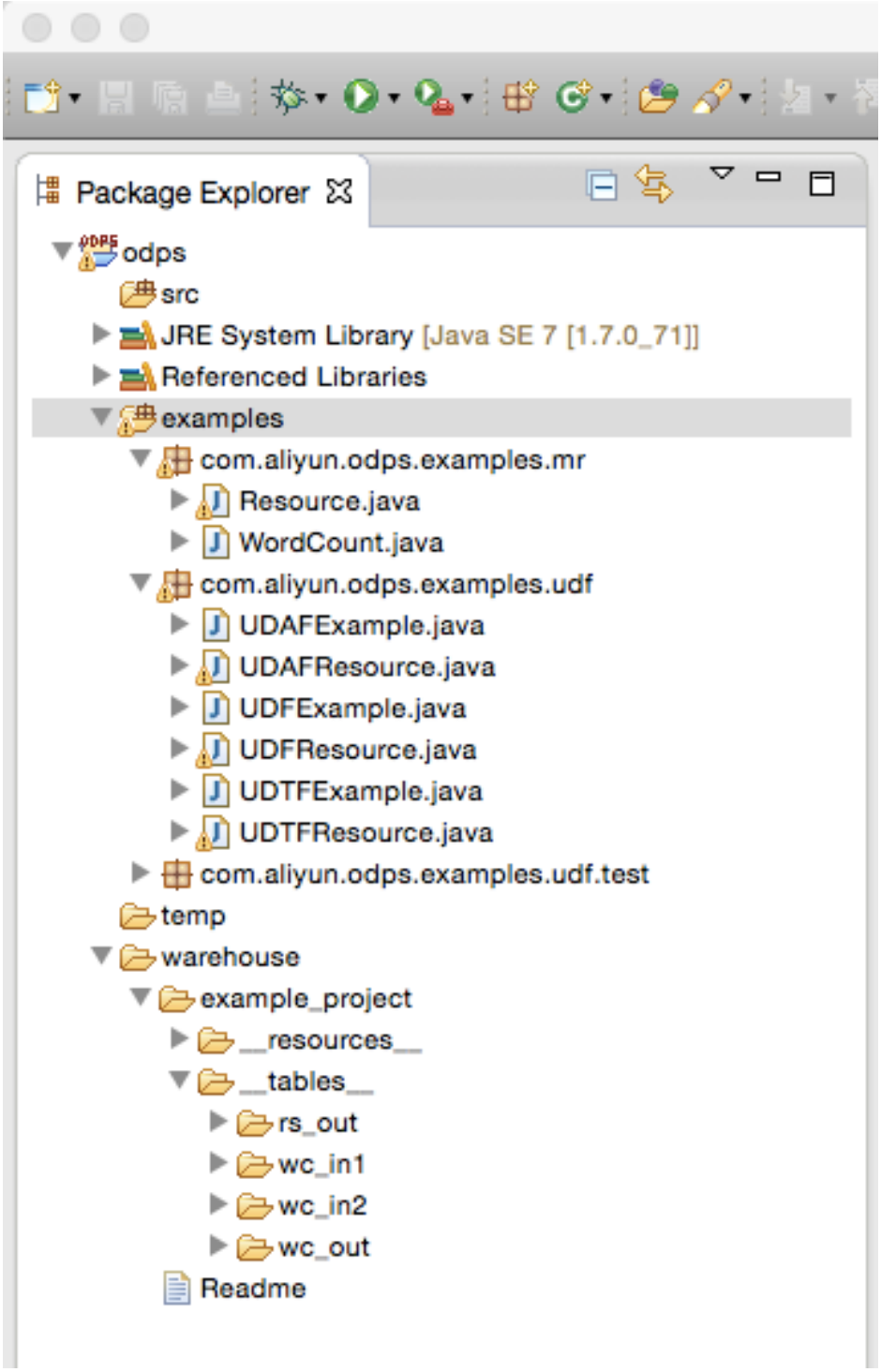
创建 MaxCompute 工程后会出现如下对话框。输入 Project name，选择 MaxCompute 客户端路径(客户端需要提前下载)，并确认(点击Finish)：



注解:

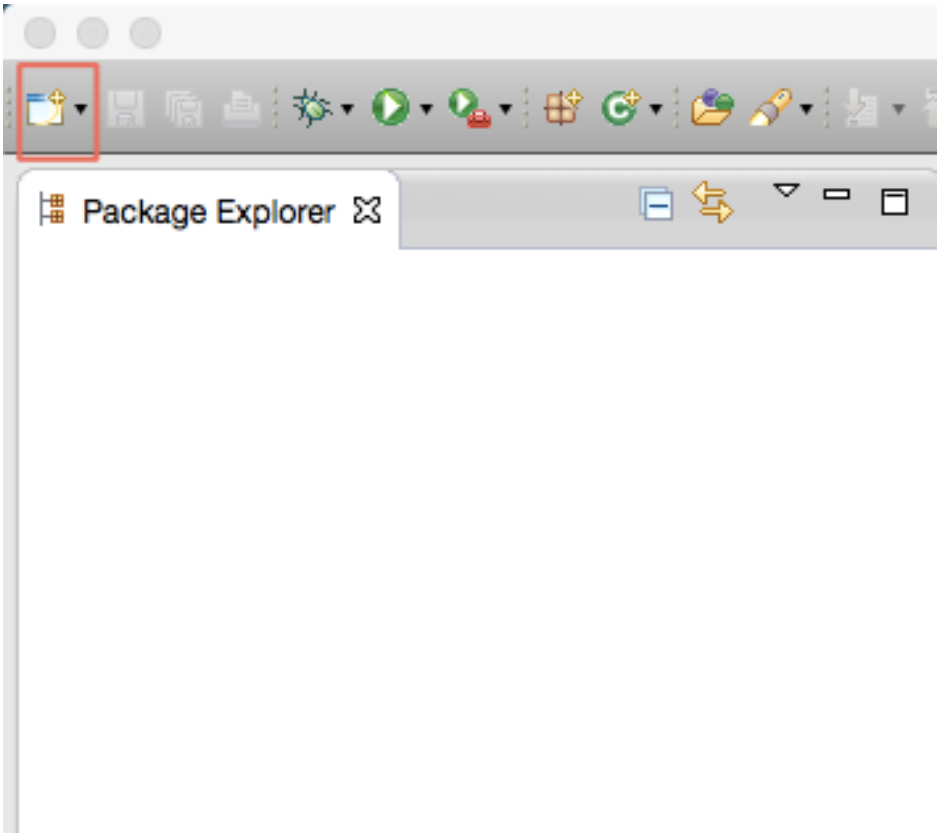
- 关于 MaxCompute 客户端的介绍请参考 客户端 。

创建好工程后，在左侧包资源管理器(Package Explorer)中可以看到如下目录结构：

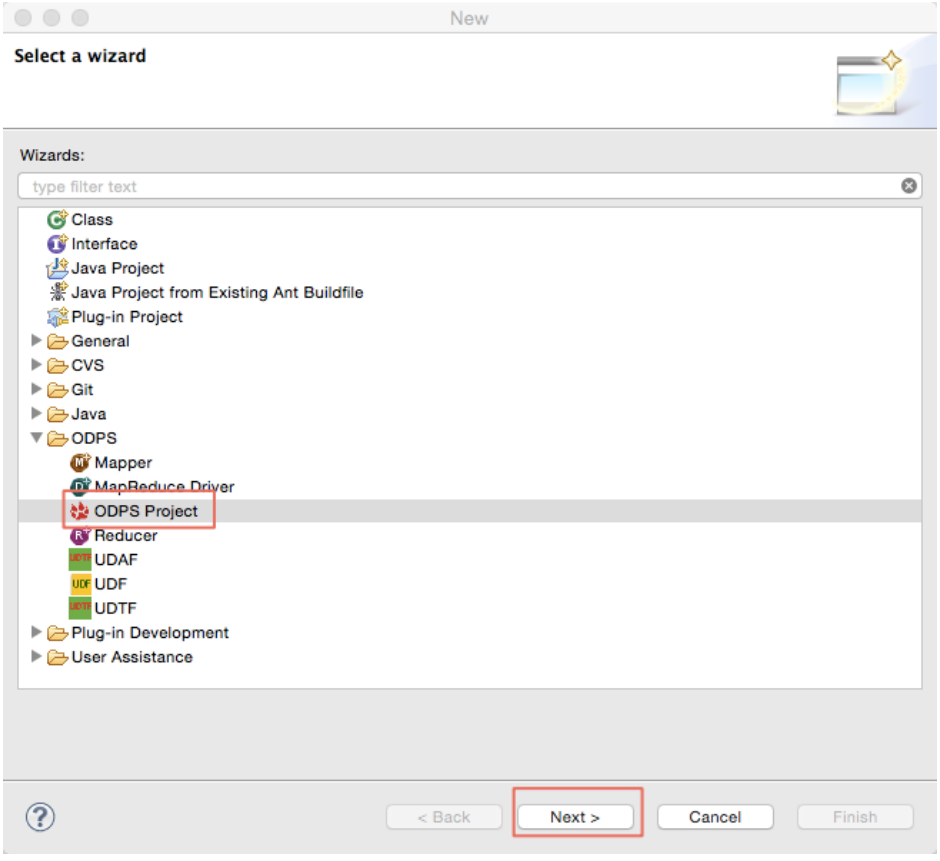


方式二

直接点击左上角的”新建”：



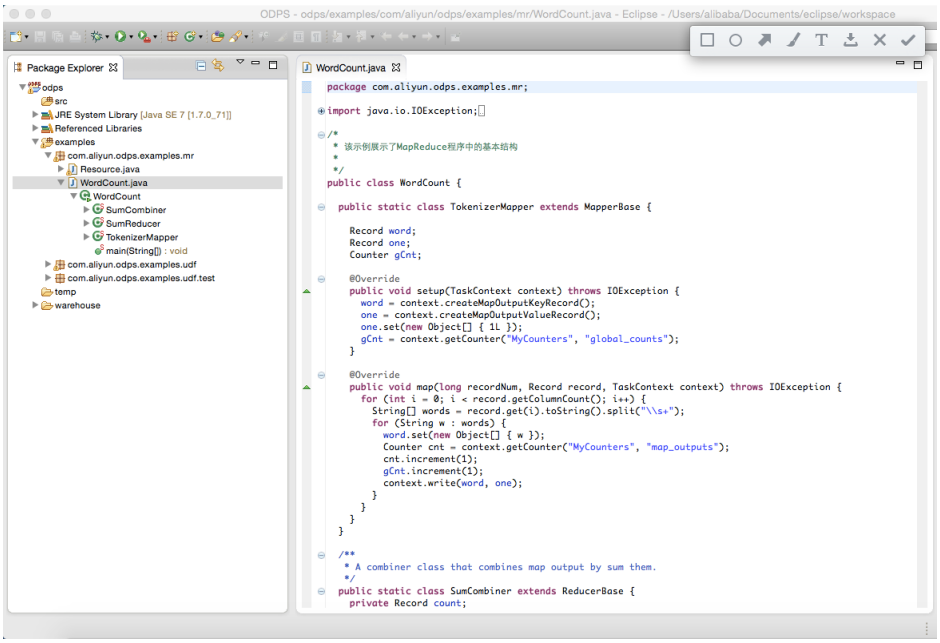
弹出对话框后，选择”ODPS Project”,点击”下一步”：



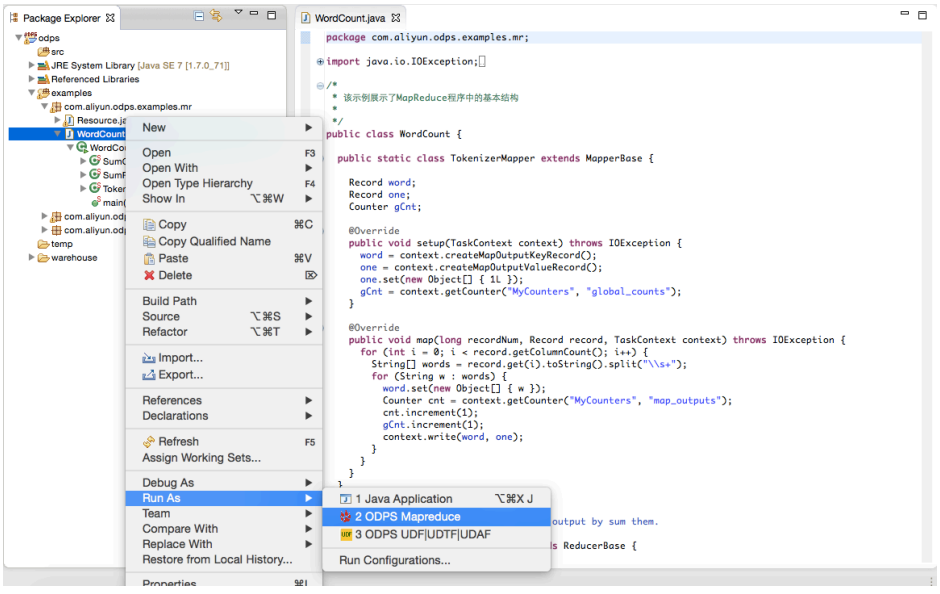
后续操作同方式一。

MaxCompute Eclipse插件安装结束。用户可以使用此插件编写MapReduce或者UDF程序。关于插件中对MapReduce的功能介绍请参考MapReduce开发插件介绍，UDF的程序编写请参考UDF开发插件介绍

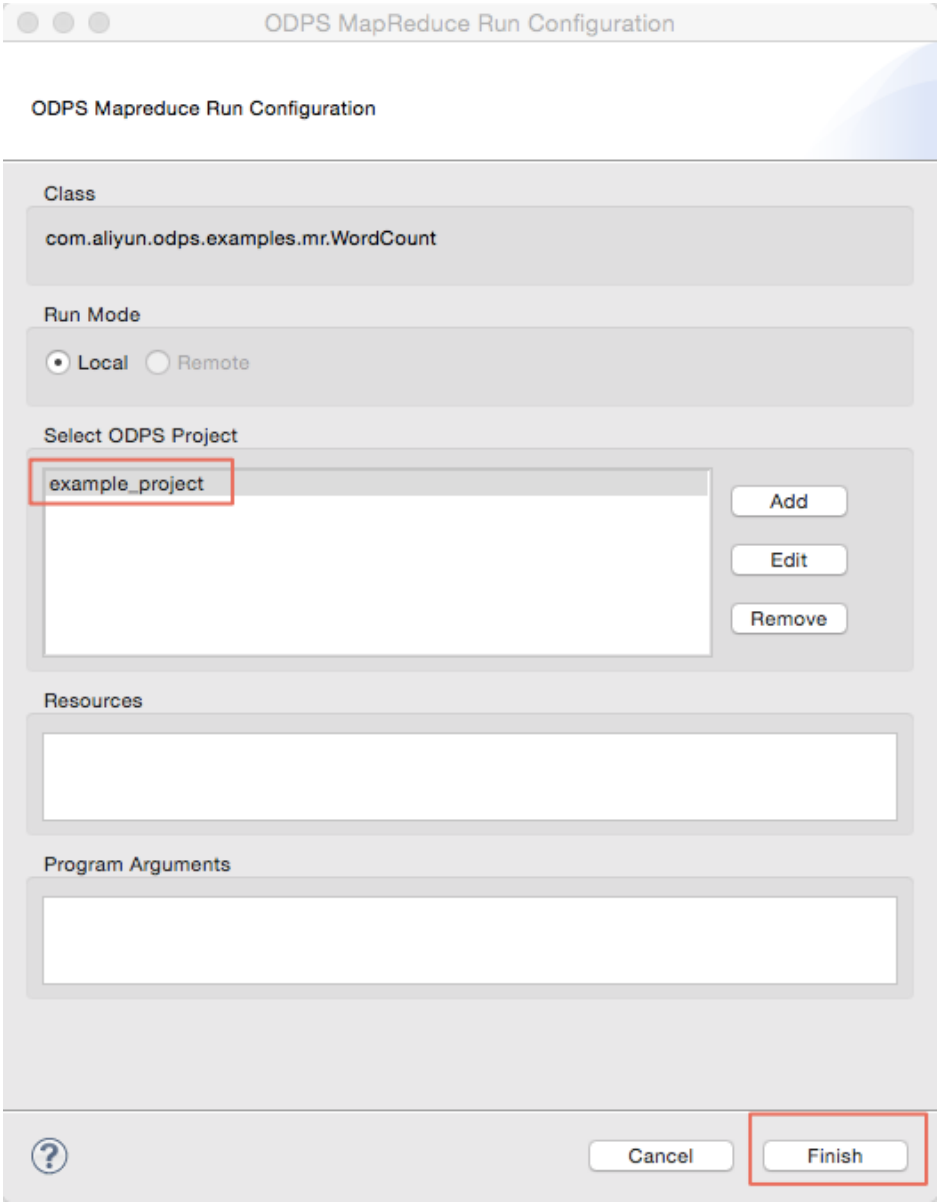
选择ODPS项目中的WordCount示例:



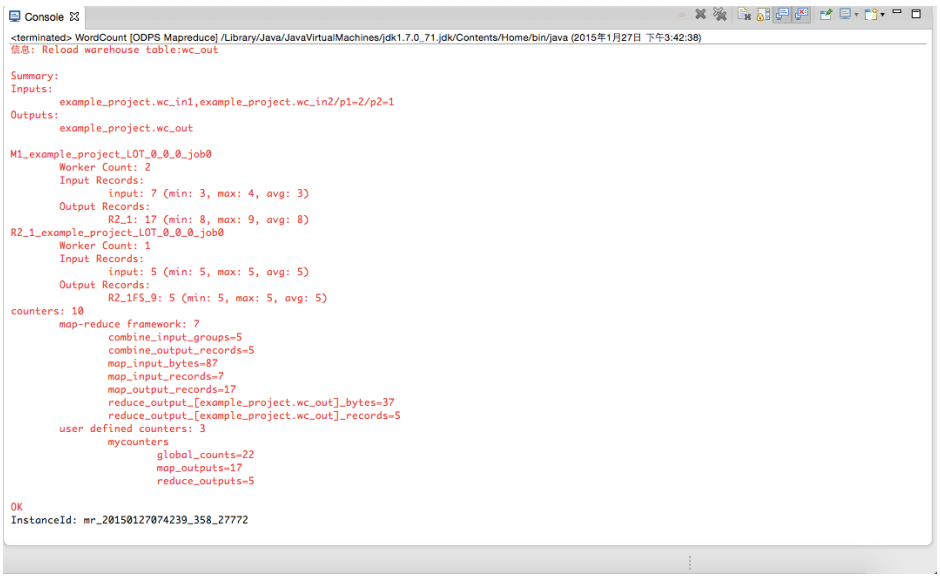
右键”WordCount.java”，依次点击”Run As”，”ODPS MapReduce”：



弹出对话框后，选择”example_project”，点击确认：

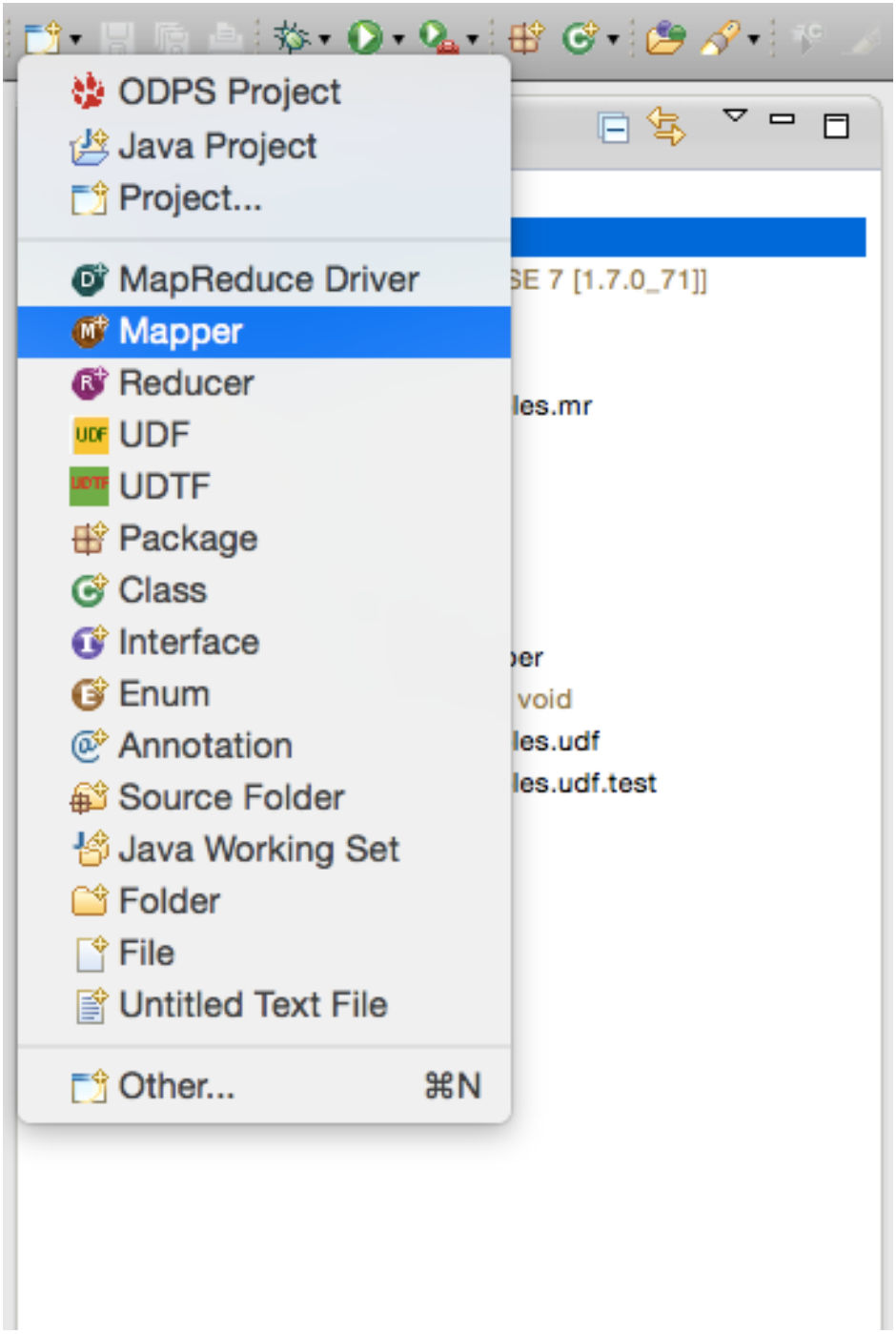


运行成功后，会出现以下结果提示：

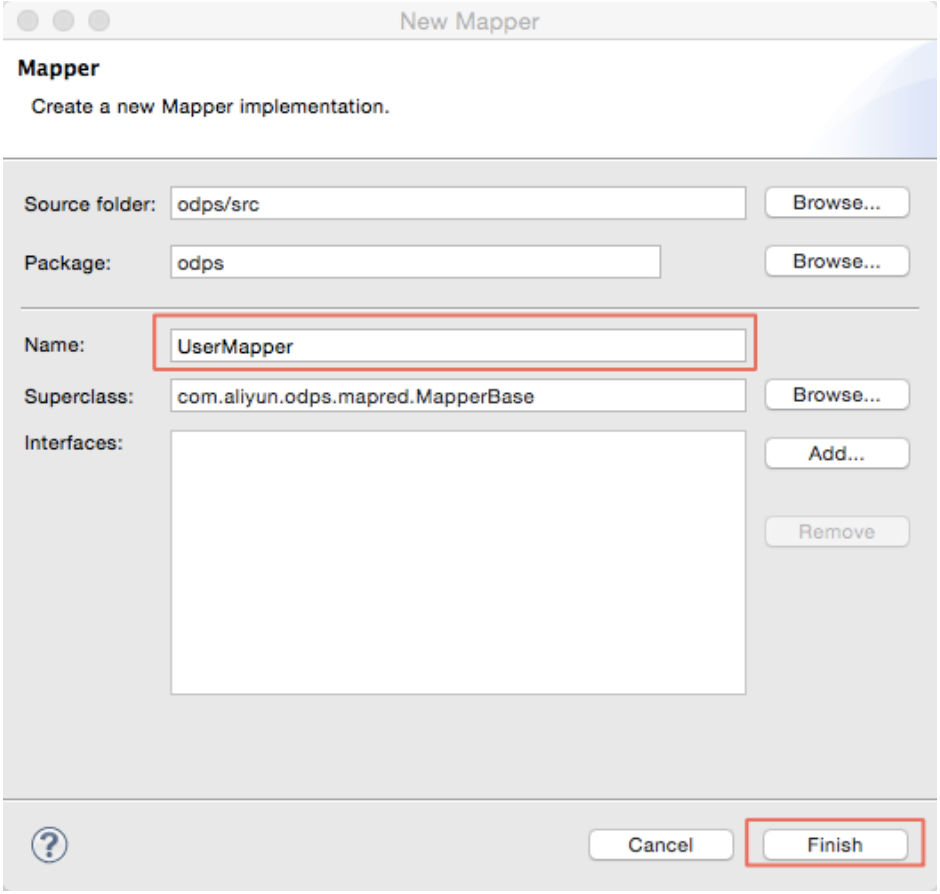


运行自定义MapReduce程序

右键选择src目录，选择新建(New) -> Mapper:



选择Mapper后出现下面的对话框。输入Mapper类的名字，并确认：



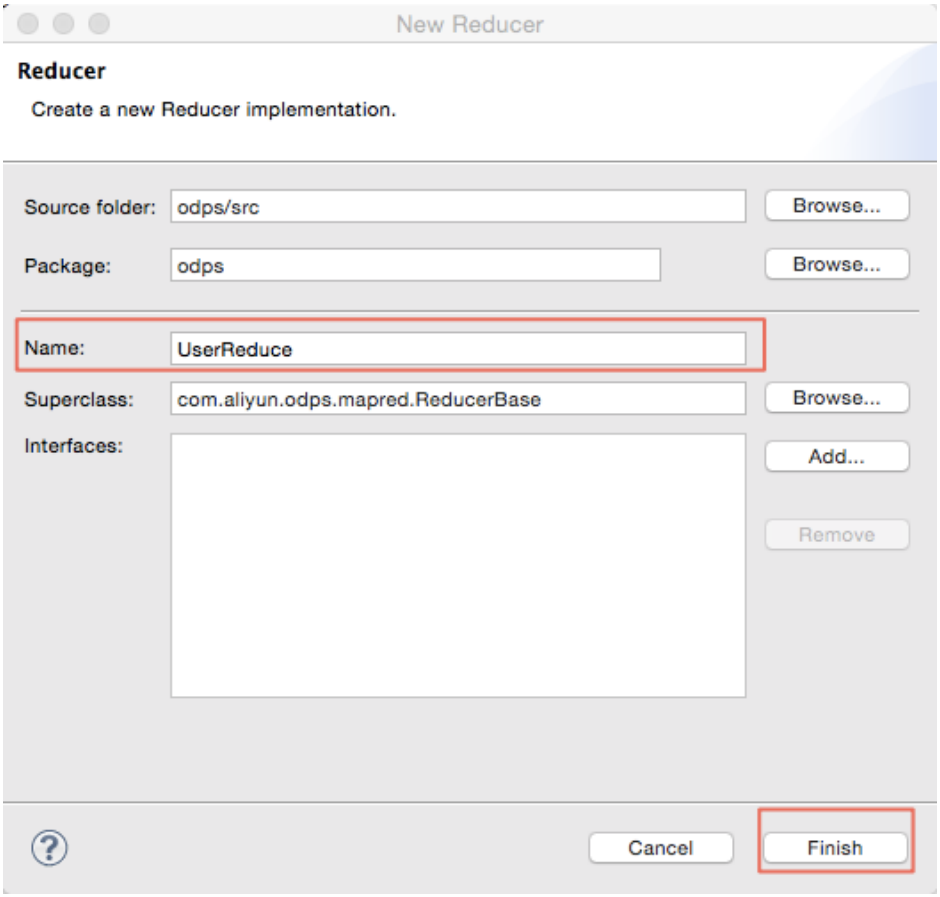
会看到在左侧包资源管理器(Package Explorer)中，src目录下生成文件UserMapper.java。该文件的内容即是一个Mapper类的模板：

```
package odps;

import java.io.IOException;

import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.MapperBase;

public class UserMapper extends MapperBase {

@Override
public void setup(TaskContext context) throws IOException {
}

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

@Override
public void cleanup(TaskContext context) throws IOException {
}

}
```

模板中，将package名称默认配置为”odps”，用户可以根据自己的需求进行修改。编写模板内容：

```
package odps;

import java.io.IOException;

import com.aliyun.odps.counter.Counter;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.MapperBase;

public class UserMapper extends MapperBase {

Record word;
Record one;
Counter gCnt;

@Override
public void setup(TaskContext context) throws IOException {
word = context.createMapOutputKeyRecord();
one = context.createMapOutputValueRecord();
one.set(new Object[] { 1L });
gCnt = context.getCounter("MyCounters", "global_counts");
}

@Override
public void map(long recordNum, Record record, TaskContext context)
throws IOException {
for (int i = 0; i < record.getColumnCount(); i++) {
String[] words = record.get(i).toString().split("\\s+");
for (String w : words) {
word.set(new Object[] { w });
Counter cnt = context.getCounter("MyCounters", "map_outputs");
cnt.increment(1);
gCnt.increment(1);
context.write(word, one);
}
}
}

@Override
public void cleanup(TaskContext context) throws IOException {
}

}
```

同理，右键选择src目录，选择新建(New)->Reduce:



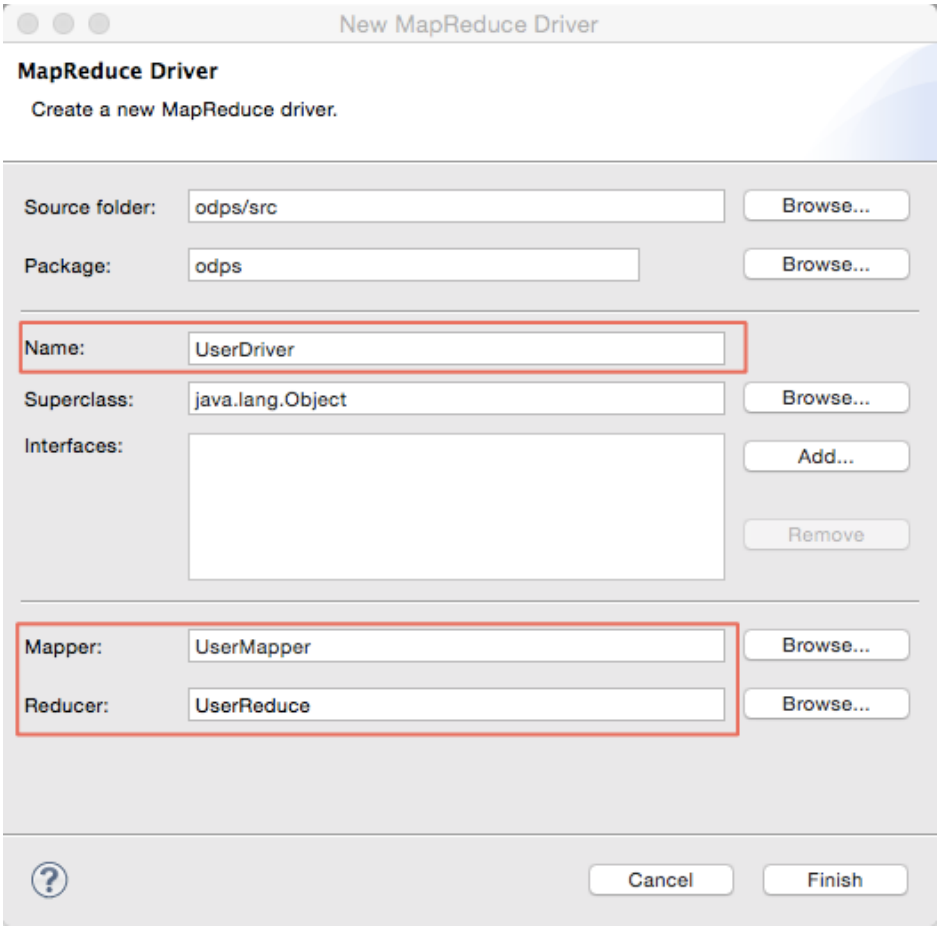
输入Reduce类的名字(本示例使用UserReduce):

同样在包资源管理器(Package Explorer)中，src目录下生成文件UserReduce.java。该文件的内容即是一个Reduce类的模板。编辑模板：

```
package odps;

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

import com.aliyun.odps.counter.Counter;
import com.aliyun.odps.data.Record;
import com.aliyun.odps.mapred.ReducerBase;

public class UserReduce extends ReducerBase {

private Record result;
Counter gCnt;

@Override
public void setup(TaskContext context) throws IOException {
result = context.createOutputRecord();
gCnt = context.getCounter("MyCounters", "global_counts");
}

@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);
Counter cnt = context.getCounter("MyCounters", "reduce_outputs");
cnt.increment(1);
gCnt.increment(1);

context.write(result);
}

@Override
public void cleanup(TaskContext context) throws IOException {
}

}
```

创建main函数: 右键选择src目录，选择新建(New) -> MapReduce Driver。填写Driver Name(示例中是UserDriver)，Mapper及Recduce类(示例中是UserMapper及UserReduce)，并确认。同样会在src目录下看到MyDriver.java文件：

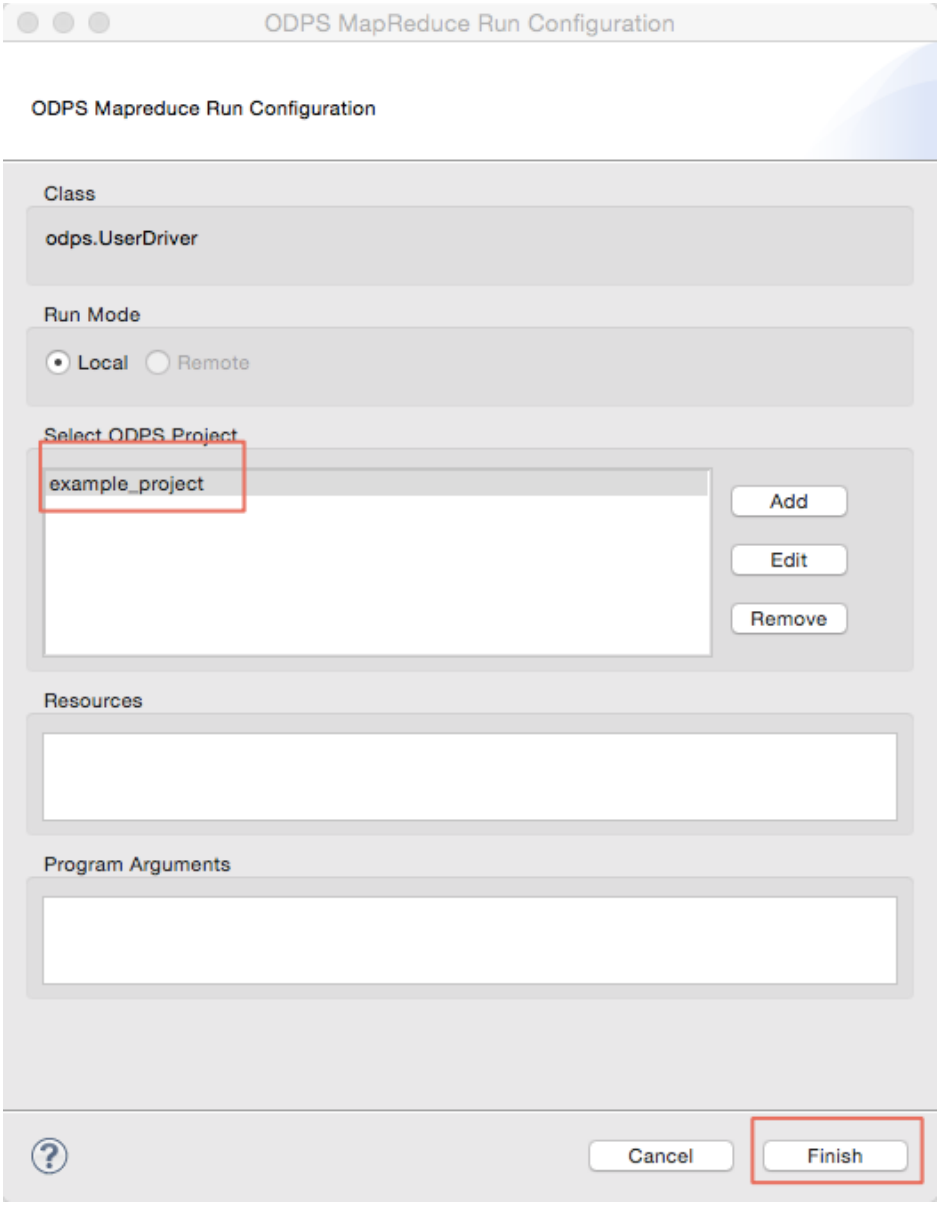


编辑driver内容：

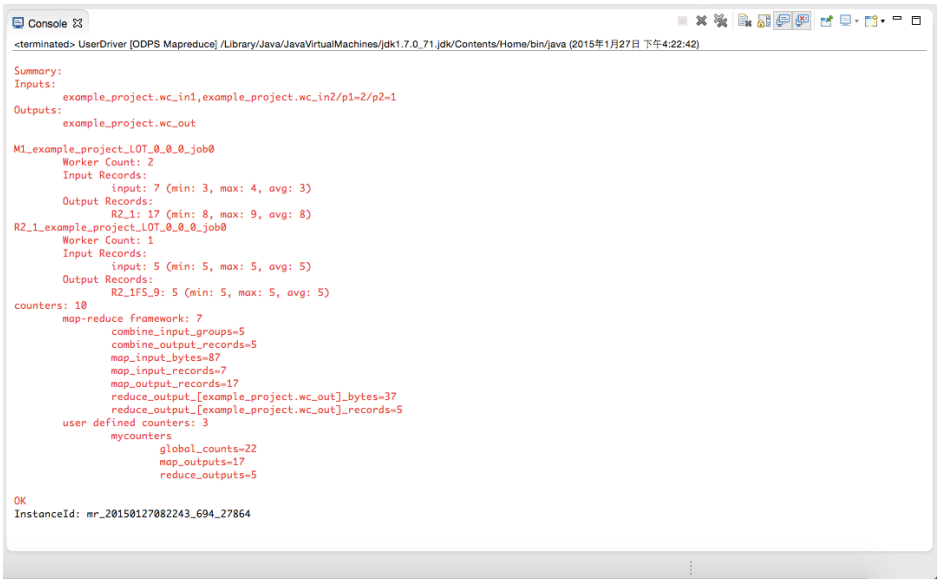
```
package odps;

import com.aliyun.odps.OdpsException;
import com.aliyun.odps.data.TableInfo;
import com.aliyun.odps.examples.mr.WordCount.SumCombiner;
import com.aliyun.odps.examples.mr.WordCount.SumReducer;
import com.aliyun.odps.examples.mr.WordCount.TokenizerMapper;
import com.aliyun.odps.mapred.JobClient;
import com.aliyun.odps.mapred.RunningJob;
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 UserDriver {

public static void main(String[] args) throws OdpsException {
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:bigint"));
```

```
InputUtils.addTable(
TableInfo.builder().tableName("wc_in1").cols(new String[] { "col2", "col3" }).build(), job);
InputUtils.addTable(TableInfo.builder().tableName("wc_in2").partSpec("p1=2/p2=1").build(), job);
OutputUtils.addTable(TableInfo.builder().tableName("wc_out").build(), job);

RunningJob rj = JobClient.runJob(job);
rj.waitForCompletion();
}

}
```

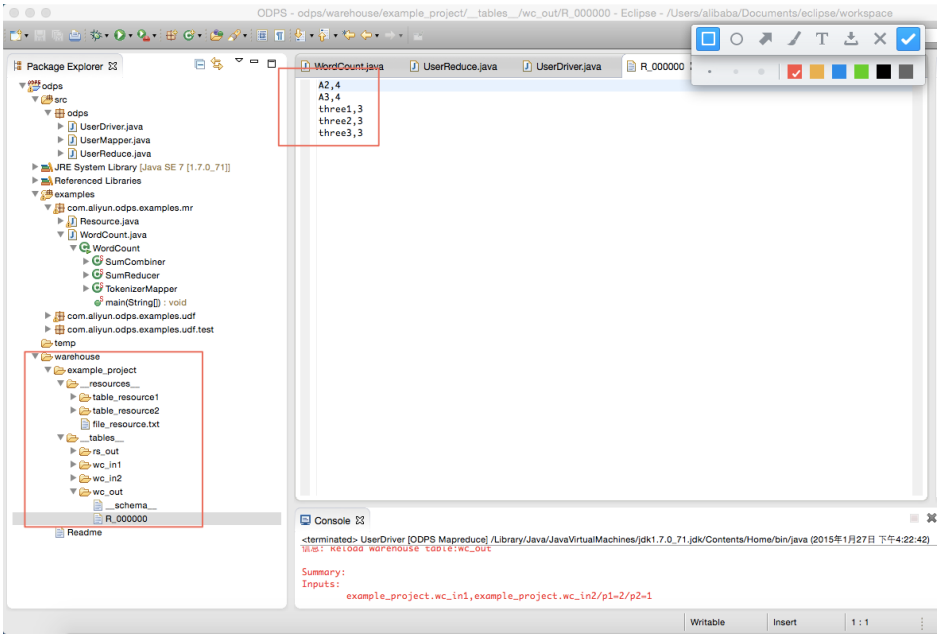
运行MapReduce程序,选中UserDriver.java，右键选择Run As -> ODPS MapReduce，点击确认。出现如下对话框：



选择ODPS Project为：example_project，点击Finish按钮开始本地运行MapReduce程序：

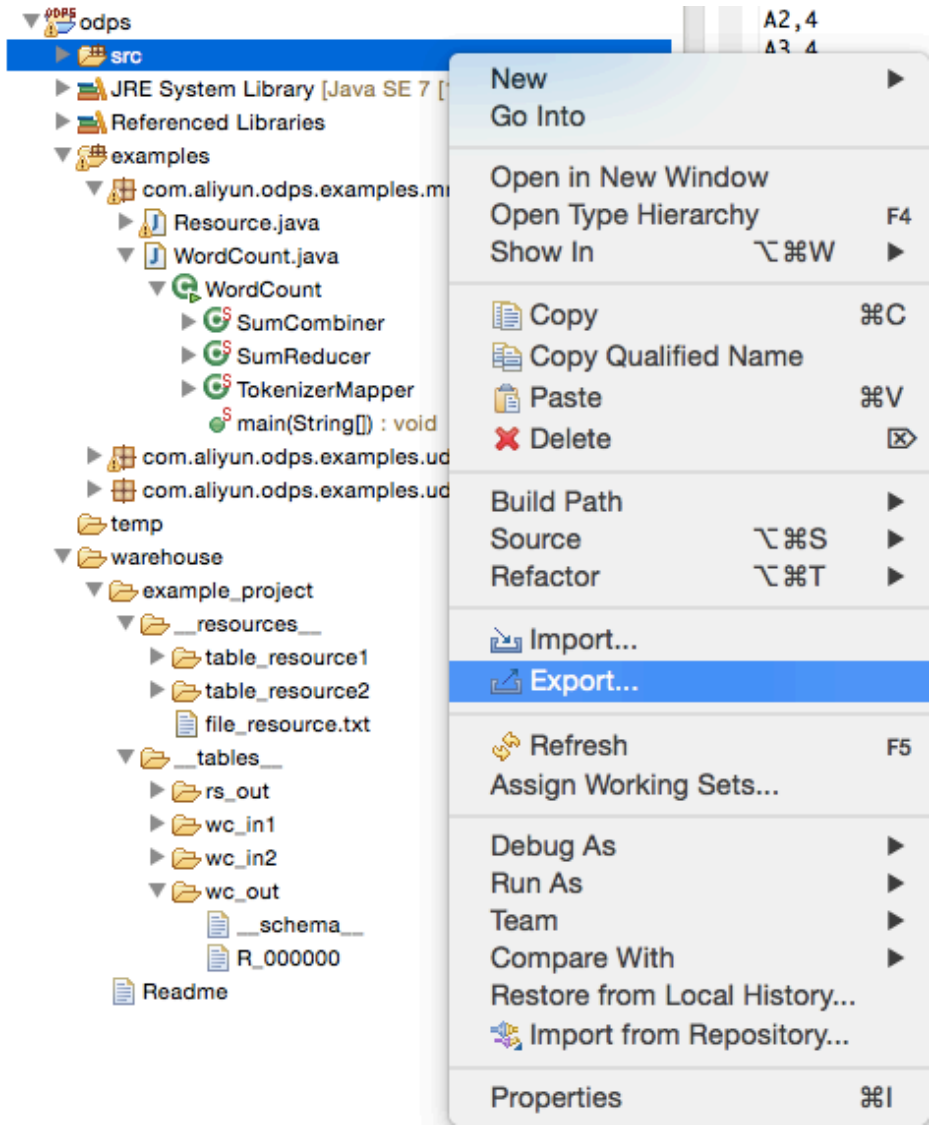


有如上输出信息，说明本地运行成功。运行的输出结果在warehouse目录下。关于warehouse的说明请参考 本地运行 。刷新ODPS工程：

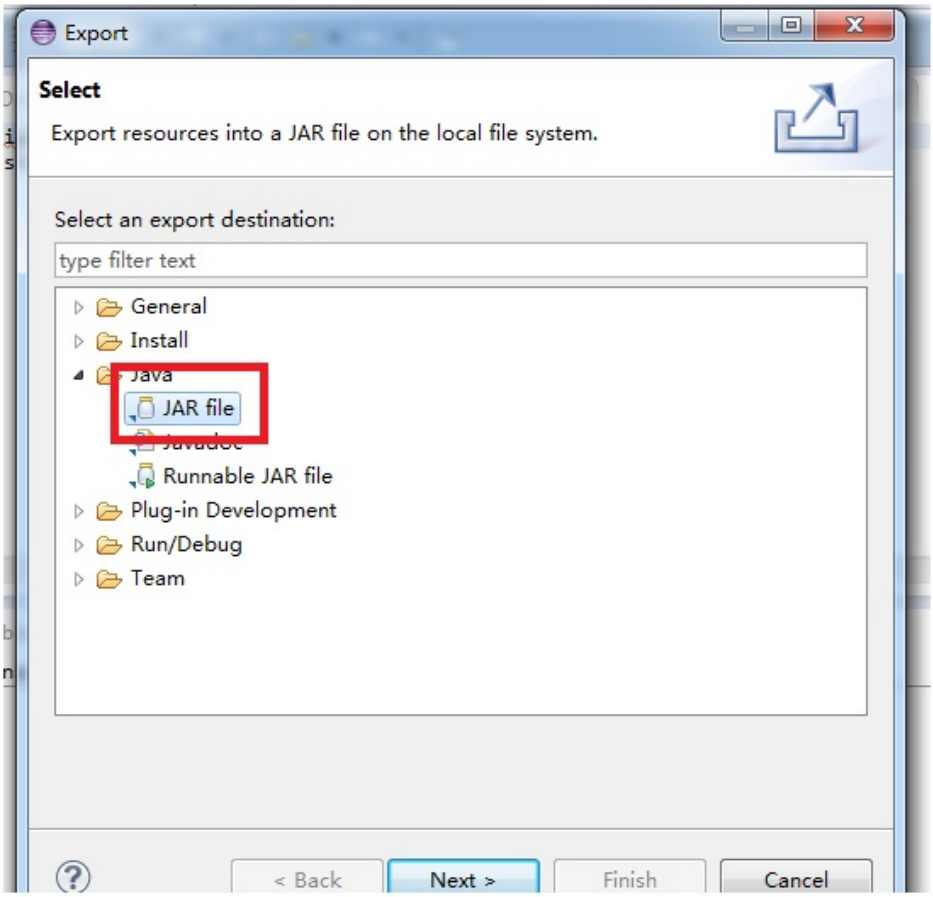


wc_out即是输出目录，R_000000即是结果文件。通过本地调试，确定输出结果正确后，可以通过Eclipse导出(Export)功能将MapReduce打包。打包后将jar包上传到ODPS中。在分布式环境下执行MapReduce，详情请参考 快速入门 。

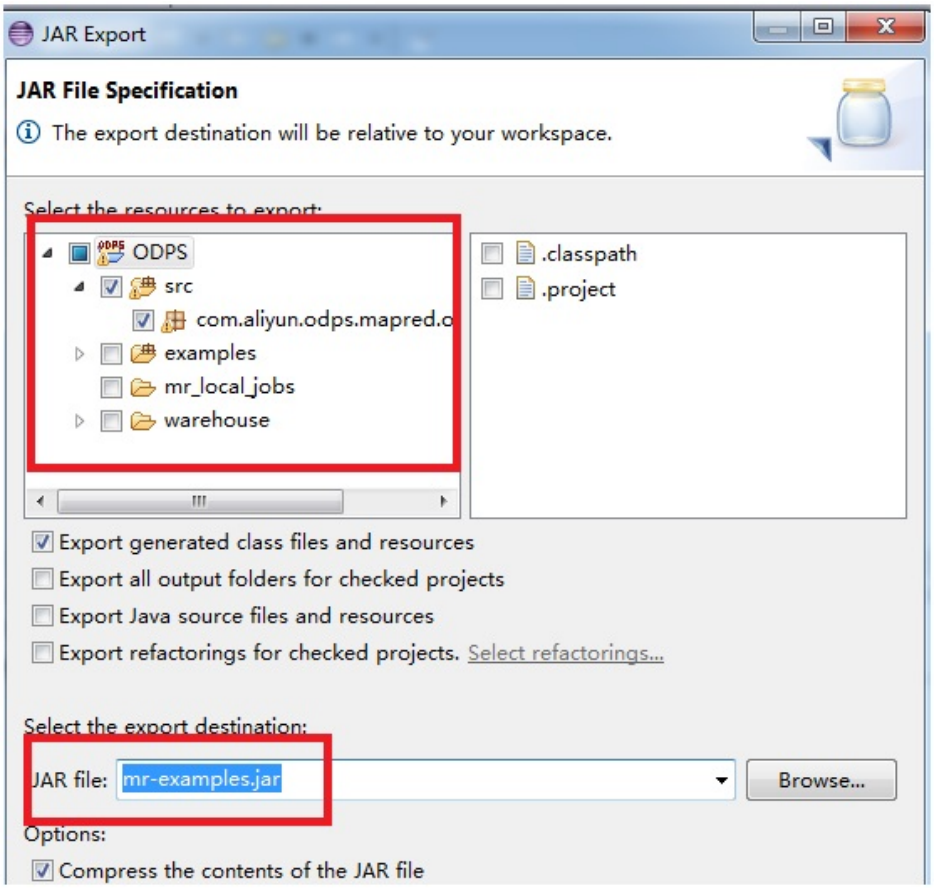
本地调试通过后，用户可以通过Eclipse的Export功能将代码打成jar包，供后续分布式环境使用。在本示例中，我们将程序包命名为mr-examples.jar。选择src目录，点击Export：



选择导出模式为Jar File：



仅需要导出src目录下package(com.aliyun.odps.mapred.open.example)，Jar File名称指定为”mr-examples.jar” ：



确认后，导出成功。

如果用户想在本地模拟新建Project，可以在warehouse下面，创建一个新的子目录(与example_project平级的目录)，目录层次结构为：

```
<warehouse>
|___example_project(项目空间目录)
|___ <__tables__>
| |__table_name1(非分区表)
| | |___ data(文件)
| | |
| | |___ <__schema__> (文件)
| |
| |__table_name2(分区表)
| |___ partition_name=partition_value(分区目录)
| | |___ data(文件)
| |
| |___ <__schema__> (文件)
|
|___ <__resources__>
|
|__table_resource_name (表资源)
| |___<__ref__>
|
|___ file_resource_name ( 文件资源 )
```

schema文件示例：

```
非分区表:
project=project_name
table=table_name
columns=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING

分区表:
project=project_name
table=table_name
columns=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING
partitions=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING

注：当前支持5种数据格式:bigint,double,boolean,datetime,string， 对应到java中的数据类型-
long,double,boolean,java.util.Date,java.lang.String。
```

data文件示例：

```
1,1.1,true,2015-06-04 11:22:42 896,hello world

\N,\N,\N,\N,\N

注：时间格式精确到毫秒级别，所有类型用\N表示null。
```

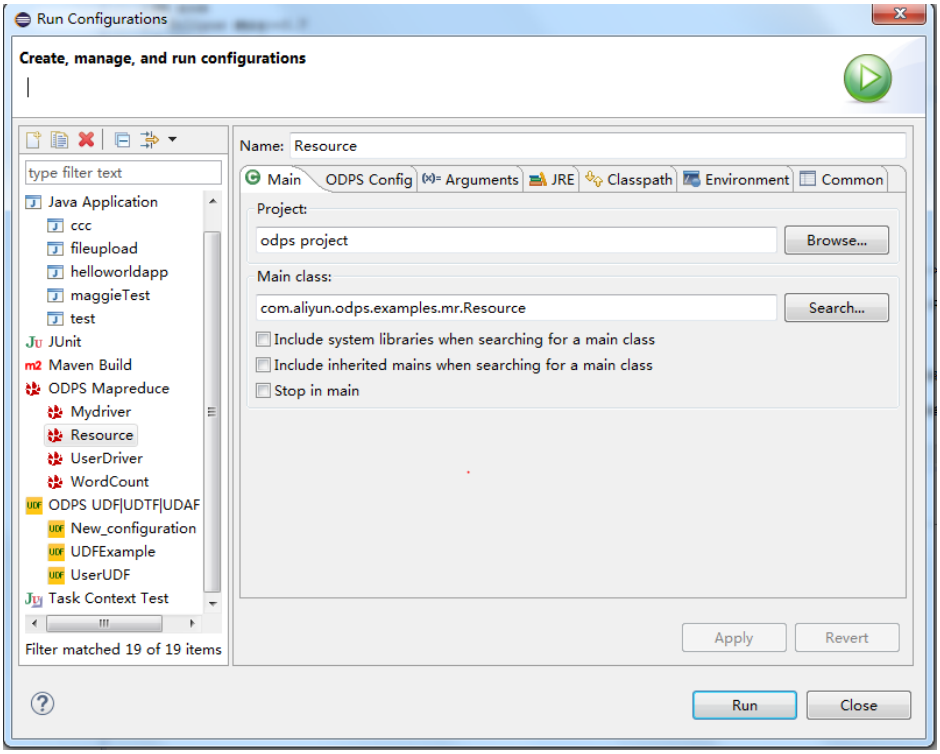
注解:

- 本地模式运行MapReduce程序，默认情况下先到warehouse下查找相应的数据表或资源，如果表或资源不存在会到服务器上下载相应的数据存入warehouse目录下，再以本地模式运行。
- 运行完MapReduce后，请刷新warehouse目录，才能看到生成的结果

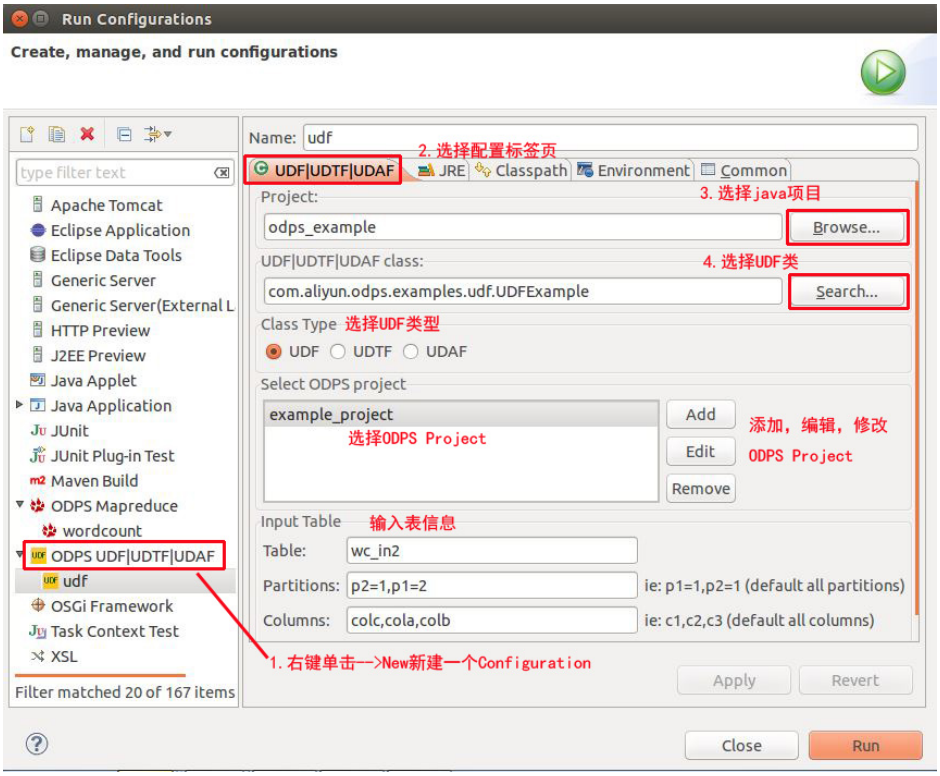
在本章节我们将介绍如何使用Eclipse插件开发并在本地运行UDF。UDAF,UDTF的编写执行过程与UDF类似，均可参考UDF的示例介绍完成。ODPS Eclipse插件提供两种运行UDF的方式：菜单栏和右键单击快速运行方式。

菜单栏运行

1. 从菜单栏选择Run—>Run Configurations...弹出如下对话框：



1. 用户可以新建一个Run Configuration，选择运行的UDF类及类型、选择ODPS Project、填写输入表信息，如：

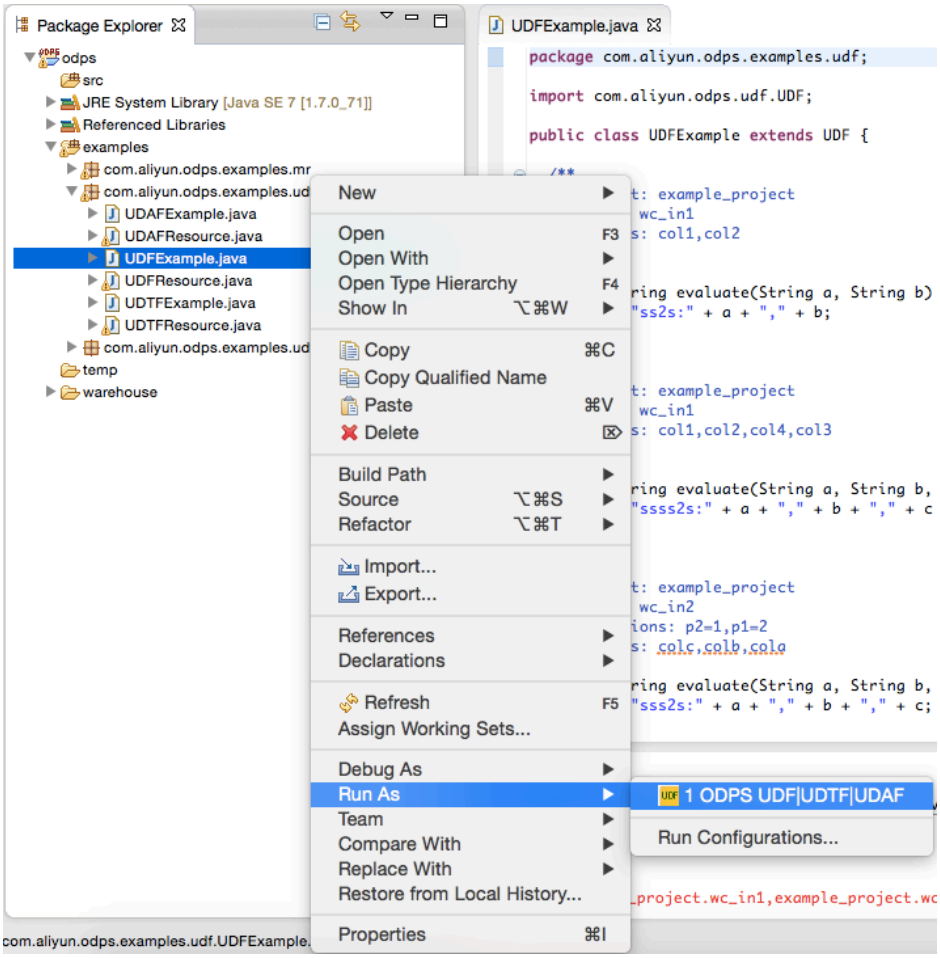


上述配置中，” Table”表示UDF的输入表，” Partitions”表示读取某个分区下的数据，分区由逗号分隔，“Columns”表示列，将依次作为UDF函数的参数被传入，列名由逗号分隔。

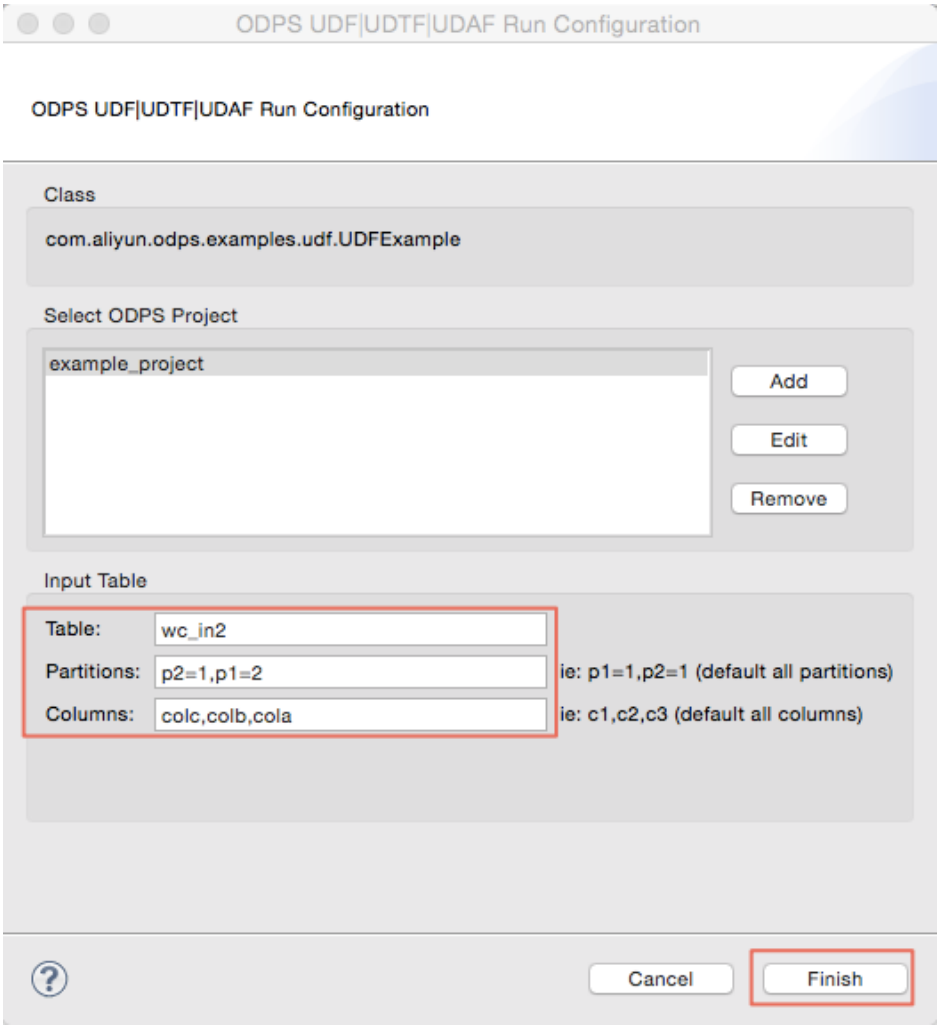
1. 点击”Run”运行，运行结果将显示在控制台中：

右键单击快速运行

1. 选中一个udf.java文件（比如：UDFExample.java）并单击鼠标右键，选择”Run As” -> “Run UDF|UDAF|UDTF”



1. 配置信息如下：



上述配置中，” Table”表示UDF的输入表，” Partitions”表示读取某个分区下的数据，分区由逗号分隔，” Columns”表示列，将依次作为UDF函数的参数被传入，列名由逗号分隔。

1. 点击” Finish”后，运行UDF，获得输出结果。

运行用户自定义UDF程序

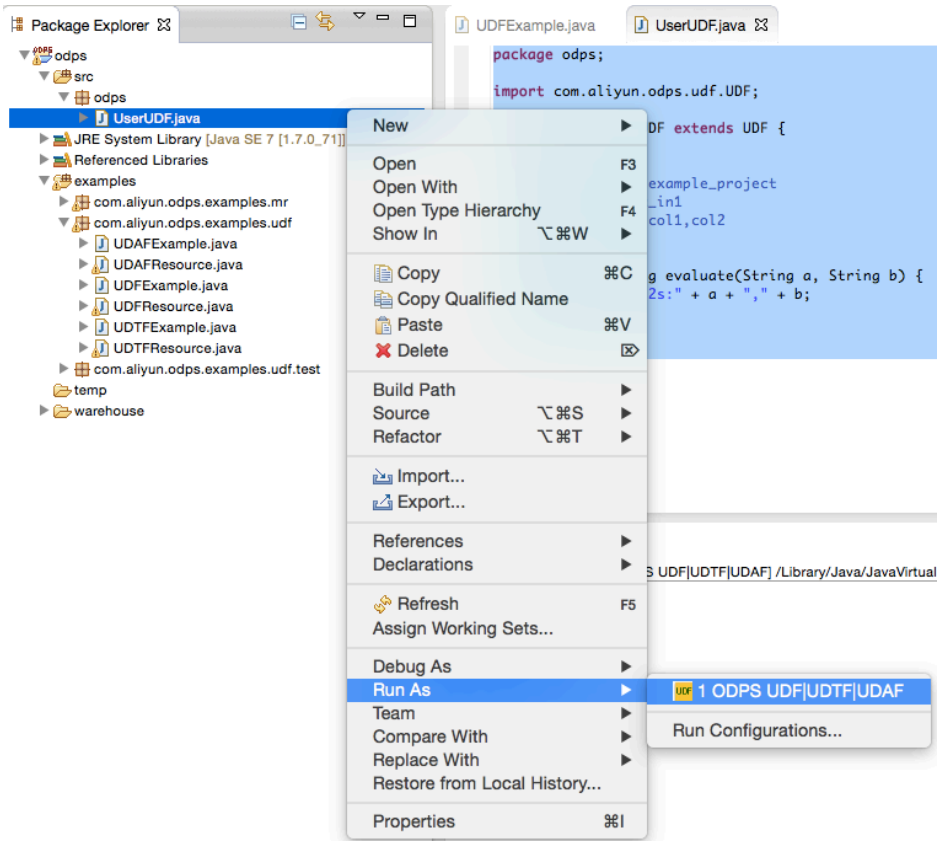
右击一个工程并选择” New—>UDF” （或者选择菜单栏File—>New—>UDF)。

填写UDF类名然后点击” Finish” 。在对应的src目录下生成与UDF类名同名的Java文件，编辑该java文件内容：

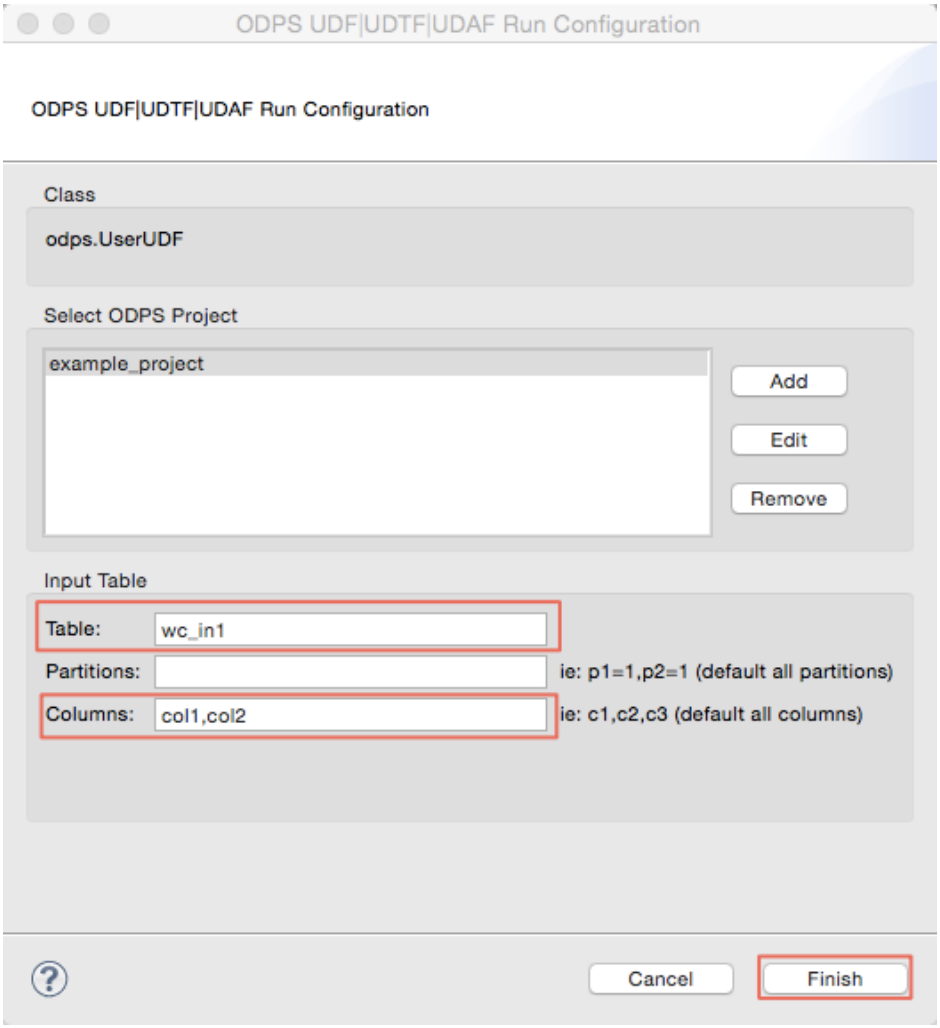
```
package odps;

import com.aliyun.odps.udf.UDF;

public class UserUDF extends UDF {

/**
* project: example_project
```

```
* table: wc_in1
* columns: col1,col2
*
*/
public String evaluate(String a, String b) {
return "ss2s:" + a + "," + b;
}

}
```

右击该java文件（如UserUDF.java），选择”Run As”，再选择”ODPS UDF|UDTF|UDAF”：



配置如下对话框：

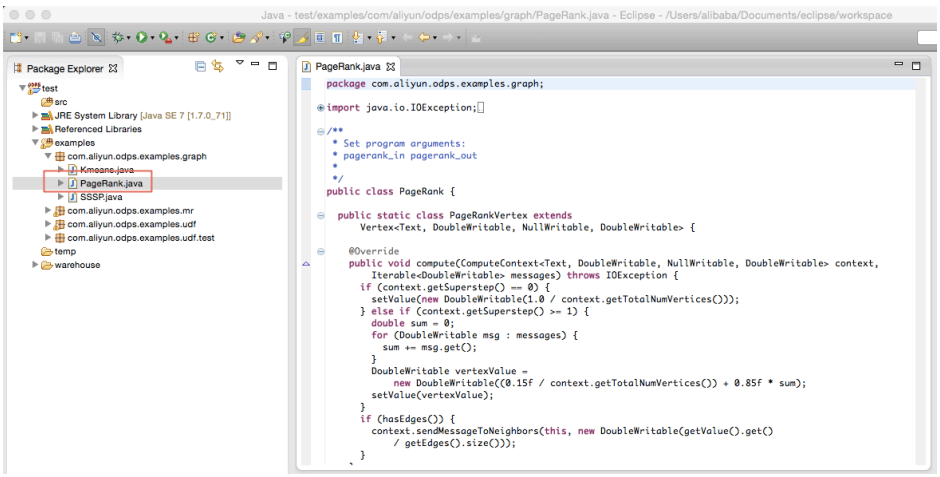


点击”finish”，得出结果：

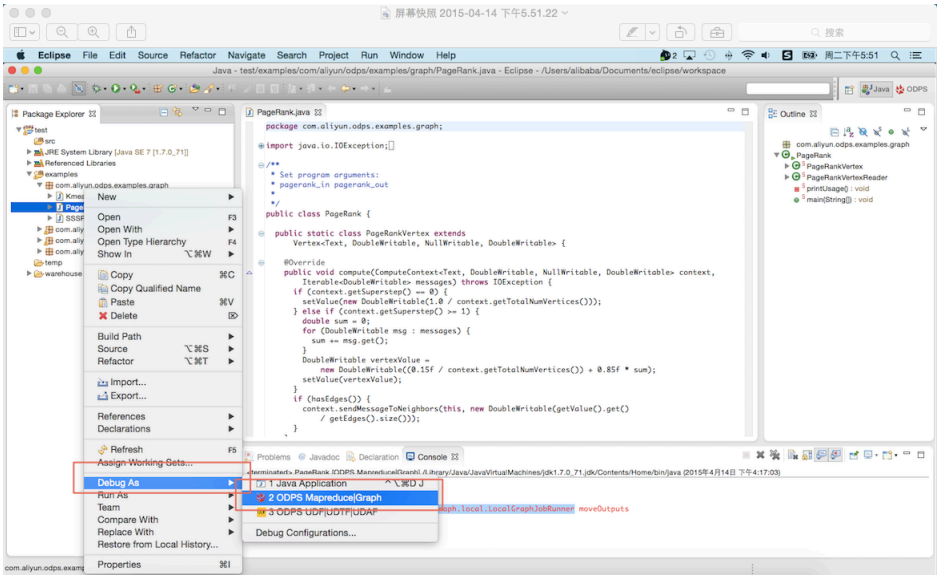
```
ss2s:A1,A2
ss2s:A1,A2
ss2s:A1,A2
ss2s:A1,A2
```

本示例中仅给出UDF的运行示例，UDTF的运行方式与UDF基本相同，不做特殊说明。

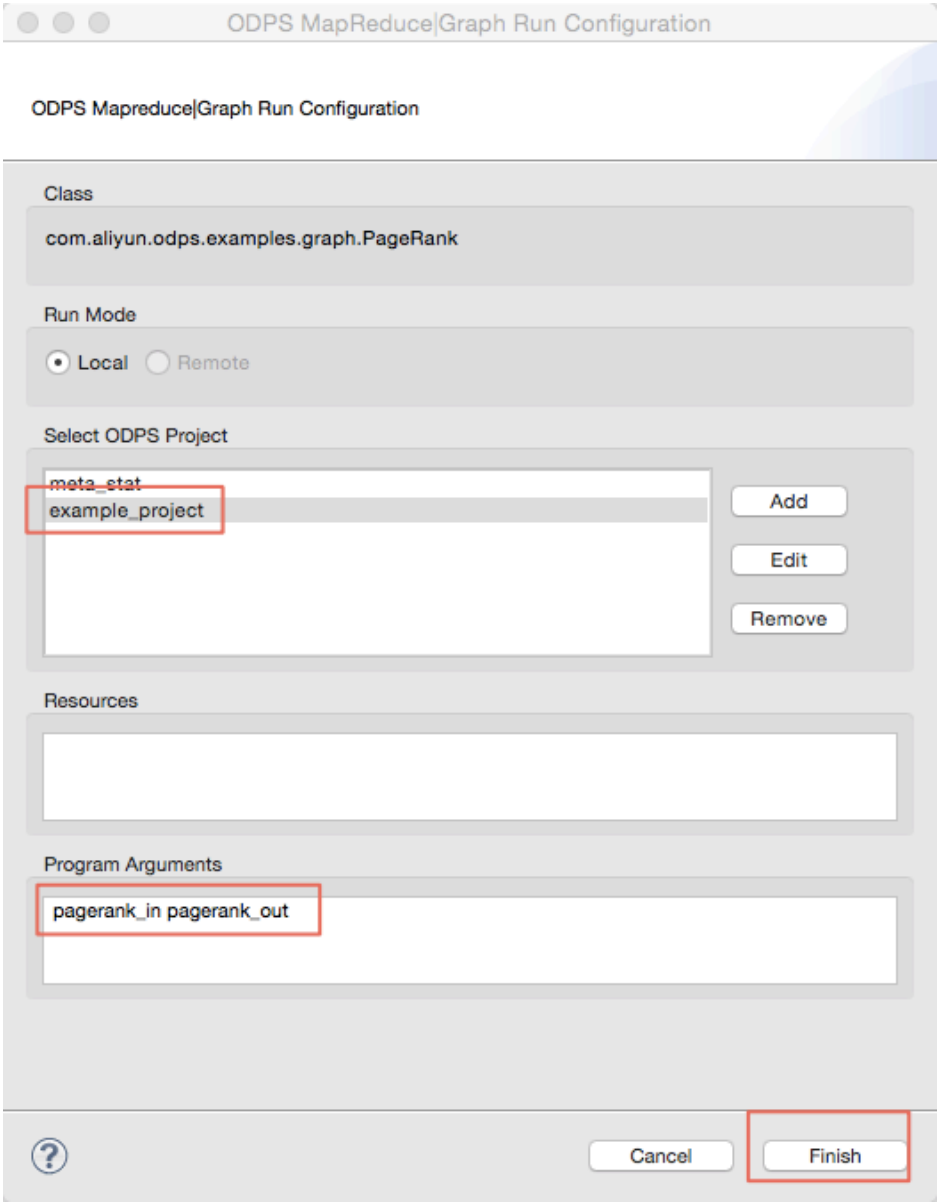
创建ODPS项目后，用户可以编写自己得Graph程序，依照以下步骤操作完成本地调试。 在此示例中，我们选用插件提供的"PageRank.java"来完成本地调试工作。 选中"examples"下的"PageRank.java"文件:



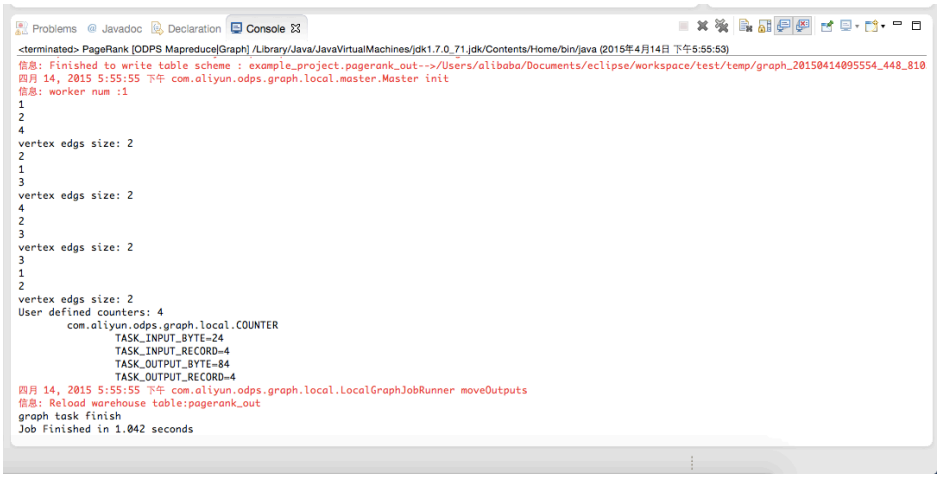
邮件点击，选择"Debug As" -> "ODPS MapReduce|Graph":



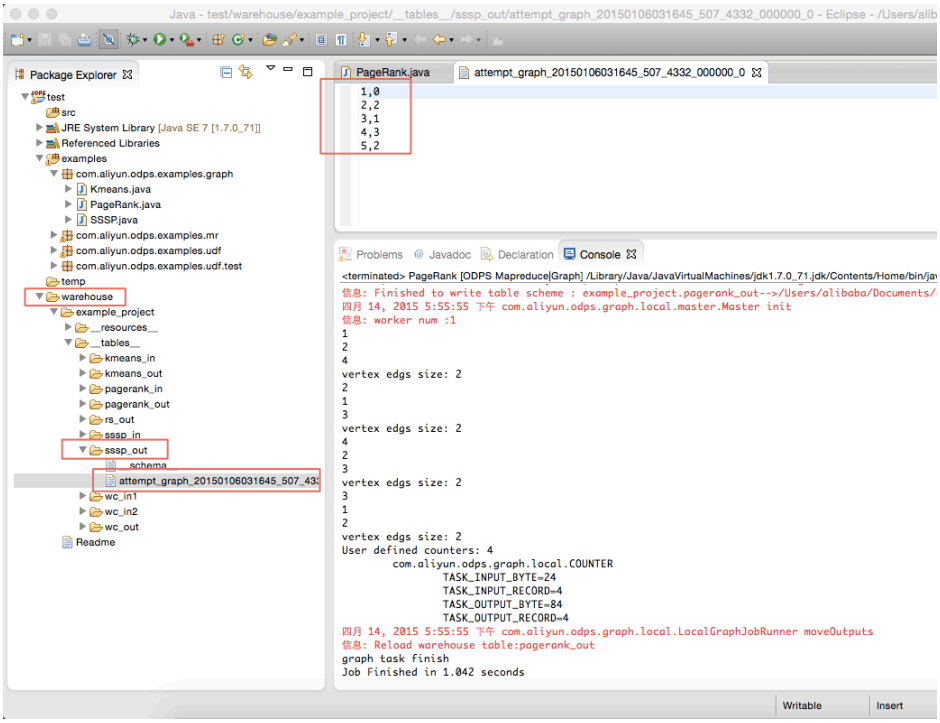
点击后出现对话框，作如下配置：


ODPS MapReduce|Graph Run Configuration
ODPS Mapreduce|Graph Run Configuration
Class
com.aliyun.odps.examples.graph.PageRank
Run Mode
Local
Remote
Select ODPS Project
meta_stat
example_project
Add
Edit
Remove
Resources
Program Arguments
pagerank_in pagerank_out
Cancel
Finish

查看作业运行结果：


Problems
Javadoc
Declaration
Console
<terminated> PageRank [ODPS Mapreduce|Graph] /Library/Java/JavaVirtualMachines/jdk1.7.0_71.jdk/Contents/Home/bin/java (2015年4月14日 下午5:55:53)
信息: Finished to write table scheme : example_project.pagerank_out-->/Users/alibaba/Documents/eclipse/workspace/test/temp/graph_20150414095554_448_810
四月 14, 2015 5:55:55 下午 com.aliyun.odps.graph.local.master.Master init
信息: worker num :1
1
2
4
vertex edgs size: 2
2
1
3
vertex edgs size: 2
4
2
3
vertex edgs size: 2
3
1
2
vertex edgs size: 2
User defined counters: 4
com.aliyun.odps.graph.local.COUNTER
TASK_INPUT_BYTE=24
TASK_INPUT_RECORD=4
TASK_OUTPUT_BYTE=84
TASK_OUTPUT_RECORD=4
四月 14, 2015 5:55:55 下午 com.aliyun.odps.graph.local.LocalGraphJobRunner moveOutputs
信息: Reload warehouse table:pagerank_out
graph task finish
Job Finished in 1.042 seconds

可以查看在本地的计算结果：



调试通过后，用户可以将程序打包，并以Jar资源的形式上传到ODPS，并提交Graph作业。

备注：

- 打包过程可参考MapReduce Eclipse插件介绍。
- 有关本地结果的目录结构介绍，也可以参考MapReduce Eclipse插件介绍。
- 关于上传Jar资源的详细介绍，可参考基本操作 中创建Jar资源部分。
- 有关提交Graph作业，可参考 Graph功能介绍中Jar命令的介绍。

- SDK下载信息：使用Maven的用户可以从Maven库中搜索”odps-sdk”获取不同版本的Java SDK
- 新版客户端下载：点击这里
- Eclipse开发插件下载：点击这里
- Intelij 开发插件下载：InteliJ开发插件