MaxCompute

Tools and Downloads

Tools and Downloads

Client

This article describes how to use the basic functions of the MaxCompute with the help of client command line tool. Before using the MaxCompute client, first install and configure the client.

NOTE:

- Do not perform the analysis operation based on the output format of the client. The output format of the client is not ensured for forward compatibility. Clients in different versions are different in their command formats and behaviors.
- For more information about basic commands of the client, see Basic commands.

After the client is installed and configured, you can use a command line to perform the following operations.

Get Help

To view the help information of the client, the command format is as follows:

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

You can also enter h; or help; (case-insensitive) in an interactive mode.

The client also provides the help [keyword]; command to get the command prompts related to the keyword. For example, enter help table; to get command prompts related to the table operation as follows:

```
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]
```

Start Parameters

When start the client, you can specify a series of parameters as follows:

```
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
```

Take the -f parameter as an example, the operation is as follows:

1. Prepare the local script file **script.txt**. Suppose that the file is located in the disk D, and the content is shown as follows:

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

2. Run the following command:

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

Interactive mode

Run the client to directly enter the interactive mode:

```
[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;
```

Enter the command at the cursor position (use a semicolon as a statement terminator), and press Enter to run.

Continuous running

- When using **-e** or **-f** option to run command, if there are multiple statements, and you want to start running from a middle statement, you can specify the parameter **-k**, indicating to ignore the previous statements and to start running from the specified position. When the parameter <= 0 is specified, the execution starts from the first statement.
- Each statement separated by a semicolon is considered as a valid statement. The statements which run successfully or fail to run are printed out at runtime.

For example, suppose there are three SQL statements in the file **/tmp/dual.sql**:

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

To ignore the first two statements, and start running from the third statement, the command format is as follows:

```
odpscmd –k 3 –f dual.sql
```

Get current logon user

To get current logon user, the command format is as follows:

```
whoami;
```

Use example:

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

Use the preceding command to get the current logon user Alibaba Cloud account, endpoint configuration, and project name.

Exit

To exit the client, the command format is as follows:

```
odps@ > quit;
```

You can also use the following command to exit the client:

```
odps@ > q;
```

MaxCompute Studio

What is Studio

MaxCompute Studio is a big data integrated development environment (IDE) tool that is provided by the Alibaba Cloud MaxCompute platform and installed on the developer’ s client. It is a development plug-in based on the popular integrated development platform IntelliJ IDEA, helping users develop data conveniently.

This article describes functional interfaces and common application scenarios of MaxCompute Studio.

Basic user interface

MaxCompute Studio is a plug-in on the IntelliJ IDEA platform, which shares basic development interfaces with IntelliJ IDEA. For more information about the IntelliJ IDEA interfaces, see the Interface operation guide.

Based on the IntelliJ IDEA interfaces, MaxCompute Studio provides the following functional interfaces:

SQL Editor: Provides features such as SQL syntax highlighting, code complementing, real-time error prompting, local compilation, and job submission.

- **Compiler View**: Displays locally compiled prompts and error messages, and locates the code in the editor.

Project Explorer: Connects to a MaxCompute project, and browses table structures, custom functions, and resource files in the project.

- **Table Details View**: Displays details and sample data of tables, views, and other resources.

Job Explorer: Browses and searches for historical jobs of MaxCompute.

Job Details View: Displays running details of a job, including the execution plan and

details of each execution task, that is, all information displayed using the Log view tool.

Job Output View: Displays output information of a running job.

Job Result View: Displays the output result of the SELECT job.

MaxCompute Console: Integrates the MaxCompute client, on which MaxCompute client commands can be input and executed.

Connect to MaxCompute project

Before using most features of MaxCompute Studio, you must Create a project connection. After the project connection is created, you can view related data structures and resource information in the **Project Explorer**. MaxCompute Studio automatically creates a local metadata backup task for each project to increase the access frequency to MaxCompute metadata and reduce the latency.

NOTE:

You must specify the target project connection to modify SQL scripts, submit jobs, view job information, open the MaxCompute console, and implement other functions using MaxCompute Studio. Therefore, creating a connection to the MaxCompute project is necessary.

For more information about MaxCompute projects, see Project.

For more information about project management using MaxCompute Studio, see Project space connection management.

Manage data

You can use the **Project Explorer** of MaxCompute Studio to quickly browse table structures, custom functions, and resource files in the project. The tree control can be used to list data tables, columns, partition columns, virtual views, custom functions, function signatures, and resource files and types of all project connections. It also supports fast locating.

You can double-click a data table to open the **Table Details View** and view metadata, structure, and sample data of the data table. If you do not have the permission for a project, an error message is prompted.

MaxCompute Studio integrates MaxCompute Tunnel and supports local data upload and download. For more information, see Import and export data.

Compile an SQL script

You can compile a MaxCompute SQL script on MaxCompute Studio.

Procedure

Open MaxCompute Studio and select **File > New > Project** or **File > New > Module**.

Create a MaxCompute Studio project or module.

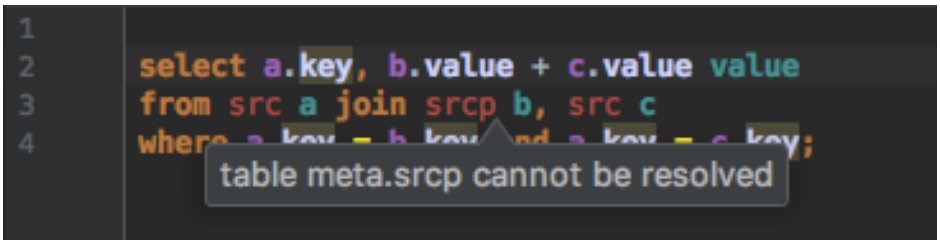
Select **File > New > MaxCompute Script** or right-click the menu and select **New > MaxCompute Script** to create a MaxCompute SQL script file.

NOTE:

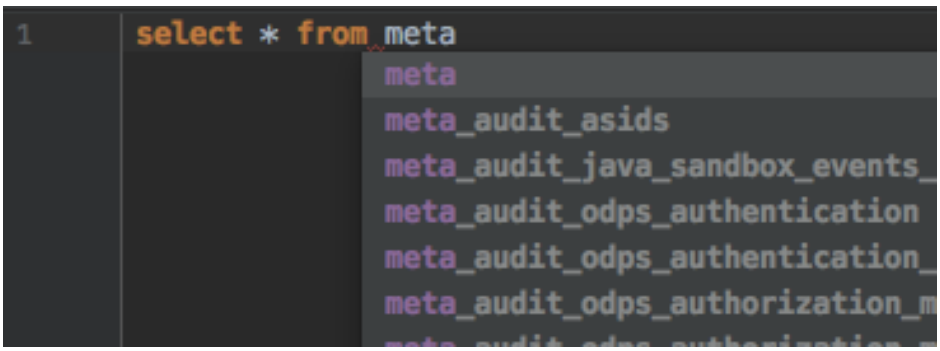
When a MaxCompute SQL script is created, MaxCompute Studio prompts you to select an associated MaxCompute project. You can also modify the associated project using the **project selector** on the right of the toolbar on the SQL editor. The editor automatically checks metadata (such as the table structure) and reports errors of an SQL statement based on the project associated with the SQL script. The editor also sends the SQL statement to the associated project for execution when it submits the SQL statement for running. For more information, see Compile an SQL script.

SQL code intelligent prompt

After you enter the code, the SQL editor provided by MaxCompute Studio intelligently prompts the syntax errors, type matching errors, or warnings of SQL statements, and marks them on the code in real time, as shown in the following figure.

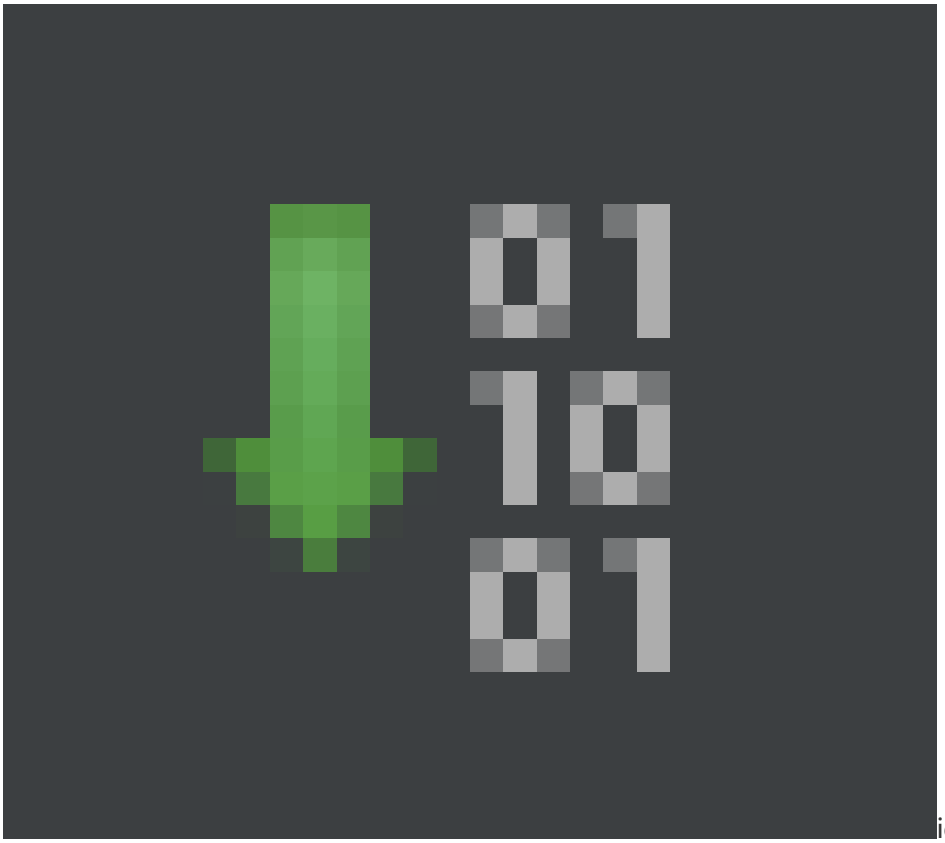


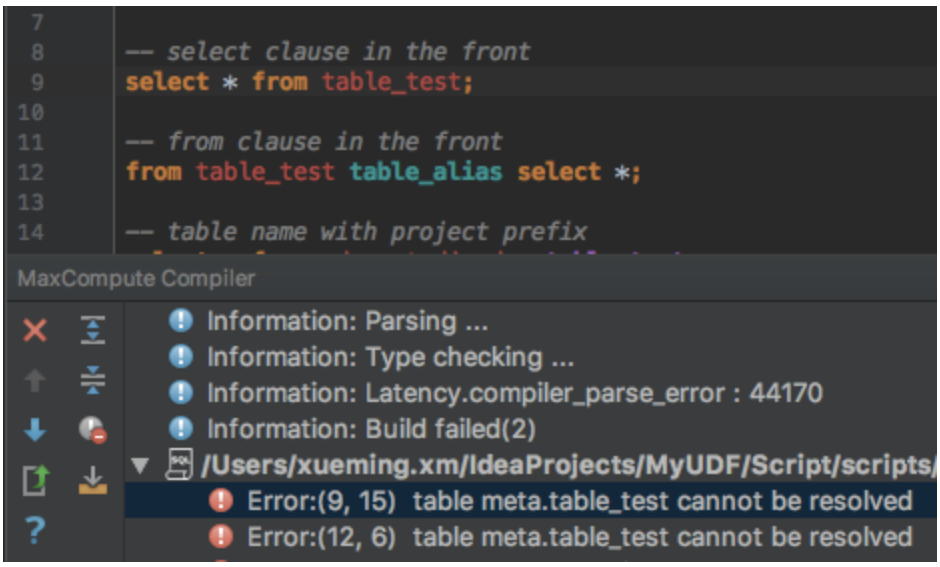
By using the code complementing function, MaxCompute Studio prompts you the name, table, field, function, type, and code keyword of a project based on the code context, and automatically complements the code based on your selections, as shown in the following figure.



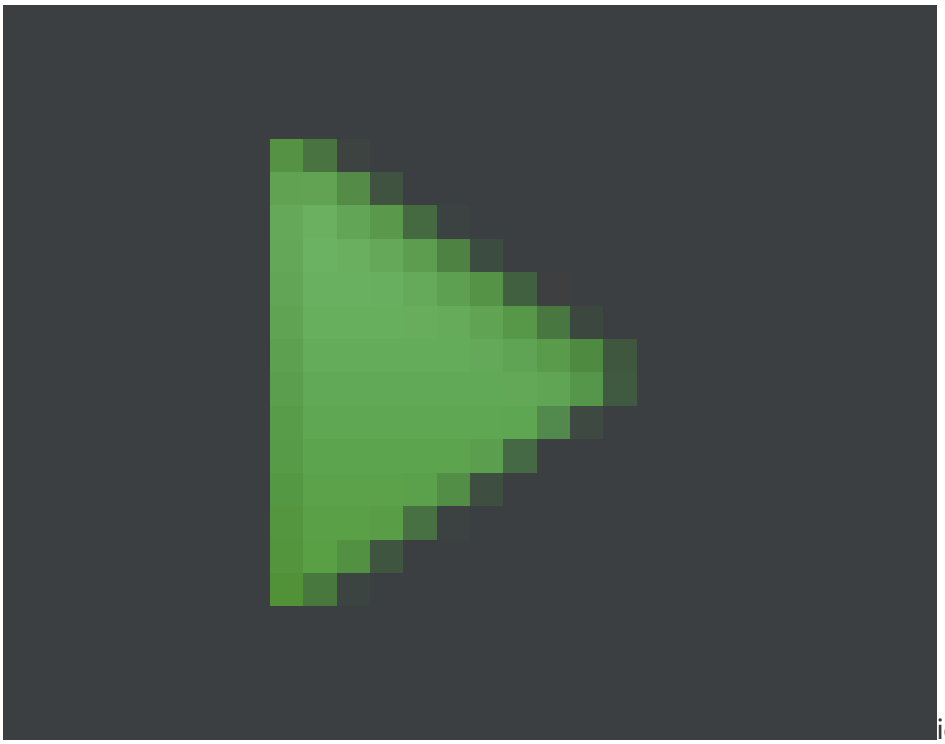
Compile and submit a job

Compile a job

Click the  icon on the toolbar of the SQL editor to locally compile an SQL script. If syntax or semantic errors occur, the editor reports it.



Submit a job

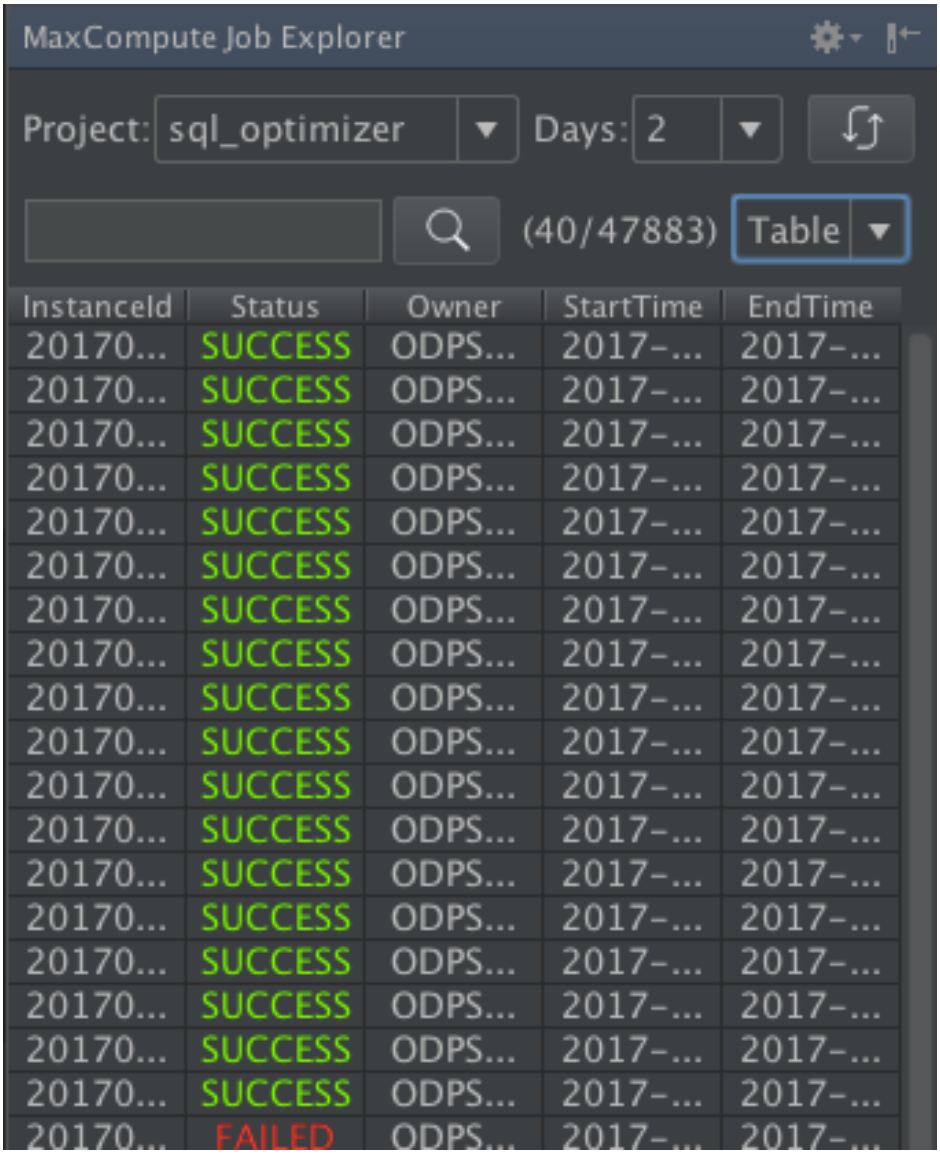
Click the  icon on the toolbar of the SQL editor to submit an SQL script to the queue of the project specified by MaxCompute.

View history jobs

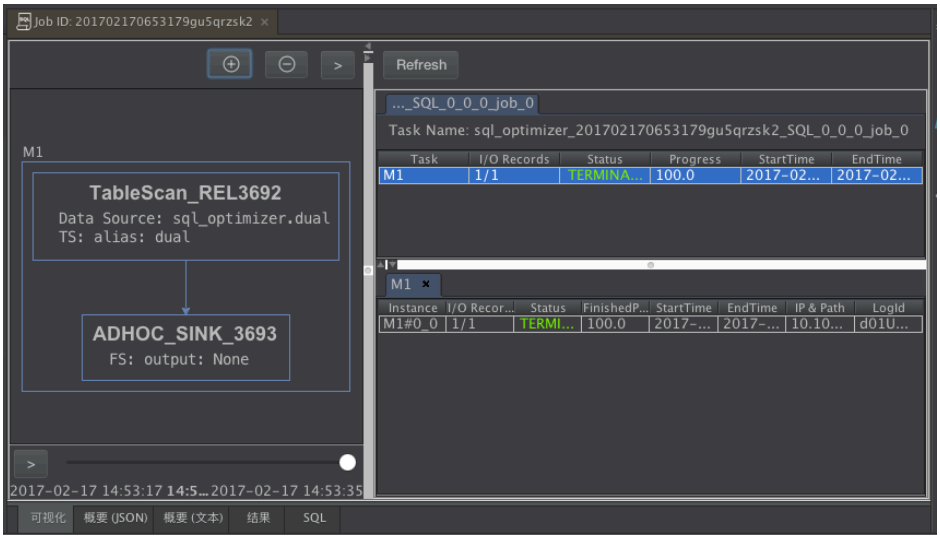
Open **Job Explorer** to view recently executed jobs in the specified project.

NOTE:

List only displays jobs submitted by the user ID of the current connection.



Double-click a job to view the job details, as shown in the following figure.



If you have the Log view URL of a job, you can select **MaxCompute > Open Log view** from the menu to go to the details page of the job.

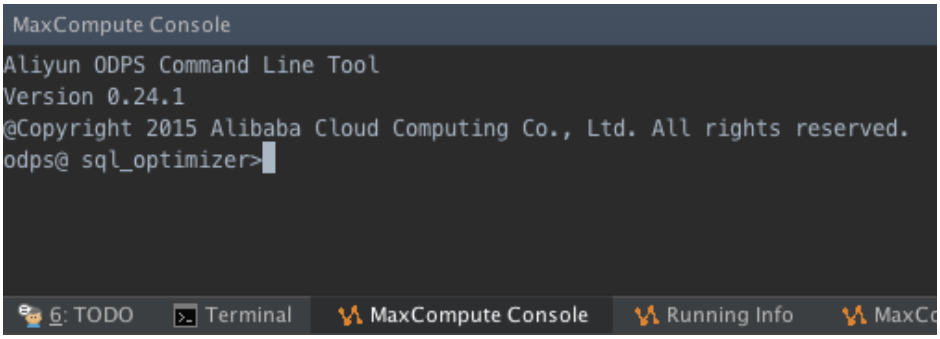
Develop a MapReduce program and UDF program

MaxCompute Studio also allows you to develop MapReduce and Java UDF programs.

Connect to a MaxCompute client

MaxCompute Studio is integrated with the MaxCompute Client of the latest version. Alternatively, you can specify the path of the locally installed MaxCompute client on the Configuration page of MaxCompute Studio.

On the **Project Explorer**, right-click a project and select **Open Console** to open the **MaxCompute Console** window.



Next step

Now, you know the functional interfaces and common application scenarios of MaxCompute Studio. Continue to the next tutorial. In this tutorial, you will learn how to install MaxCompute Studio. For more information, see Install IntelliJ IDEA.

Tool installation and version information

Install IntelliJ IDEA

This document describes how to install the basic platform IntelliJ IDEA of MaxCompute Studio.

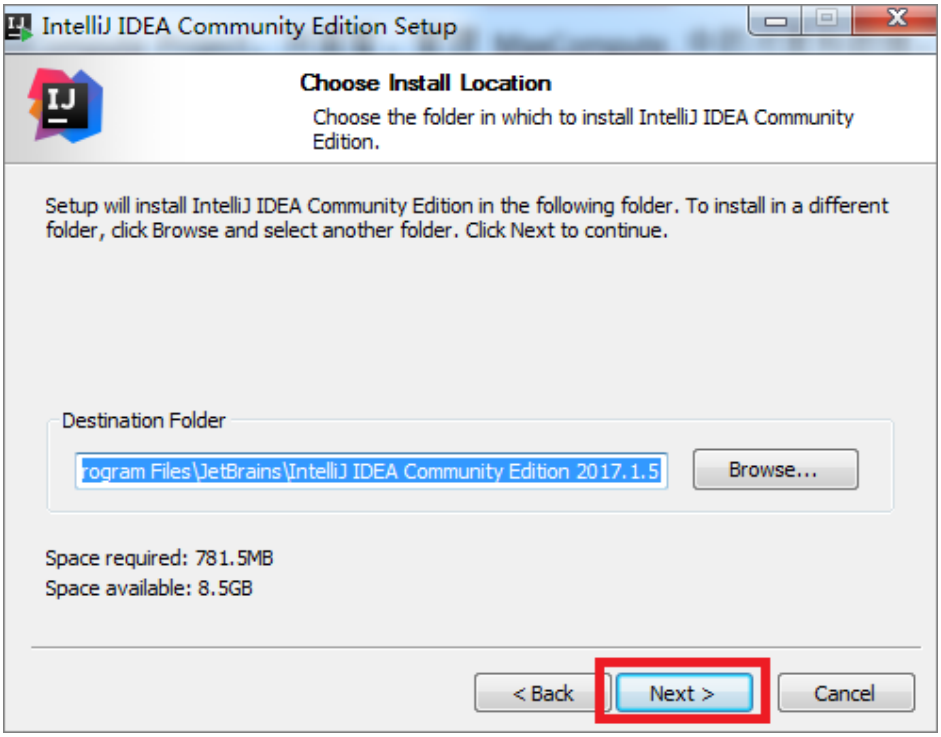
Procedure

Click here to download the IntelliJ IDEA of the version corresponding to your operating system (Windows, macOS, or Linux). The following assumes that the Windows operating system is used.

> NOTE:
>
> Download IntelliJ IDEA 14.1.4 or a later version. (The Ultimate version, PyCharm version, and free Community version are supported.)

After the download is complete, double-click the installation program to enter the installation page, and click **Next**, as shown in the following figure.

Specify the installation directory, and click **Next**, as shown in the following figure.



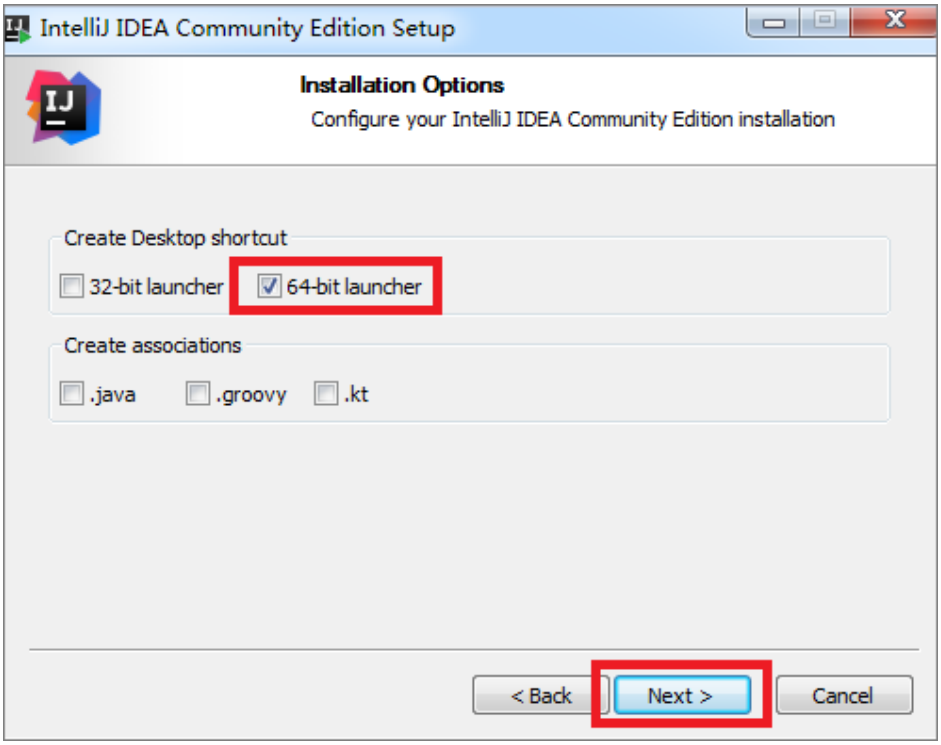
Select the 32-bit or 64-bit IntelliJ IDEA based on the version of the local operating system.

NOTE:

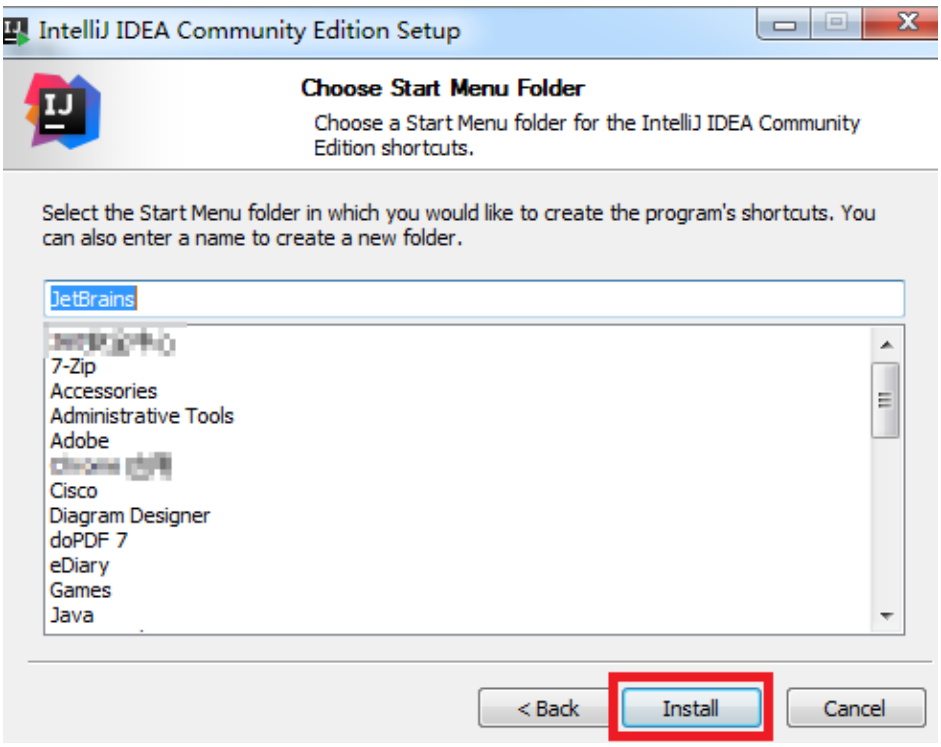
You can query the local operating system version by following these steps:

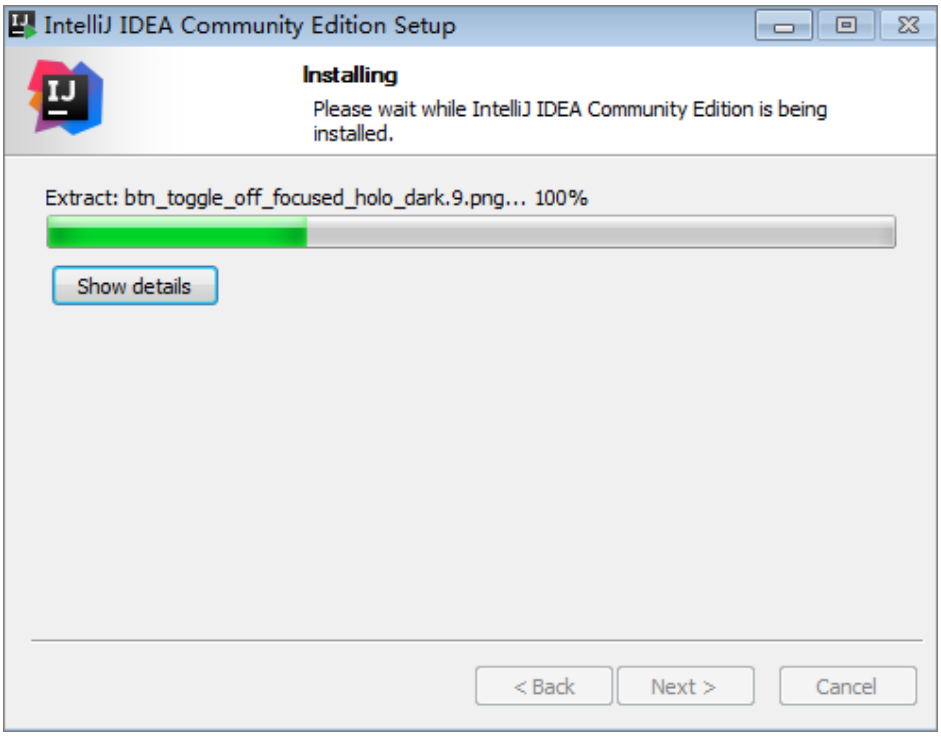
i. Open Windows Resource Manager, right-click **Computer** and select **Properties** from the shortcut menu.
ii. In the displayed window, check the type of the operating system.

Select the corresponding system type and click **Next**, as shown in the following figure.



Click **Install** to start installation, as shown in the following figure.





After the installation is complete, click **Finish**.

Next step

Now, you know how to install IntelliJ IDEA. Continue to the next tutorial. In this tutorial, you will learn how to install the MaxCompute Studio plugin. For more information, see Install the MaxCompute Studio plugin.

Installation procedure

Environment requirements

IntelliJ IDEA can be installed on Windows, macOS, and Linux. For more information about the hardware and system environment requirements, click here. IntelliJ IDEA-based MaxCompute Studio can also be installed on clients running these operating systems.

MaxCompute Studio has the following requirements on the your environment:

- A client running Windows, macOS, or Linux.
- IntelliJ IDEA 14.1.4 or a later version is installed. (The Ultimate version, PyCharm version, and free Community version are supported.)
- JRE 1.8 is installed. (JRE 1.8 has been bound to the latest IntelliJ IDEA.)

- JDK 1.8 is installed. (Optional: JDK is required if you need to develop and debug Java UDF.)

Installation method

MaxCompute Studio is a plugin of IntelliJ IDEA, which can be installed using either of the following two methods:

- Online installation using the plugin library (recommended)
- Installation using a local file

Online installation (recommended)

The MaxCompute Studio plugin has been opened for all users on the Internet. You can install MaxCompute Studio using the official IntelliJ IDEA plugin library.

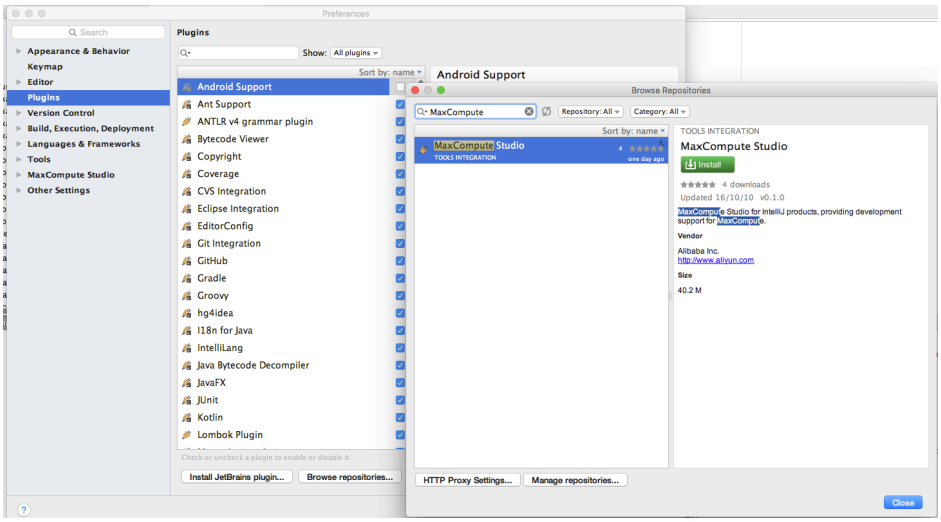
Procedure

1. Open the plugin configuration page on IntelliJ IDEA. (If you are a Windows/Linux user, choose **File > Settings > Plugins**. If you are a macOS user, choose **IntelliJ IDEA > Preferences > Plugins**.)

   Click **Browse repositories...** and search for MaxCompute Studio.

   On the MaxCompute Studio plugin page, click **Install**.

   After the installation is confirmed, restart IntelliJ IDEA to complete installation.



Local installation

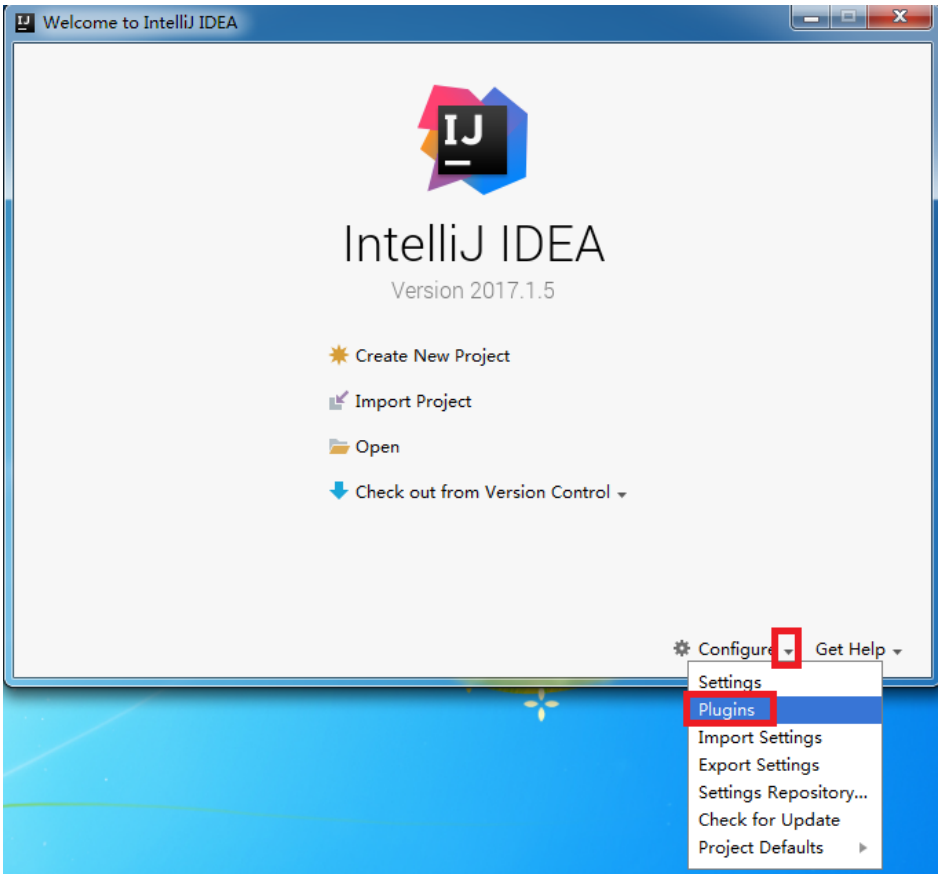
MaxCompute Studio can also be installed in a local environment.

Procedure

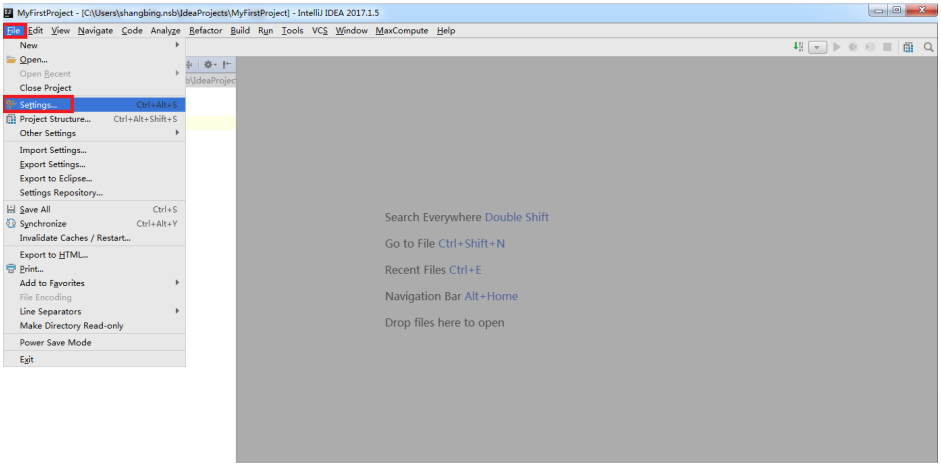
Go to the MaxCompute Studio plugin page to download the plugin package.

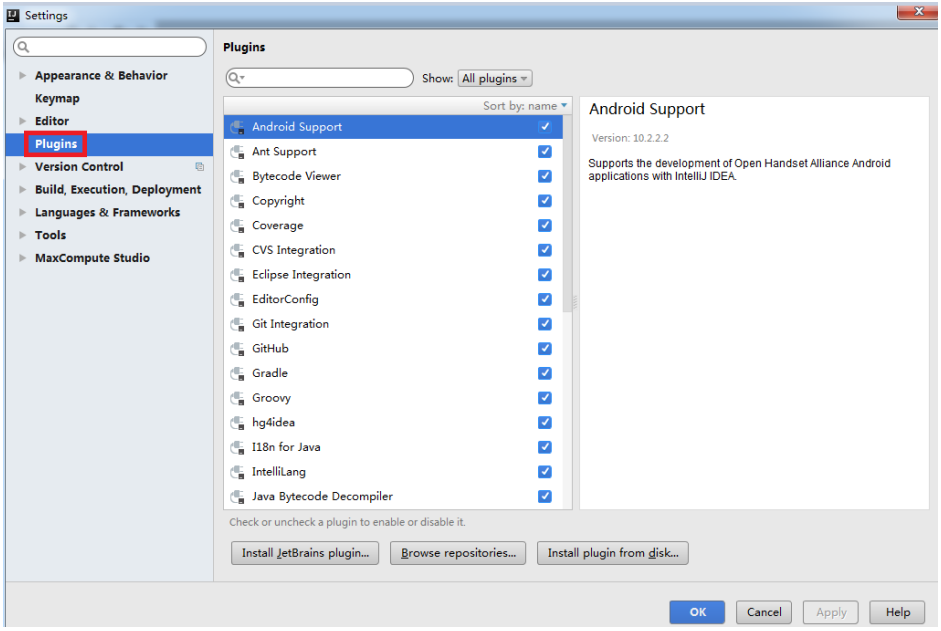
Run IntelliJ IDEA.

If you access IntelliJ IDEA for the first time, a welcome page is displayed. Click **Configure** and select **Plugins** from the shortcut menu, as shown in the following figure.

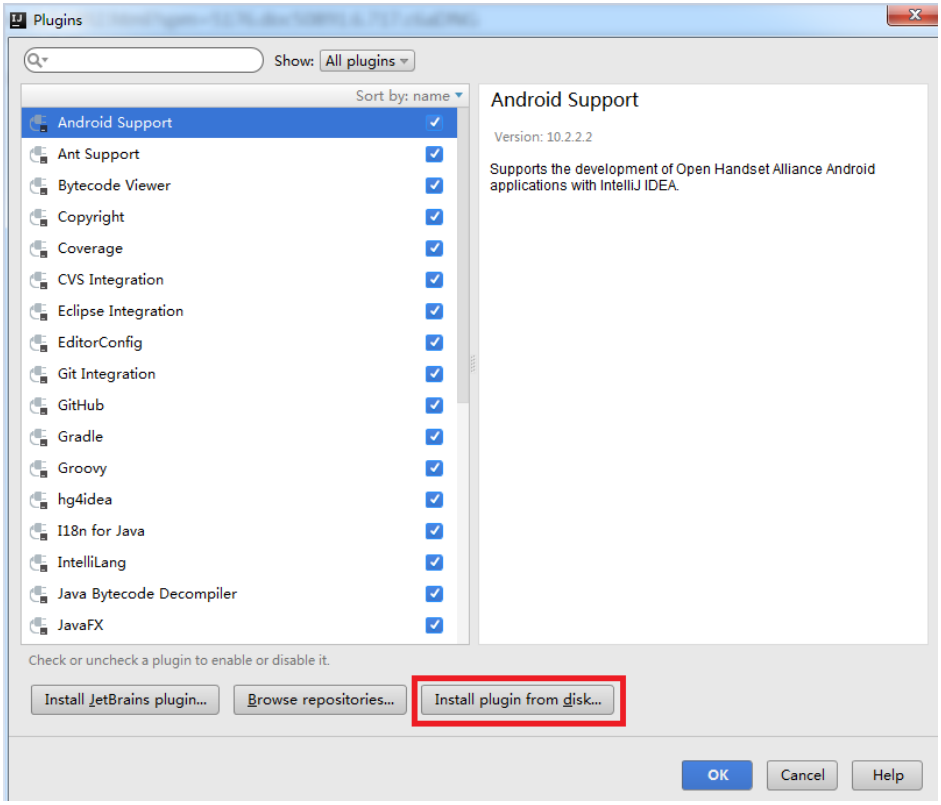


If you have accessed IntelliJ IDEA before, choose **File > Settings > Plugins** to enter the same page, as shown in the following figure.

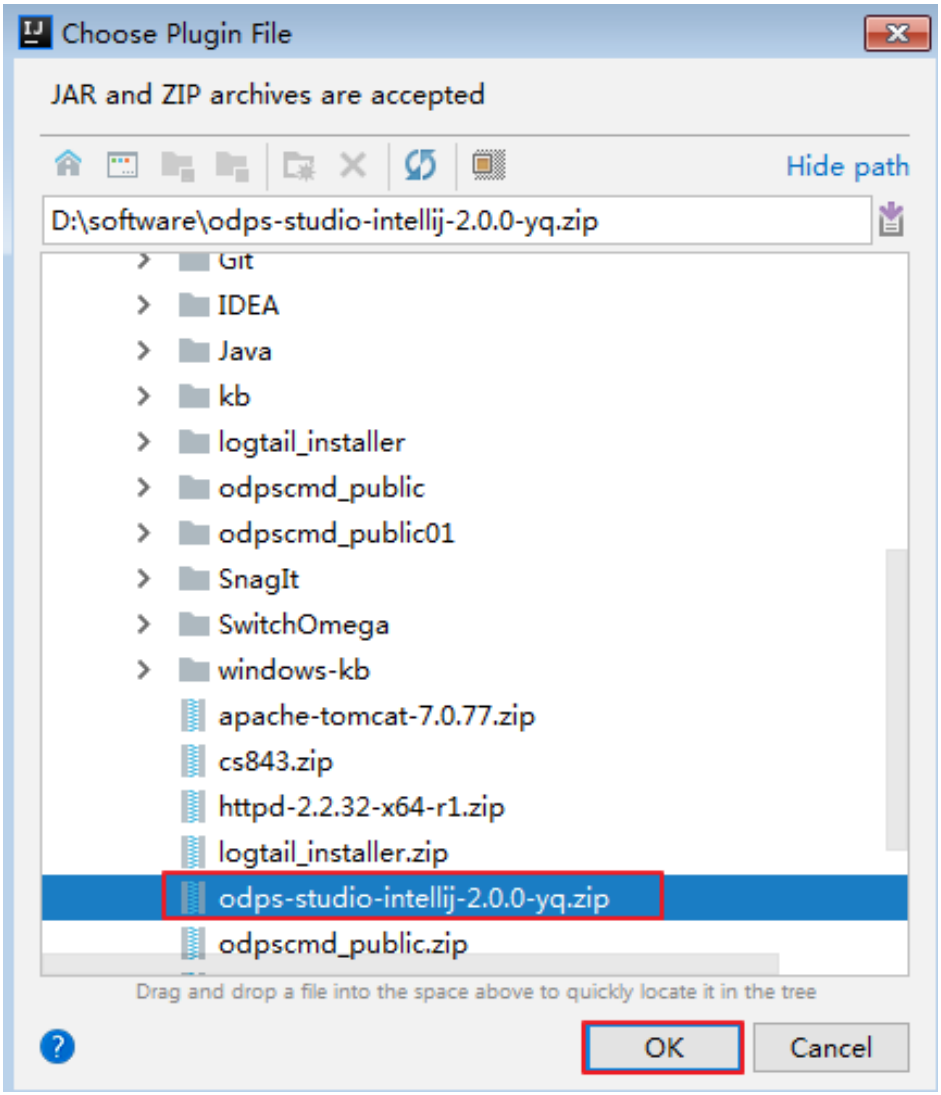




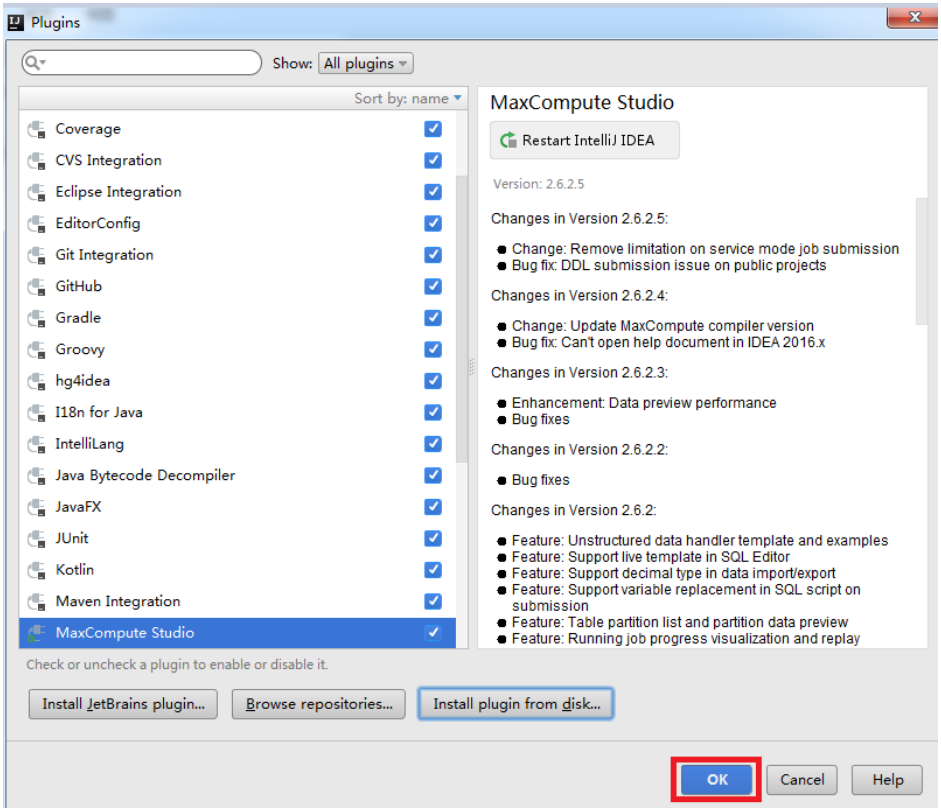
On the **Plugins** page, click **Install plugin from disk...**, as shown in the following figure.



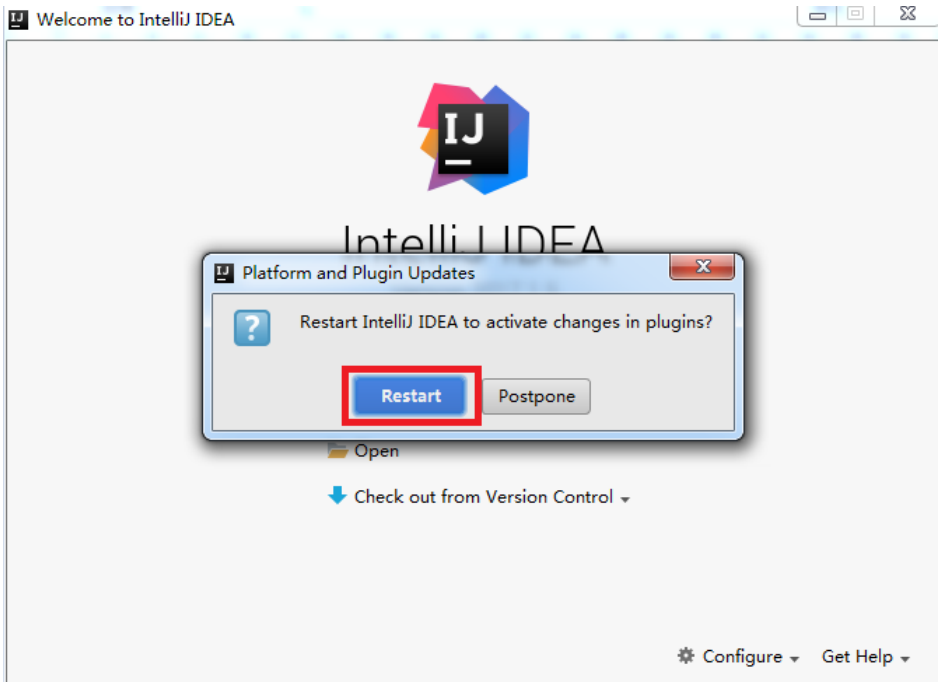
In the displayed window, click the gray icon before a directory for navigation, find the plugin file, select it, and click **OK**.



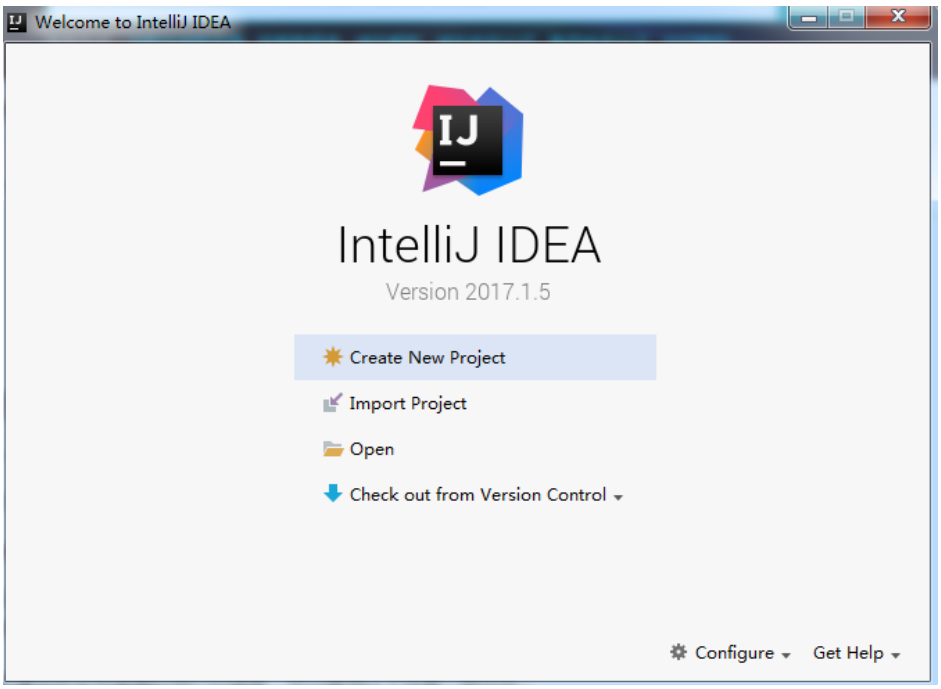
Return to the **Plugins** page and click **OK** to install the local plugin.



After the installation is complete, a dialog box is displayed, prompting you to restart IntelliJ IDEA. Click **Restart**.



After IntelliJ IDEA is restarted, the page is displayed as shown in the following figure.



Next step

Now, you know how to install the MaxCompute Studio plugin. Continue to the next tutorial. In the tutorial, you will learn how to configure a MaxCompute project connection to manage data and resources. For more information, see Create a MaxCompute project connection.

View and upgrade the version

View the MaxCompute Studio version

Follow these steps to view the MaxCompute Studio version.

1. Go to the **Settings/Preferences** page (by pressing Ctrl-Alt-S in Windows or in macOS).
2. Select **Plug-ins** on the left bar of the dialog box and search for *MaxCompute Studio*.
3. View the MaxCompute Studio version number and release information.

Alternatively, you can select **MaxCompute Studio** on the left bar of the **Settings** page to view the current version number.

Check new versions

By default, MaxCompute Studio automatically detects new versions. If a new version is available, MaxCompute Studio automatically notifies you.

After receiving an update prompt, you can select:

- **Install**: Click the **Install** link in the update prompt. The new version is automatically downloaded and installed. After the installation is complete, restart IntelliJ IDEA.
- **Configure**: Click the **Configure** link in the update prompt. You can configure whether to detect new versions automatically.

If the automatic update function is disabled, follow these steps to check and install a new version of MaxCompute Studio.

1. Go to the **Settings/Preferences** page (by pressing Ctrl-Alt-S in Windows or in macOS).
2. Select **MaxCompute Studio** on the left bar of the dialog box.
3. On the MaxCompute Studio configuration page, click **Check new versions**.
4. If a new available version is detected, MaxCompute Studio notifies you of the new version number. Click **Install new version** and restart IntelliJ IDEA to complete installation.

You can use the **Automatically checks for new versions** check box to control the switch for automatic version update check.

Next step

- Create a MaxCompute project connection

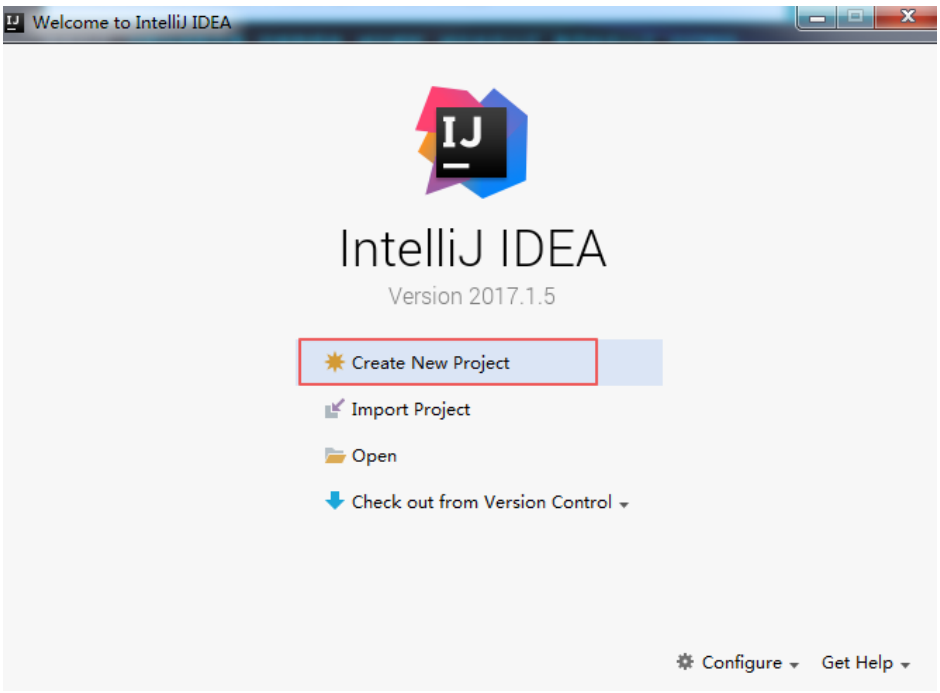
Project space connection management

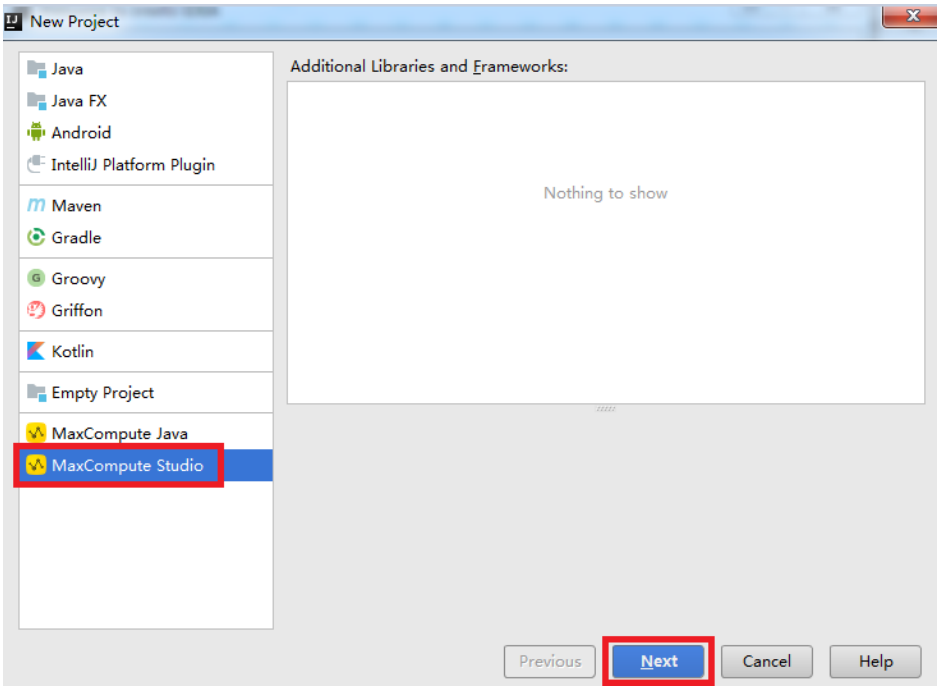
One of the core features of MaxCompute Studio is to browse resources of a MaxCompute project, including **Table**, **UDF**, and **Resource**. To realize this feature, create a project connection first.

Initial steps

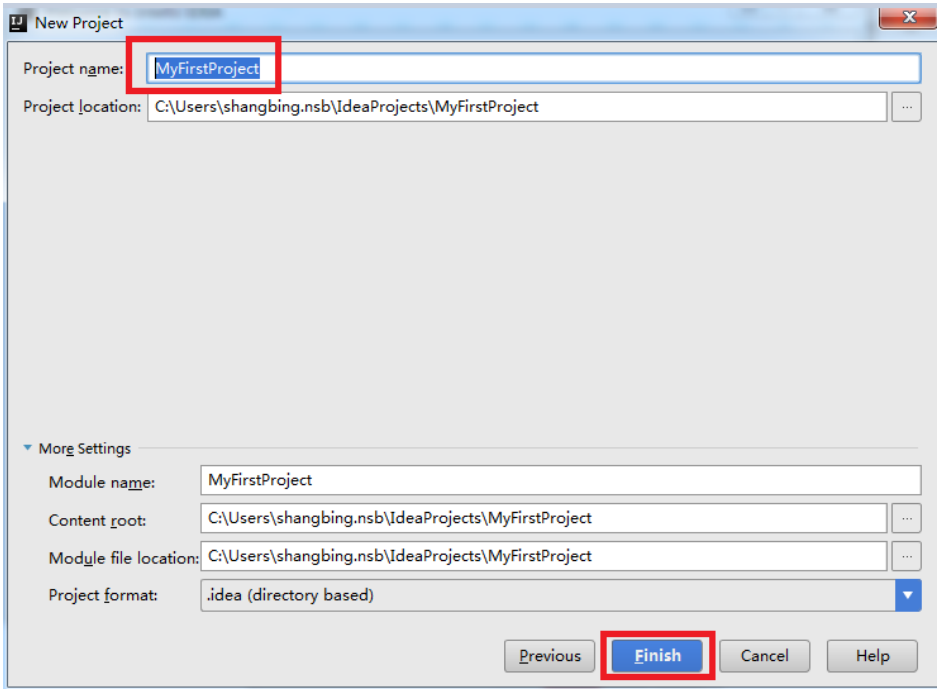
To display **Tool Windows** of IntelliJ IDEA, you must open an IntelliJ IDEA project, and the MaxCompute project must be configured on IntelliJ IDEA using **MaxCompute Project Explorer** in **Tool Windows**. Therefore, before creating a MaxCompute project connection, add or import an IntelliJ IDEA project. This document uses adding a project under Windows as an example.

Open IntelliJ IDEA, click **Create New Project**, select **MaxCompute Studio** on the displayed page, and click **Next**.





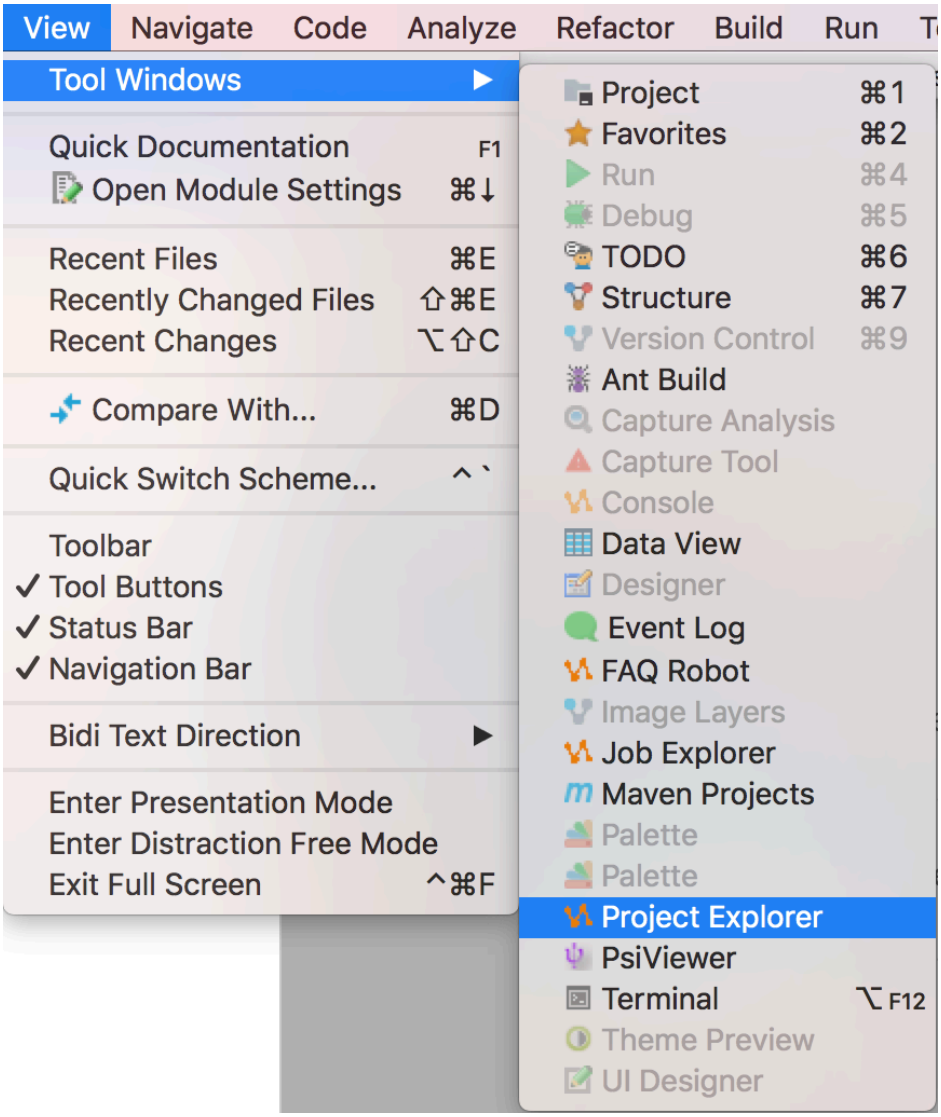
Enter the project name, and click **Finish**.



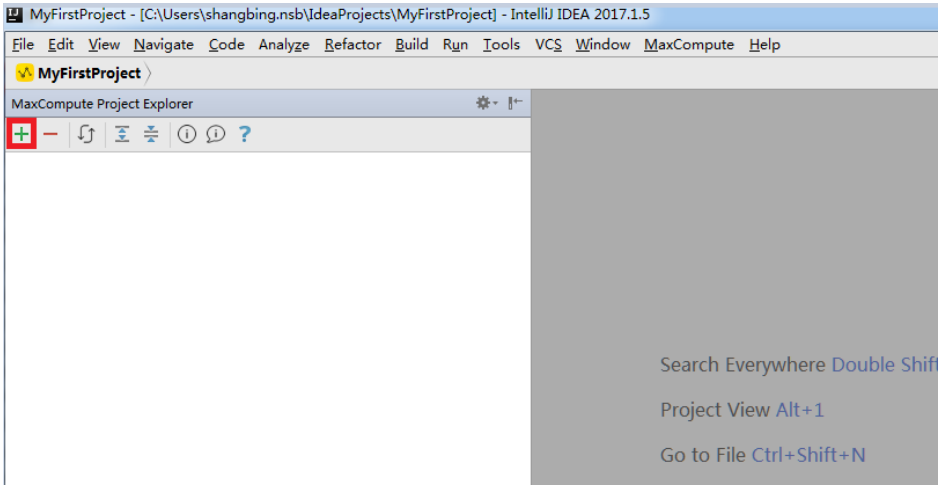
Create a MaxCompute project connection

Procedure

Select **View** > **Tool Windows** > **MaxCompute Project Explorer**.



Click plus sign (+) at the upper left corner to add a MaxCompute project.

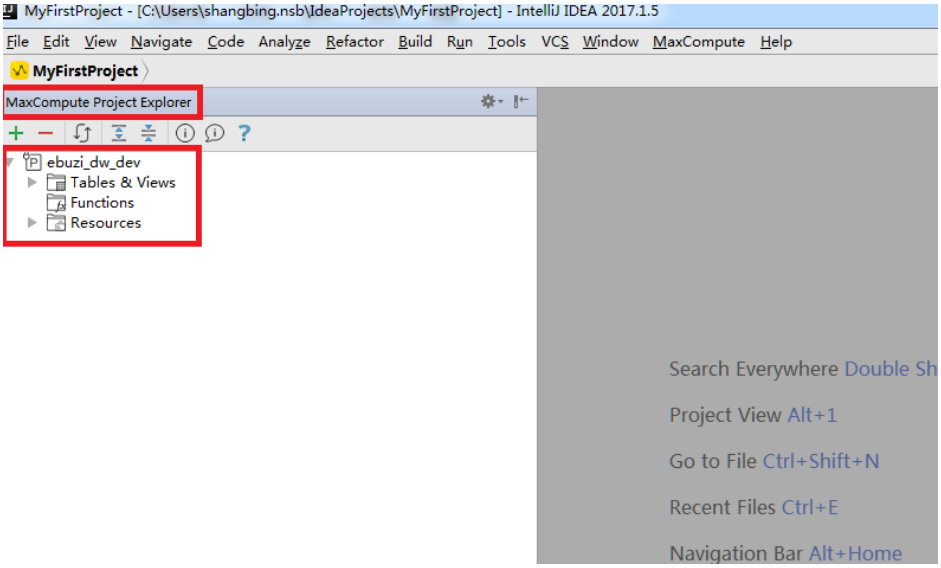


In the **Add MaxCompute Project** dialog box, set configuration options.

NOTE:

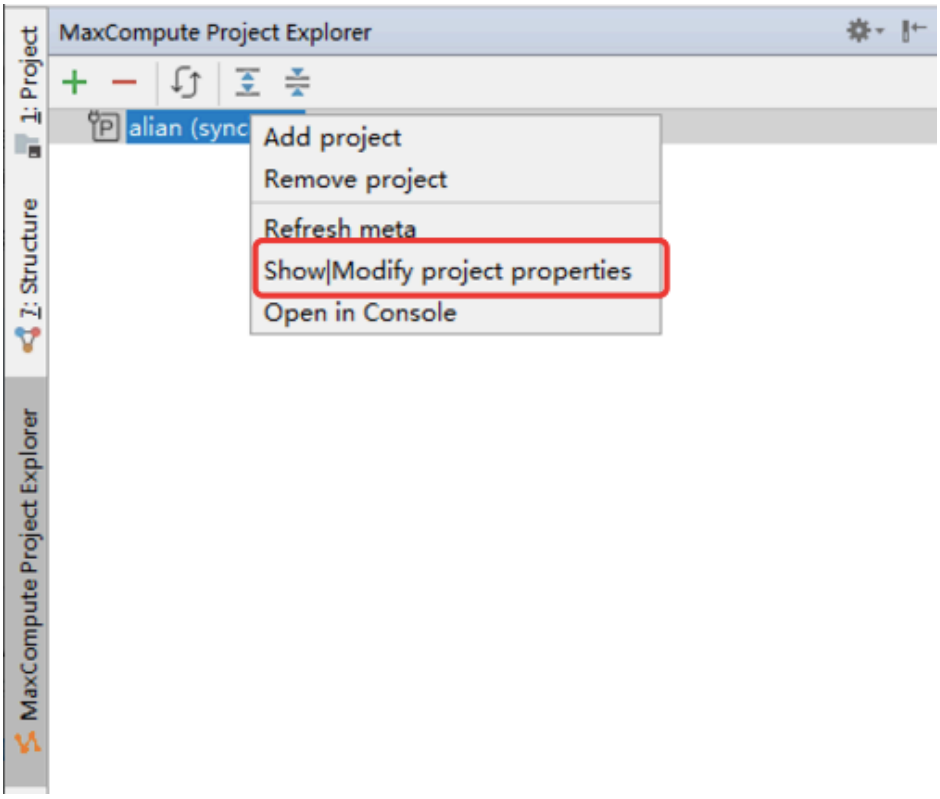
Click question mark (?) at the lower left corner of the dialog box to go to the online document page. If the synchronization times out, you can consider increasing the time-out duration for synchronizing metadata to the local host on the **Setting** tab.

After the preceding settings, click **OK**. Information about the MaxCompute project is displayed on the left of **MaxCompute Project Explorer**. You can click **Tables & Views**, **Functions**, and **Resources** to view tables, views, functions, and resources of the project.



View and modify a MaxCompute connection

In **MaxCompute Project Explorer**, right-click a MaxCompupte project and select **Show** > **Modify Project Properties**. In the displayed dialog box, you can view or modify connections and settings of the MaxCompute project.



Subsequent operations

Now, you know how to create and manage a project connection. You can continue to the next tutorial. In the tutorial, you will learn how to query metadata, clear data, and upload and download data to manage data and resources. For more information, see Manage data and resources.

Manage data and resources

View tables and UDF

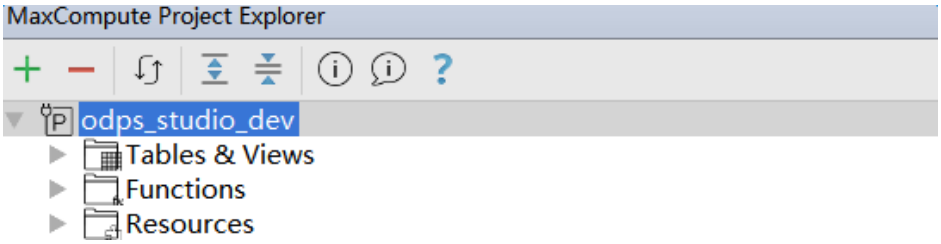
View tables and functionsView tables and functions

In the **Project Explorer** window, you can view tables, functions, and resources with connections added. For tables and functions to be viewed in the **Project Explorer** window, the MaxCompute project connections must be added, for more information, see Add MaxCompute project connections.

Browse tables and functions

To browse tables and functions in the project space, follow these steps.

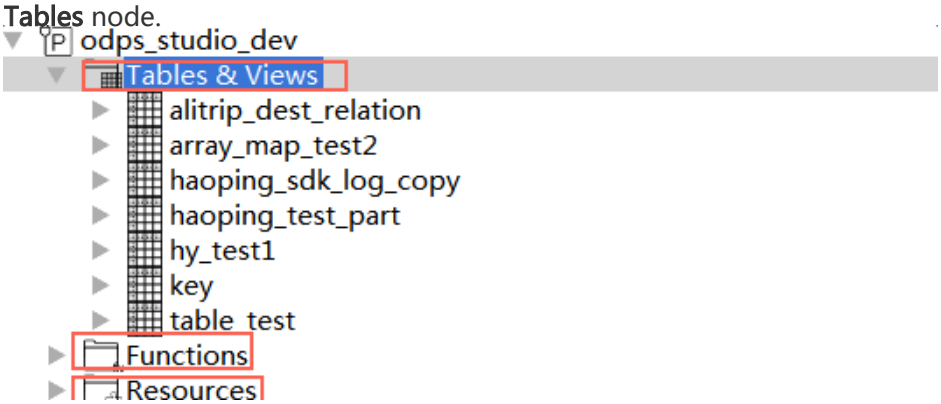
Open the **Project Explorer** window and you can view the added **Project** node tree.



The toolbar is displayed at the top of the node tree, and includes:

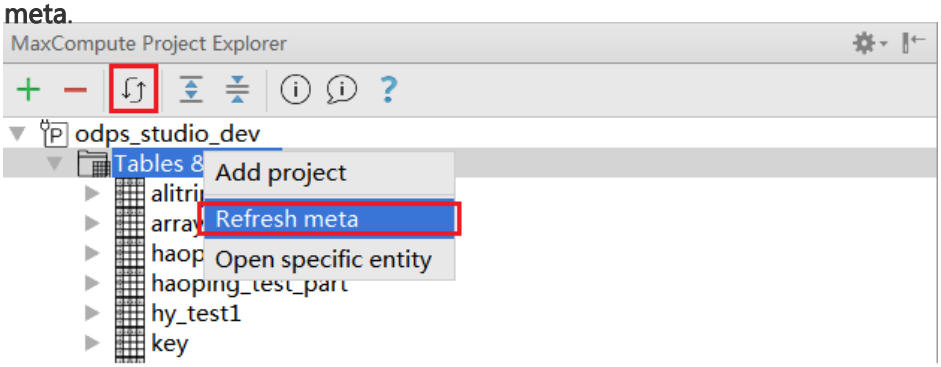
- **Add Project**: Adds a connection to the MaxCompute project space.
- **Delete Project**: Deletes a connection from **Project Explorer**, which has no impact on the project space on the server end.
- **Update Metadata**: Updates metadata information from the project space on the server end and updates the locally buffered metadata.
- **Expand Node**: Expands all tree nodes.
- **Fold Node**: Folds all tree nodes.
- **User Feedback**: Submits user feedback.
- **Online Documentation**: Opens online documents.

Double-click the **Tables** node or click the drop-down arrow to expand the **Tables** node to list all tables in the project (including virtual views). The table name list serves the same purpose as the **show tables** command. You must have the List Table permission in the project. The methods for the **Functions** and **Resources** nodes are similar to that of the **Tables** node.



MaxCompute Studio downloads project metadata on the server to the local device. When metadata on the server end is updated, for example, a new table is added, you must manually trigger a refresh to reload changed metadata to the local device. The refresh can be performed at the **Project** or **Table** level. The procedure is as follows:

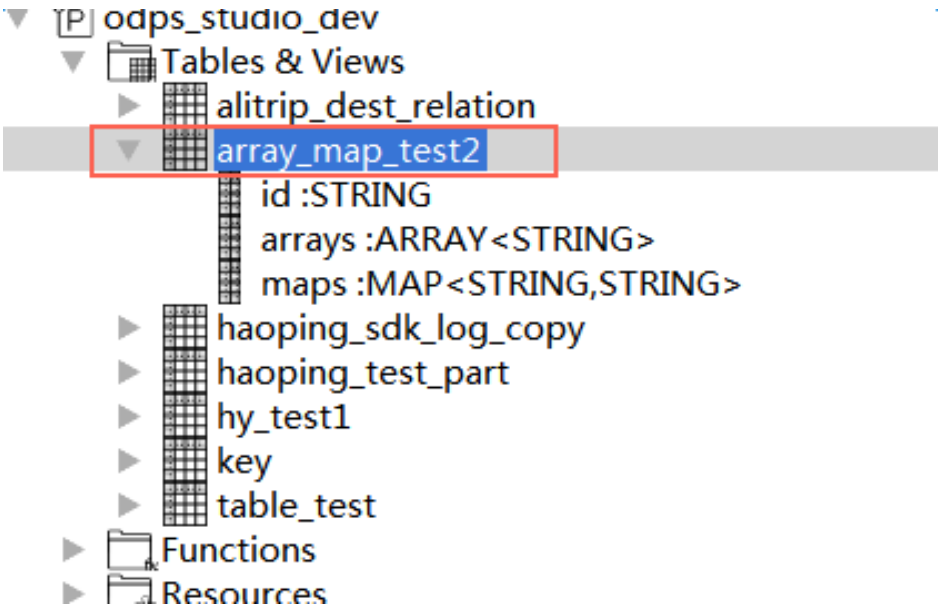
i. Select a node.
ii. Click the Refresh icon on the toolbar or right-click the node and select **Refresh meta**.



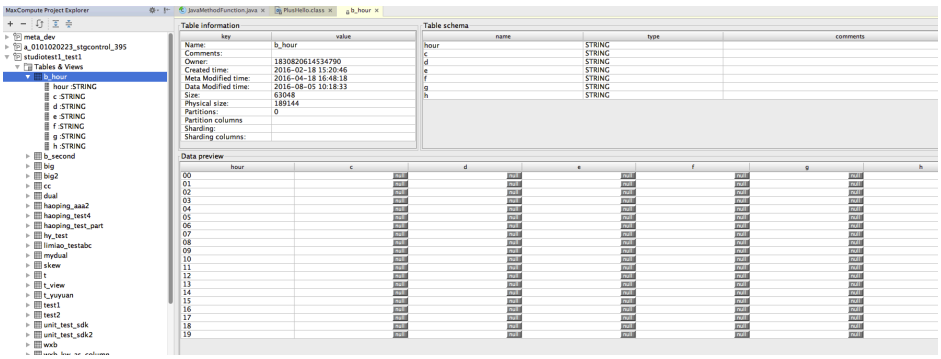
View table details

You can view data table information in **Table Details View** of MaxCompute Studio.

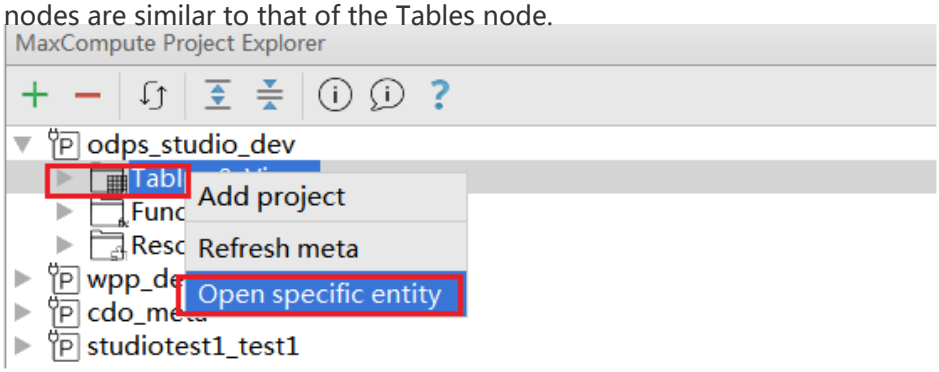
In the node tree, expand a table node to view the column name and type.

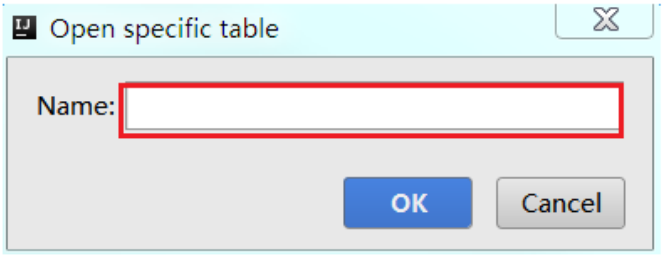


Double-click a table or right-click a table and choose **Show Table Detail** to view the table details. The table details include metadata, such as owner, size, and column, table structure information, and data preview.



Right-click **Tables & Views** and select **Open specific entity** to display the details of the specific table. Note that the complete table name must be specified. If you do not have the List permission on the project and only have the permission on a specific table, you can also view details of the table using this method. The methods for the Functions and Resources nodes are similar to that of the Tables node.

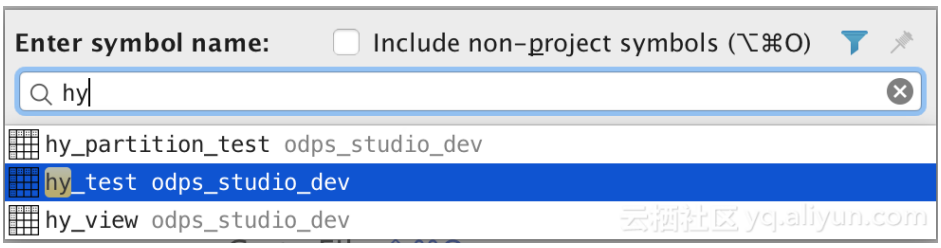




NOTE:

Intellij IDEA supports searching by default. After a table is expanded, you can directly press keys on the keyboard to perform fuzzy match.

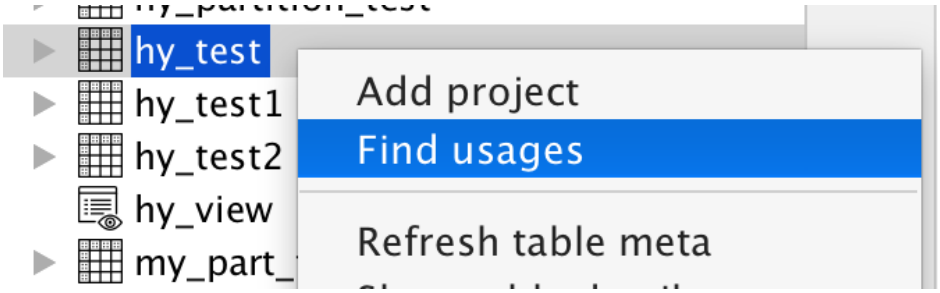
MaxCompute Studio also supports quick search for the table, you can use the shortcut key (Windows: Ctrl + Alt + Shift + N, macOS: + + O) to call the navigation bar, then enter the name of the table and press Enter.



NOTE:

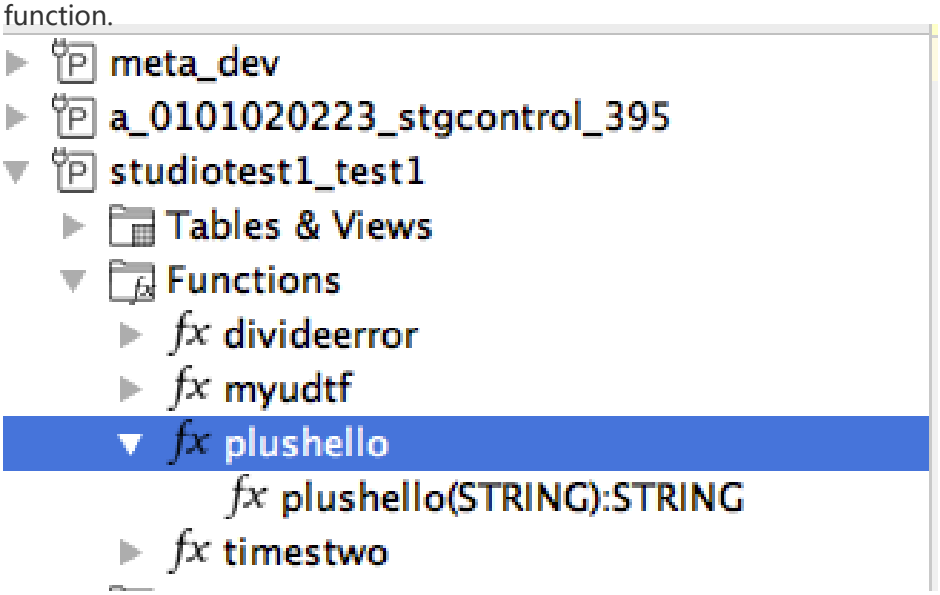
You can narrow the search by using the pre-keyword (table:, function:, or resource:). For example, to search for the function count, enter function:count.

To know the scripts in which the table is used, right-click the table and select **Find usages**.



View function details

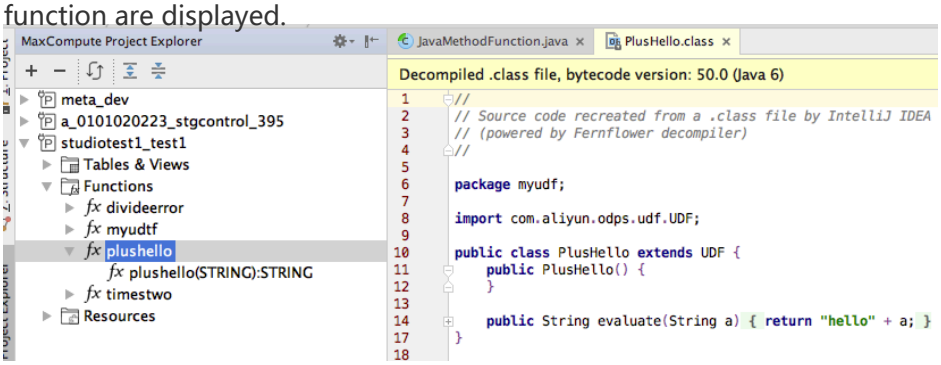
Expand a function node under the **Functions** node to display the method signature of this function.



NOTE:

To enable the Python UDF to parse the signature, install PyODPS (MaxCompute Python SDK) first. Install pip: sudo/usr/bin/python get-pip.py (Download get-pip.py from Google manually) and then PyODPS: sudo/usr/bin/python -m pip install PyODPS. Note that the Mac operating system has Python, which is stored in /usr/bin/python. Install PyODPS in this directory.

Double-click a function node under the **Functions** node. Alternatively, double-click the source code resource of the function under the **Resources** node. In this case, codes of this function are displayed.



NOTE:

The Java code is obtained by decompiling JAR, which is not the source code.

Import and export data

Import and export table data

MaxCompute Studio can import local data files in CSV or TSV format to MaxCompute tables and export MaxCompute table data to local files.

MaxCompute Studio completes data import and export by using Batch data tunnel provided by the MaxCompute platform.

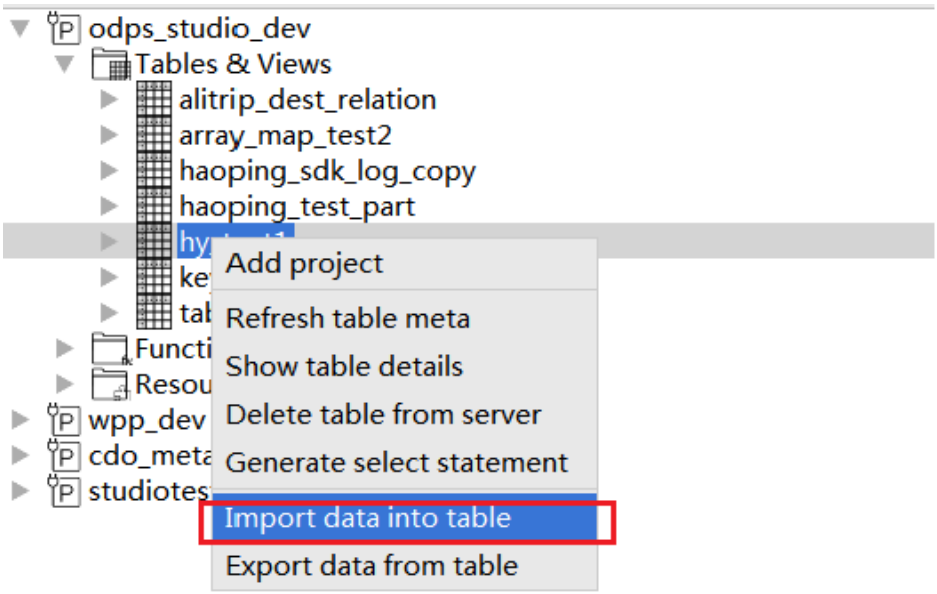
Usage instructions

The MaxCompute Tunnel service must be used for data import and export. Therefore, the MaxCompute project added in Studio must be configured with the Tunnel service.

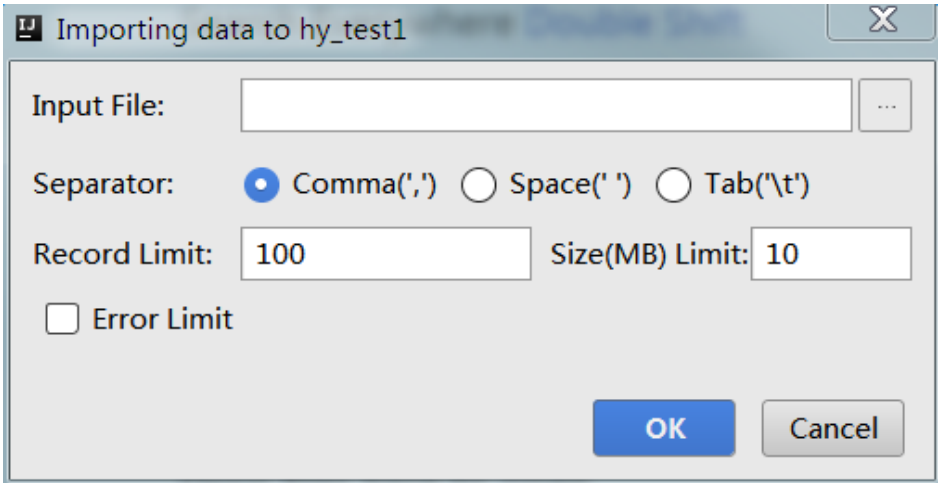
Related permissions must be granted for table import and export.

Import data

Open the **Project Explorer** window, right-click a table name or a field attribute in **Data preview** of **Table details** and select **Import data into table**.



In the **Import data** dialog box that appears, select the path of the imported data file, column separator, size limitation, and number of lines for an error tolerance, and click **OK**.

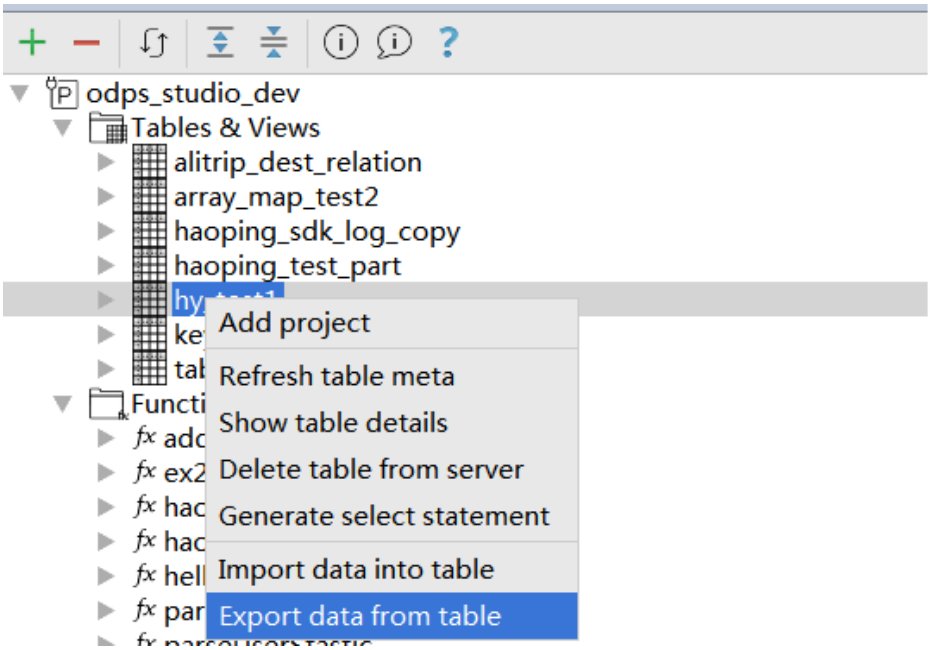


If **Import data success** is displayed, data import is successful and imported data can be viewed in the table.

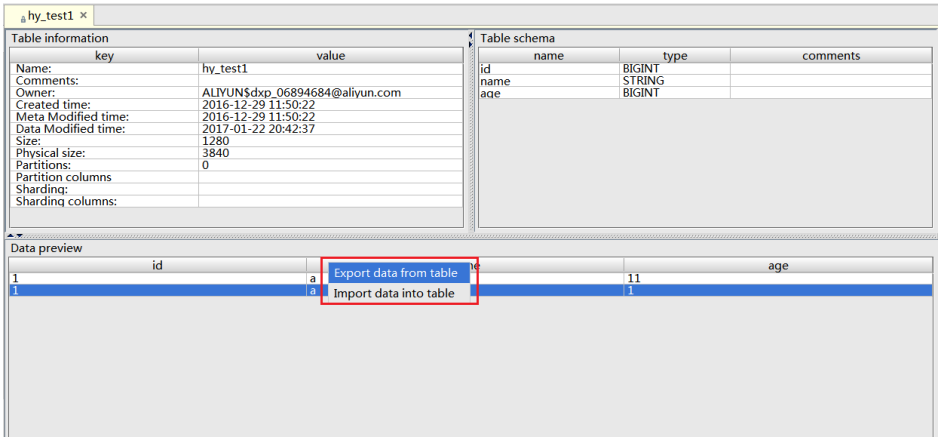
Export data

1. Two methods are provided for table data export.

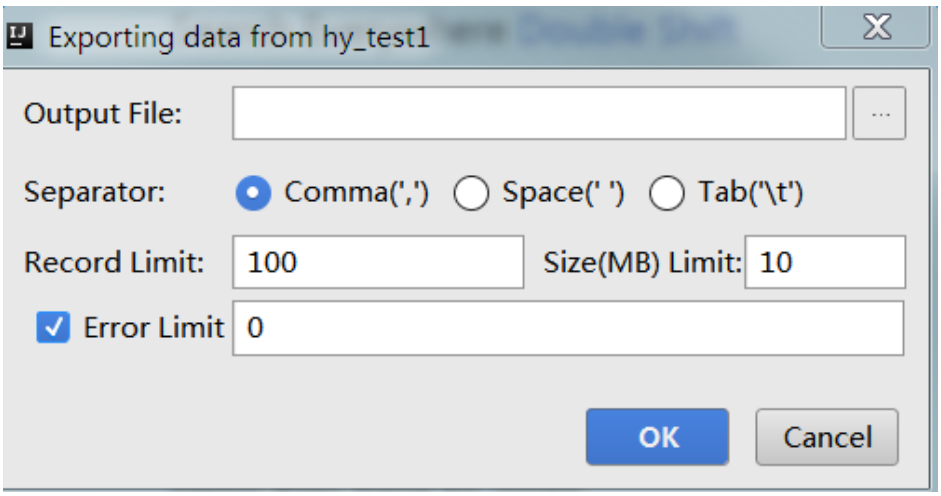
Right-click a table name and select **Export data from table**.



Right-click a field attribute in **Data preview** of **Table details** and select **Export data from table**.

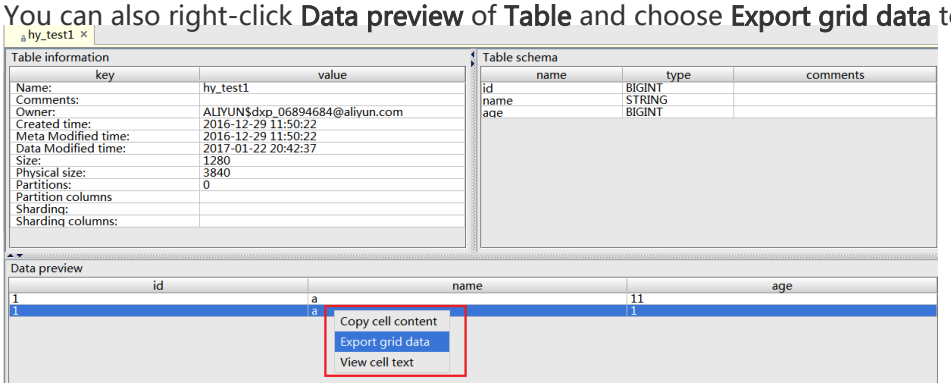


In the Export data dialog box that appears, select the path for saving the exported data file, column separator, size limitation, and number of lines for an error tolerance, and click **OK**.



If **Export data success** is displayed, data export is successful and exported data can be viewed in the target file.

You can also right-click **Data preview** of **Table** and choose **Export grid data** to export data.



NOTE:

The data export function in **Data preview** is used only to export data displayed in **Data sample** instead of all data in the table.

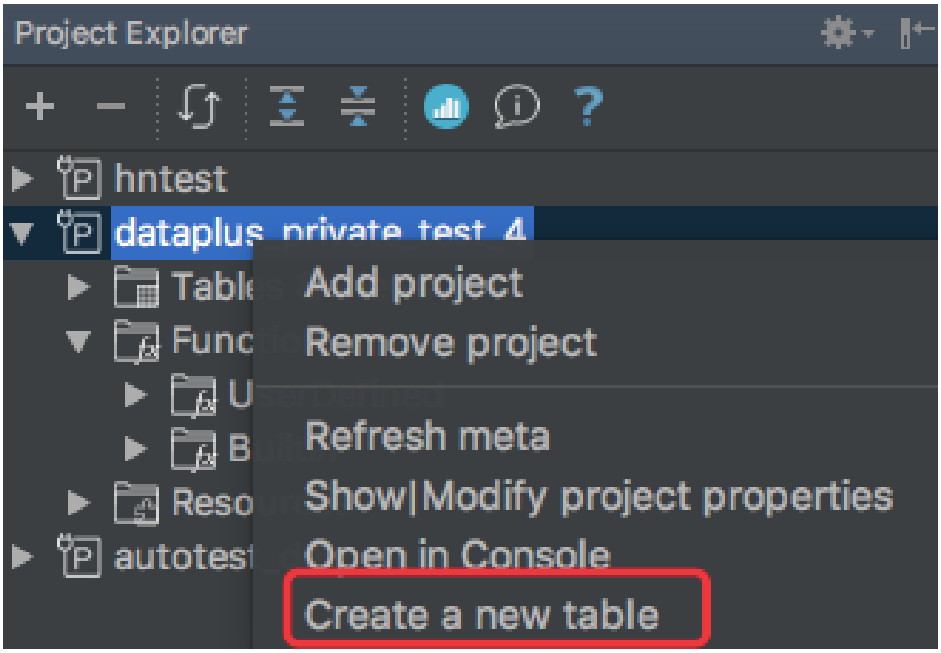
Visualizations of creating, modifying, and deleting a table

The Project Explorer of MaxCompute Studio provides the visualized table structure editor used to create and modify tables.

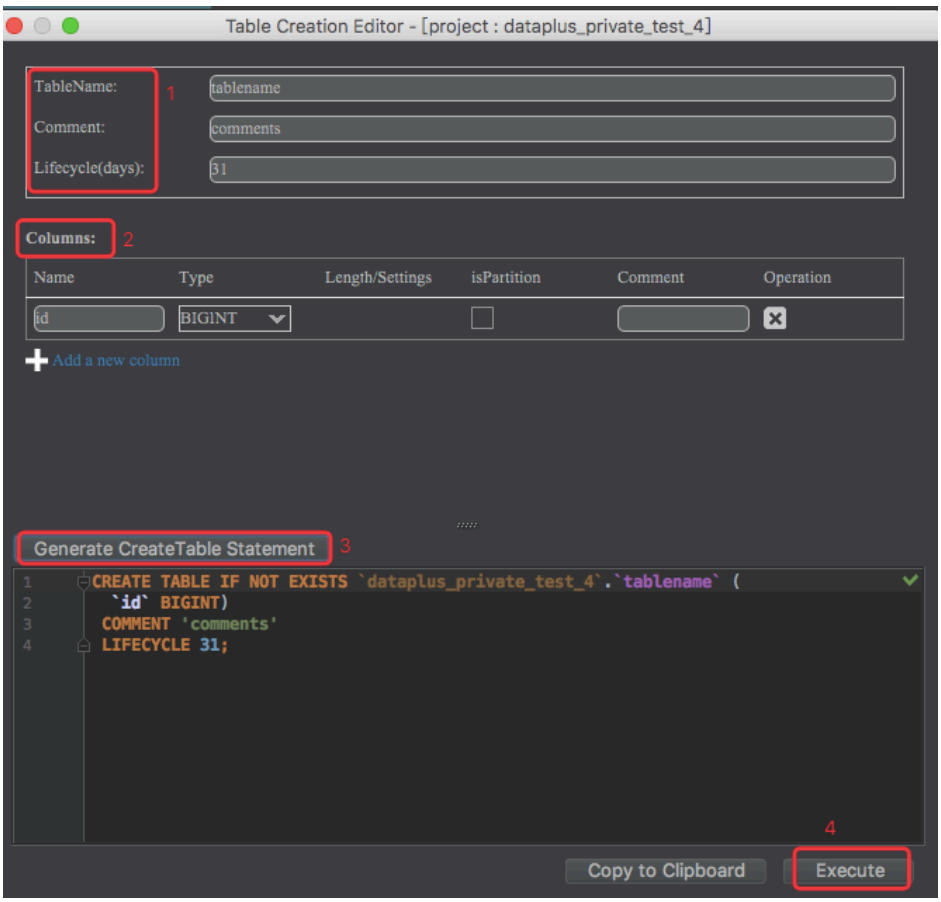
Visualization of creating a table

Procedure

Right-click the target project and select **Create a new table**.



In the dialog box that appears, enter a table name and column information. Click **Generate CreateTable Statement** to generate a DDL statement. Click **Execute** to create the table.



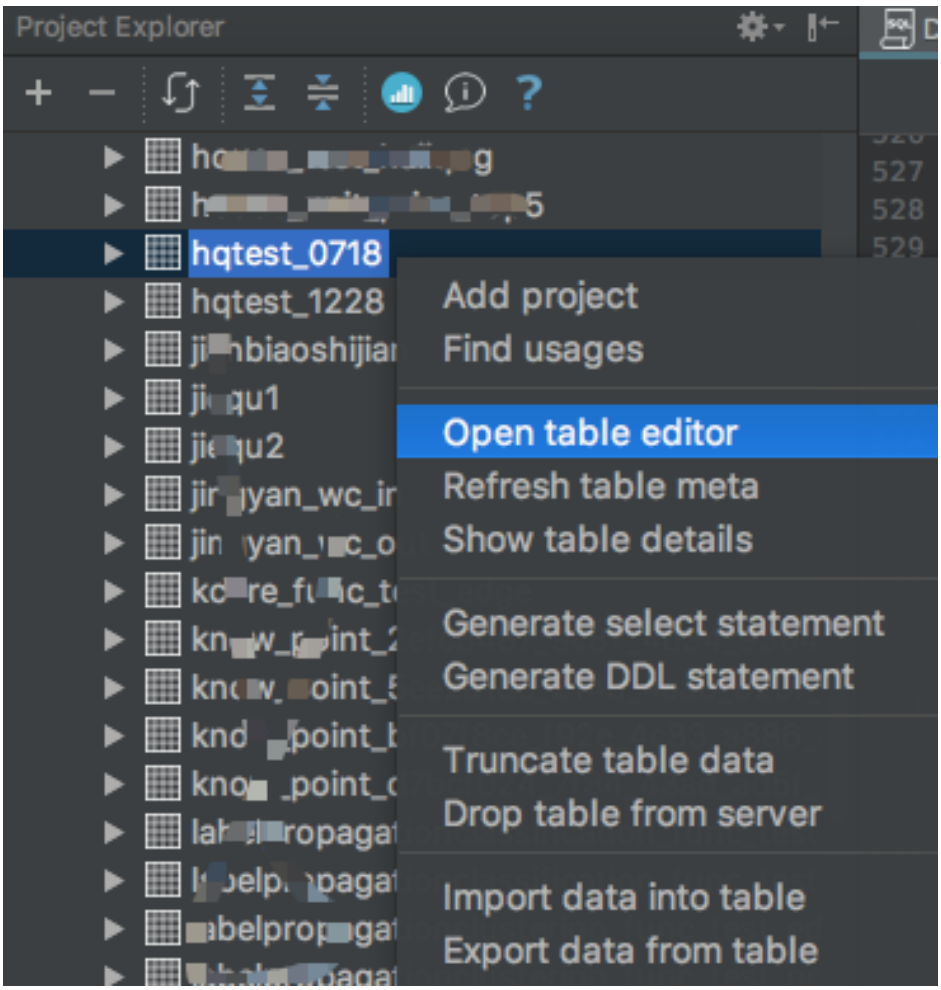
When you set the table name, column name, type, and lifecycle, observe the related requirements of MaxCompute. For more information, see the DDL documentation.

After the table is created, view the table metadata in **Tables & Views** of the Project Explorer. If no metadata is displayed, refresh the list.

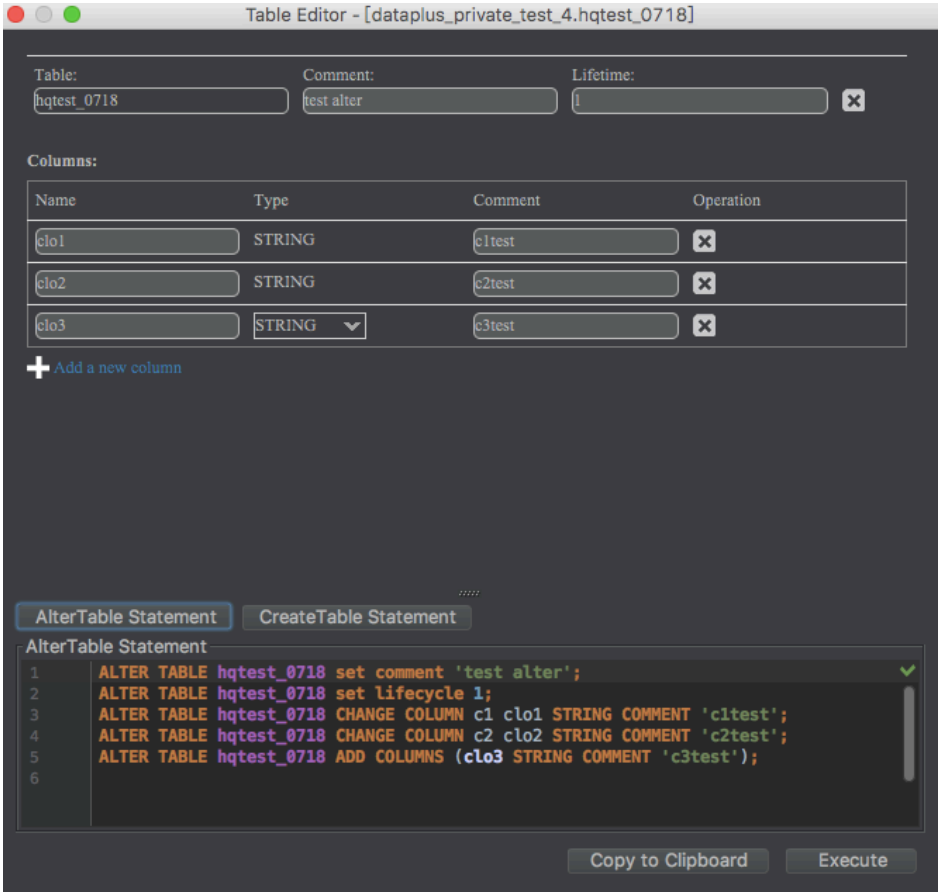
Visualization of modifying a table

Procedure

In **Tables & Views** of the Project Explorer, right-click the expected table and select **Open table editor**.



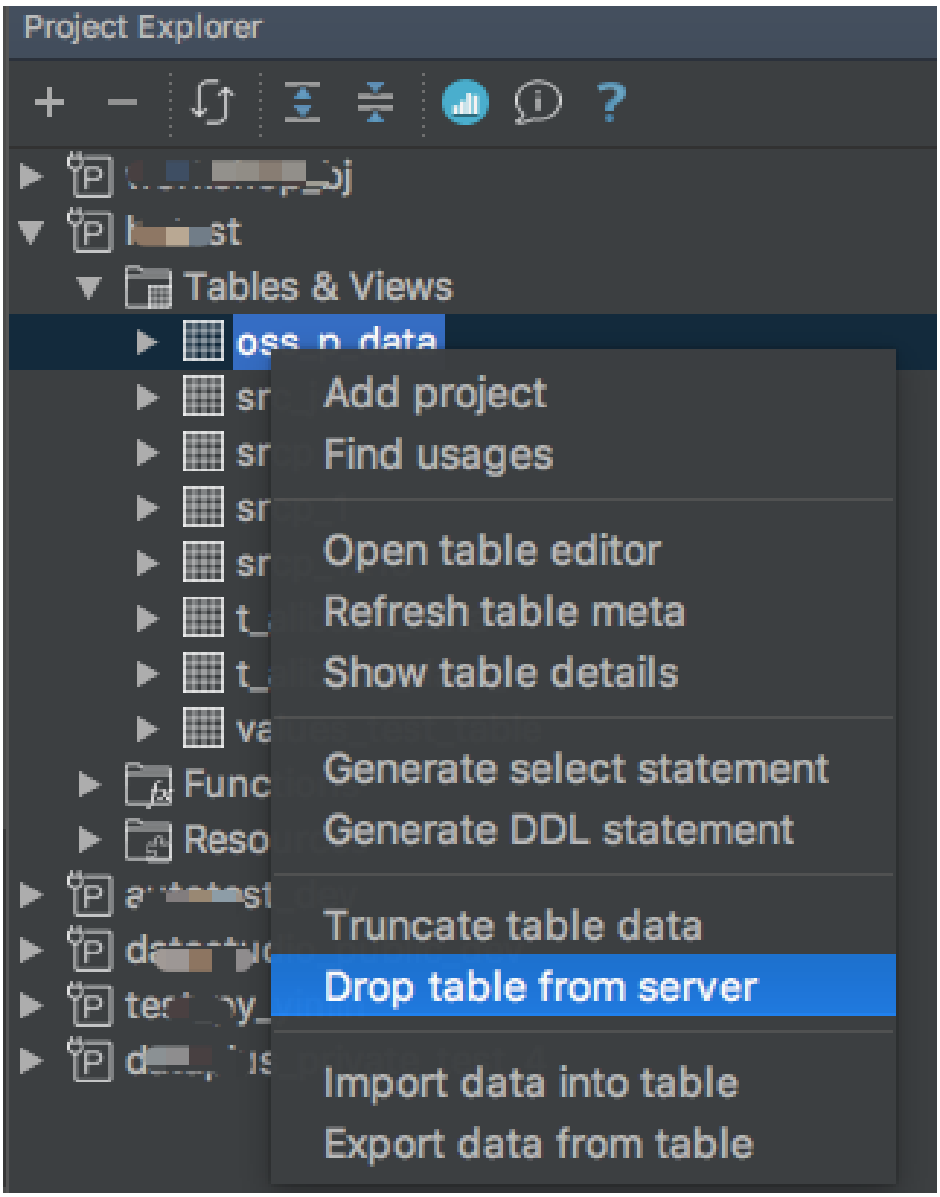
In the dialog box that appears, edit the table. You can modify the table comments, lifecycle, column name and description, and add columns. When you edit the table, observe the table-related requirements of MaxCompute. For more information, see the DDL documentation.



After you finish modification, click **Alter Table Statement** to generate an ALTER statement. Click **Execute** to apply the modification. After successful execution, view the table metadata.

Visualization of deleting a table

In **Tables & Views** of the Project Explorer, right-click the expected table and select **Drop table from server**.



In the dialog box that appears, click **OK**. Then, the table is deleted from the MaxCompute instance.

Develop SQL procedure

Create MaxCompute Script module

Before developing MaxCompute Script, you must create a MaxCompute Script module in either of

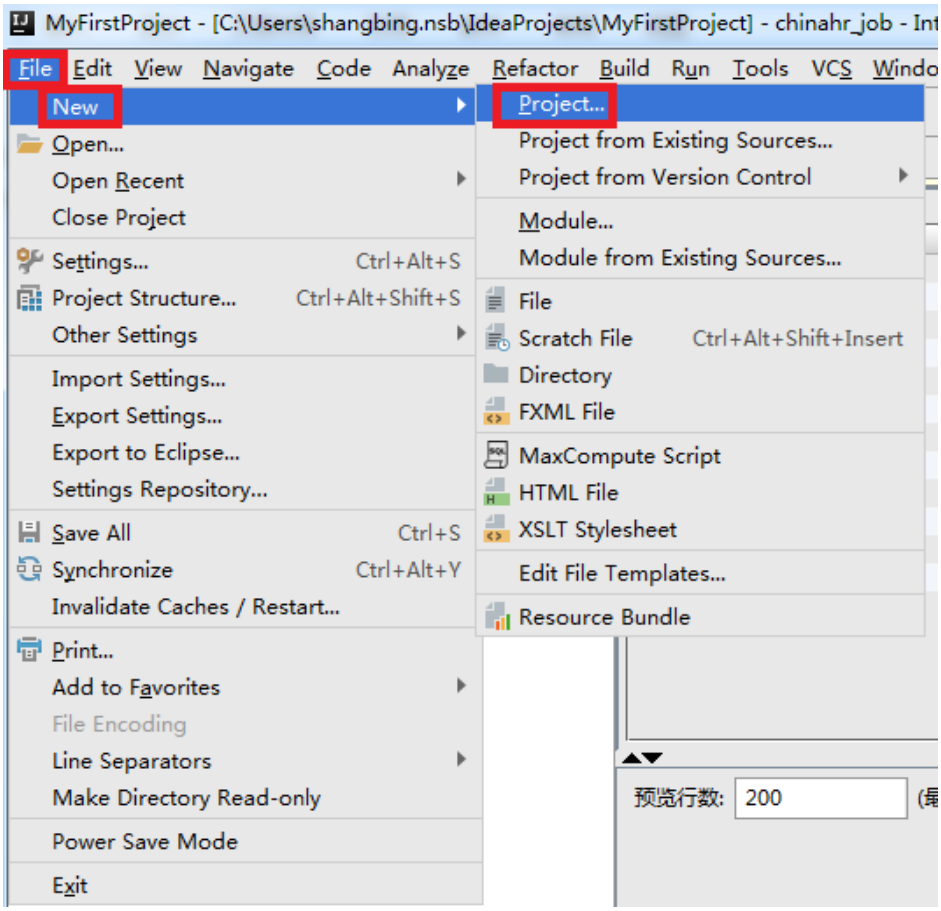
the following scenarios.

No script file exists locally

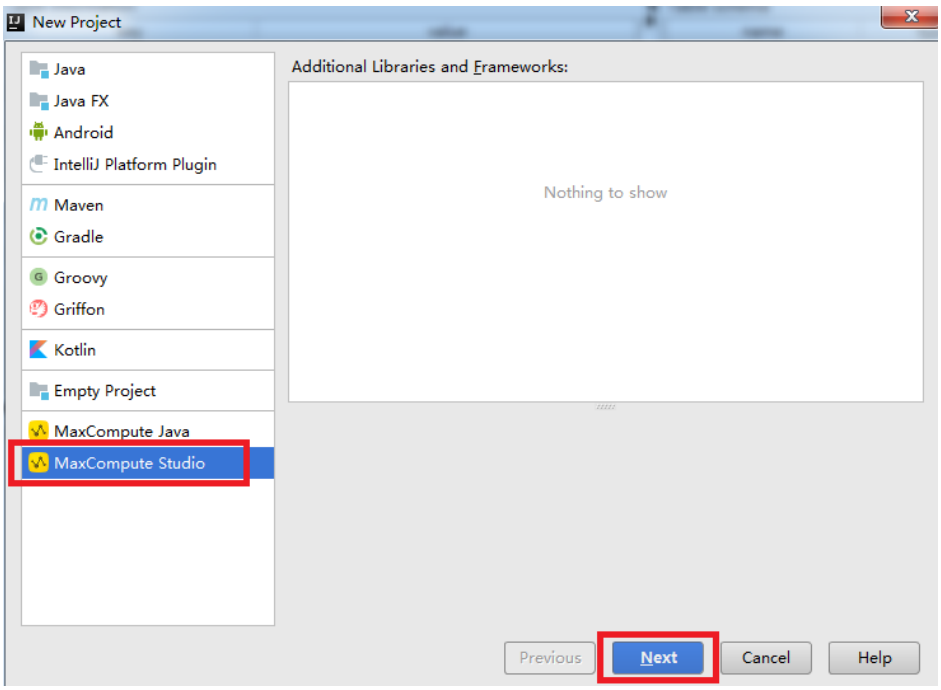
If no script file exists locally, you can use Intellij IDEA to create a new module.

Procedure

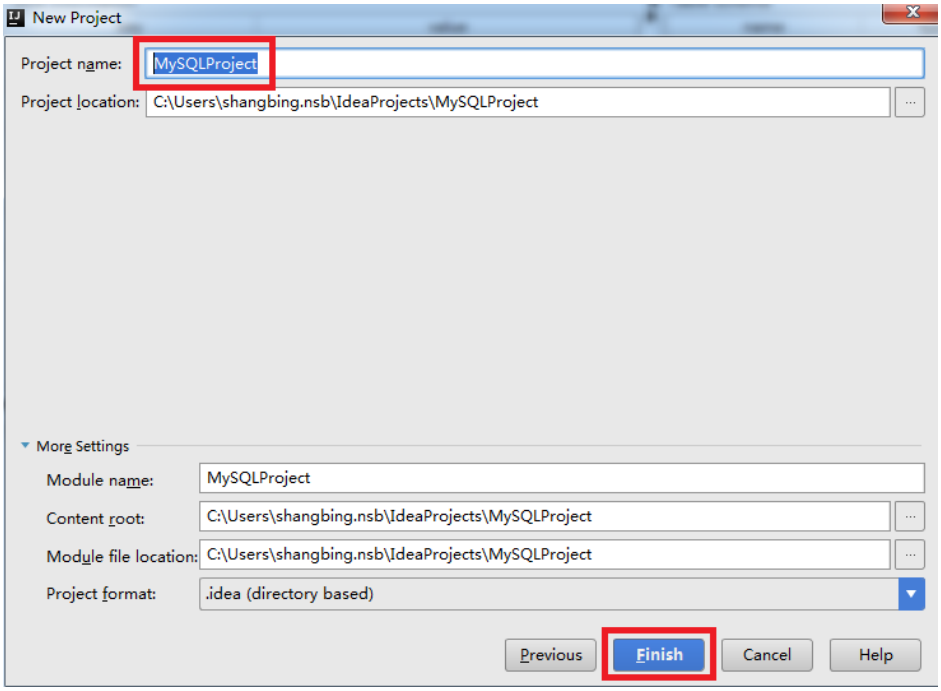
Open or create a MaxCompute Studio project. This article uses creating a project as an example. Click **File** in the menu and select **New > Project**, as shown in the following figure.



Select **MaxCompute Studio** on the left-side navigation pane, and click **Next**.

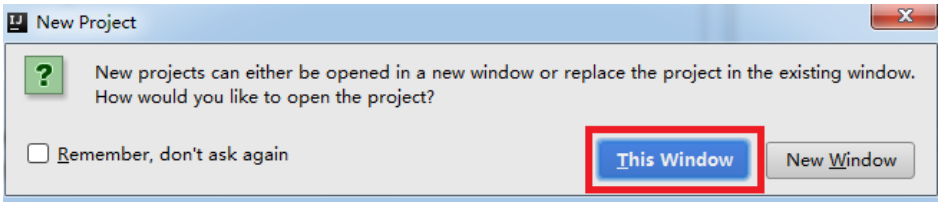


Enter the project name, and click **Finish**.

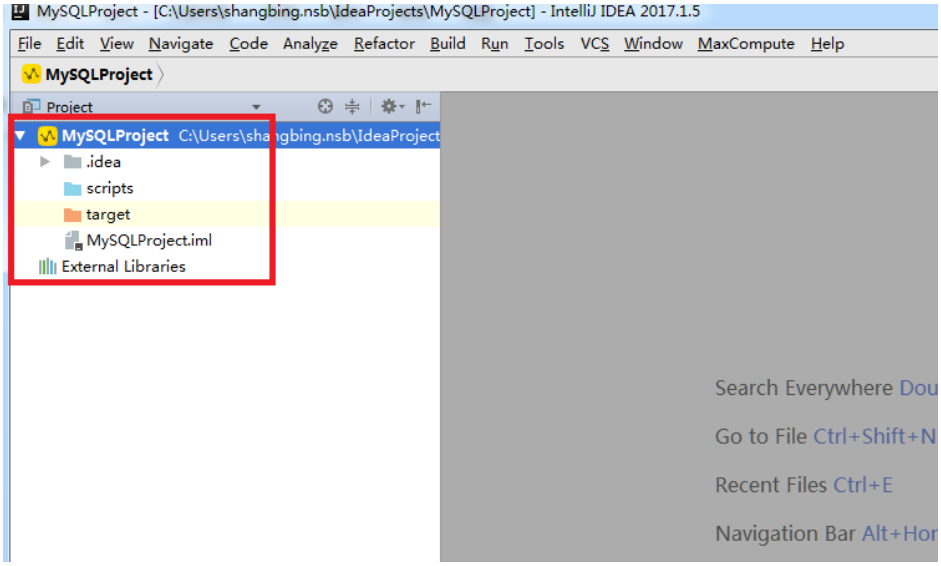


Note:

If a project has been opened before, a dialog box appears, prompting whether to open the new project in the existing window (closing the previous project). Click **This Window**.



After the project is created, the page shown in the following figure appears. You can develop SQL scripts in the project.



Script files exist locally

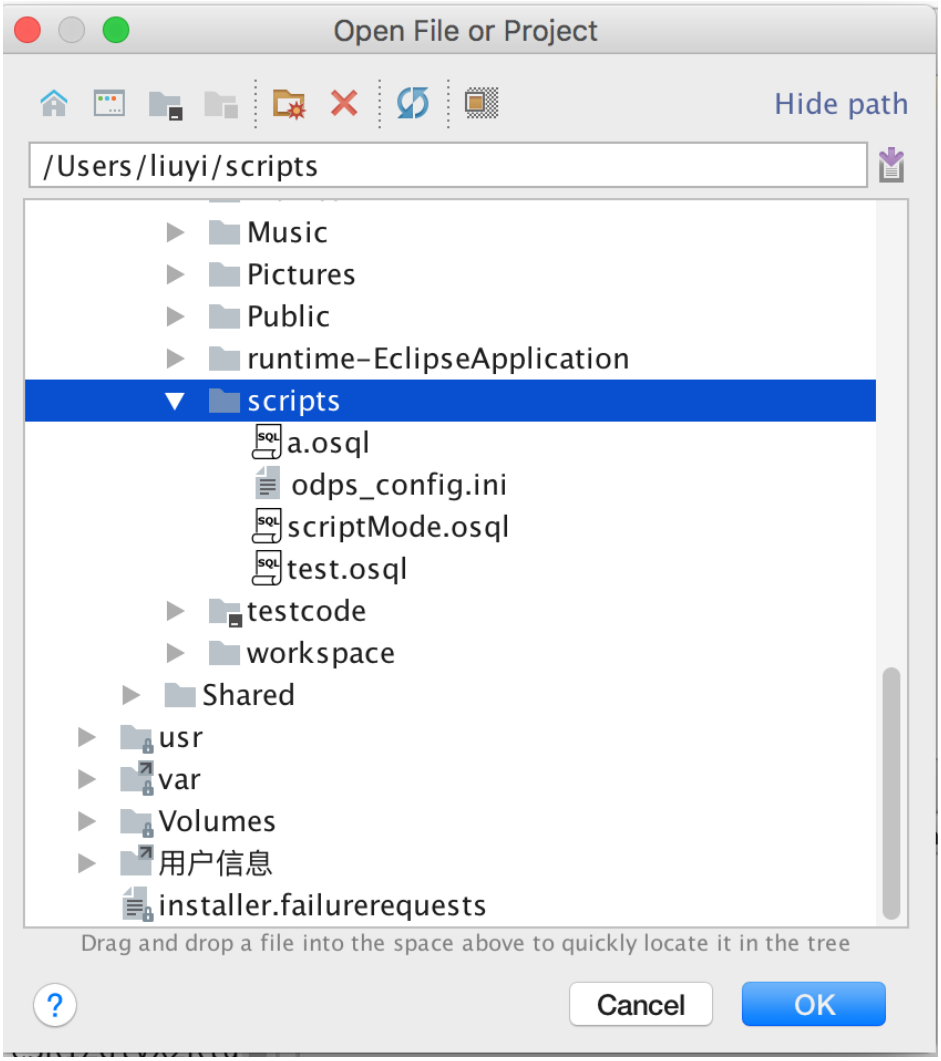
If many scripts have been stored in a local folder, MaxCompute Studio is used to edit the scripts. You can open a module directly.

Procedure

Create a connection configuration file odps_config.ini for MaxCompute in the **scripts** folder, and configure authentication information for connecting to MaxCompute.

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

Open IntelliJ IDEA, select **File > Open**, and select the **scripts** folder.



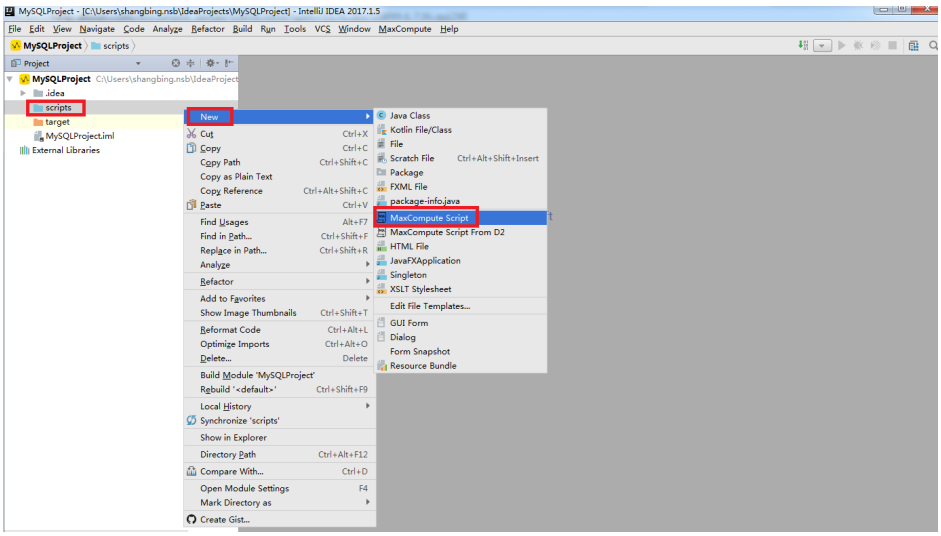
MaxCompute Studio detects whether the odps_config.ini file exists in the folder, captures metadata on the server based on the configuration information in the file, and compiles all scripts in the folder.

Write SQL scripts

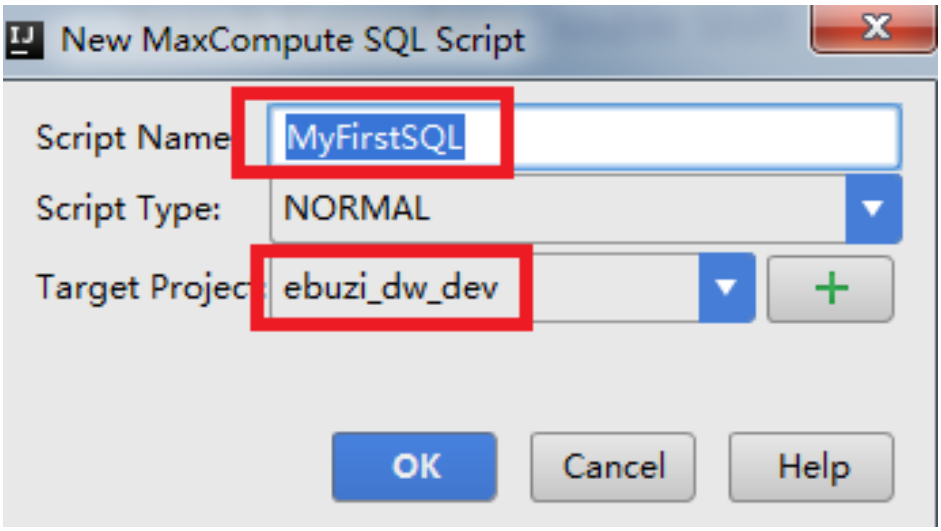
After MaxCompute Studio module is created, you can compile a MaxCompute SQL script.

Procedure

Right-click **scripts** and select **New > MaxCompute Script**.



In the dialog box that appears, specify related content and click **OK**.



- Script Name: Indicates the script name.
- Script Type: Indicates the script type.
- Target Project: Indicates the target MaxCompute project.

In the preceding dialog box, you can click **+** next to **Target Project** to create a MaxCompute project. For more informatioan about on how to configure a MaxCompute project, see Create a project connection.

Compile an SQL script on the SQL file editing page.

Note:

i. Compile the SQL script based on the tables of your MaxCompute project. You can click the upper-right corner on the toolbar to switch to different bound MaxCompute projects. Cross-project resource dependency is supported. For

example, if a script bound to Project A uses Project B Table1, MaxCompute Studio automatically uses the account of Project A to capture the metadata of Project B. MaxCompute Studio stores the metadata of the table in a local directory similar to the following figure.

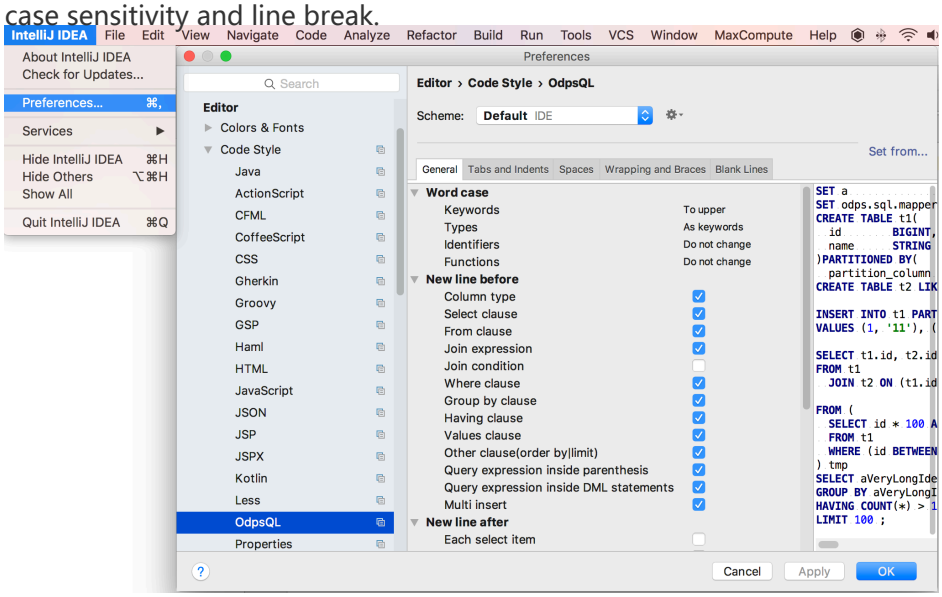
```
liuyi-MBP:hy_test liuyi$ pwd
/Users/liuyi/.odps.studio/meta/odps_studio_dev/tables/hy_test
liuyi-MBP:hy_test liuyi$ cat schema.ddl
CREATE TABLE IF NOT EXISTS `odps_studio_dev`.`hy_test` (
  `id` BIGINT,
  `name` STRING);
```

ii. You can modify the code template used to compile an SQL script on the Intellij Preferences page.

Functions of MaxCompute Studio

MaxCompute Studio supports the syntax highlighting, smart reminders, and error prompting functions. It also supports:

- **Schema annotator**: When you place the cursor over a table, its schema is displayed. When you place the cursor over a function, its signature is displayed.
- **Code folding**: You can fold subqueries to read long SQL scripts.
- **Brace matching**: If you click a left brace to highlight it, the matching right brace is also highlighted. If you click a right brace to highlight it, the matching left brace is highlighted.
- **Go to declaration**: Press Ctrl and click **table** to view table details. Click **function** to view the source code.
- **Code formatting**: Supports formatting for the current script. The keyboard shortcut is Ctrl+Alt+L. You can create custom formatting rules on the following page, such as keyword case sensitivity and line break.



- **Find usages**: Select a table or function in the editor, right-click the table, and select **Find Usages**. All scripts using the table or function will be searched for in the current IntelliJ

project.
- **Live template**: MaxCompute Studio has built-in SQL live templates, which can be opened by pressing Ctrl+J (Command+J on macOS X) in the compiler. For example, if you forget the syntax of **insert into table**, you can open the live template dialog box and search for **insert table**.
- **Built-in documentation**: Supports opening the help documentation by pressing Ctrl+Q (Ctrl+J on macOS X) in system built-in functions.
- **SQL history**: All the SQL statements submitted using the MaxCompute Studio are stored locally. Click an icon on the toolbar, and the **SQL History** window appears, listing the SQL statements that have been executed.

Submit SQL scripts

MaxCompute Studio directly submits MaxCompute SQL scripts to the server for running, and displays detailed information about the query result and execution plan. Before submission, MaxCompute Studio compiles scripts to effectively prevent compilation errors that are detected after the scripts are submitted to the server.

Prerequisites

Create a MaxCompute project connection and bind it to the target project.

Create a MaxCompute Studio module.

Before submission, perform setting as required. MaxCompute Studio provides various setting features. You can perform quick setting on the toolbar at the top of the editor page. The following three types of setting can be performed:

Compiler Mode: It can be set to **Script Mode** or **Statement Mode**.

In statement mode, scripts are separated by ; and submitted to the server one by one.

The script mode is newly developed. A whole script can be submitted to the server immediately. The server provides overall optimization, which is more efficient. Therefore, this mode is recommended.

Type System: It mainly solves the compatibility problem of SQL statements, which can be set to the following values:

Legacy TypeSystem: Indicates the type system of original MaxCompute.

MaxCompute TypeSystem: Indicates the new type system introduced by MaxCompute 2.0.

Hive Compatible TypeSystem: Indicates the type system in Hive compatibility mode introduced by MaxCompute 2.0.

Compiler Version: MaxCompute Studio provides the stable compiler and experimental compiler.

Default Version: Indicates the stable version.

Flighting Version: Includes the latest features of the compiler.

TIPS:

You can use **Global Settings** to set the submitted scripts. Select **File > Settings > MaxCompute**, select **MaxCompute SQL**, and choose **Compiler > Submit** to set the preceding attributes.

Submit SQL scripts

The top toolbar of the editor provides the **Synchronize** and **Compile and Submit** features.

* **Synchronize**: Updates metadata in SQL scripts, including table names and UDFs. If MaxCompute Studio prompts that a table or function cannot be found, but the table or function obviously exists on the server, you can use this function to update metadata.

* **Compile and Submit**: SQL scripts are compiled or submitted to the server in compliance with pre-released MaxCompute SQL rules. Details of compilation errors are displayed in the **MaxCompute Compiler** window.

Procedure

After SQL statements are compiled, click the green running icon on the toolbar, or right-click Script Editor and select Run MaxCompute SQL Script to submit the SQL statement to the server. If a variable (such as ${bizdate} exists in the SQL statement in the following figure), a dialog box is displayed, prompting you to enter the variable value.

The script will be locally compiled (depending on the project metadata you added in the Project Explorer window). If no compilation error exists, the script is submitted to the server for execution. When the SQL script is being executed, the running logs are displayed. If the script is running on the server, the **Job Details** page is displayed, showing the basic information about job running and the execution diagram.

3. You can view SQL results on the Results page. If there are multiple statements in the single-sentence mode, the result of each statement is displayed. You can select rows or columns in the table, and copy them to the Clipboard.

Develop Java procedure

Create MaxCompute Java Module

MaxCompute Studio supports Java user-defined function (UDF) and MapReduce development. First, a MaxCompute Java module must be created.

Create a module

Choose File > New > Module, set the module type to MaxCompute Java, and configure Java JDK. Click **Next**, enter a module name, and click **Finish**. MaxCompute Studio automatically creates a Maven module and introduces MaxCompute dependencies.

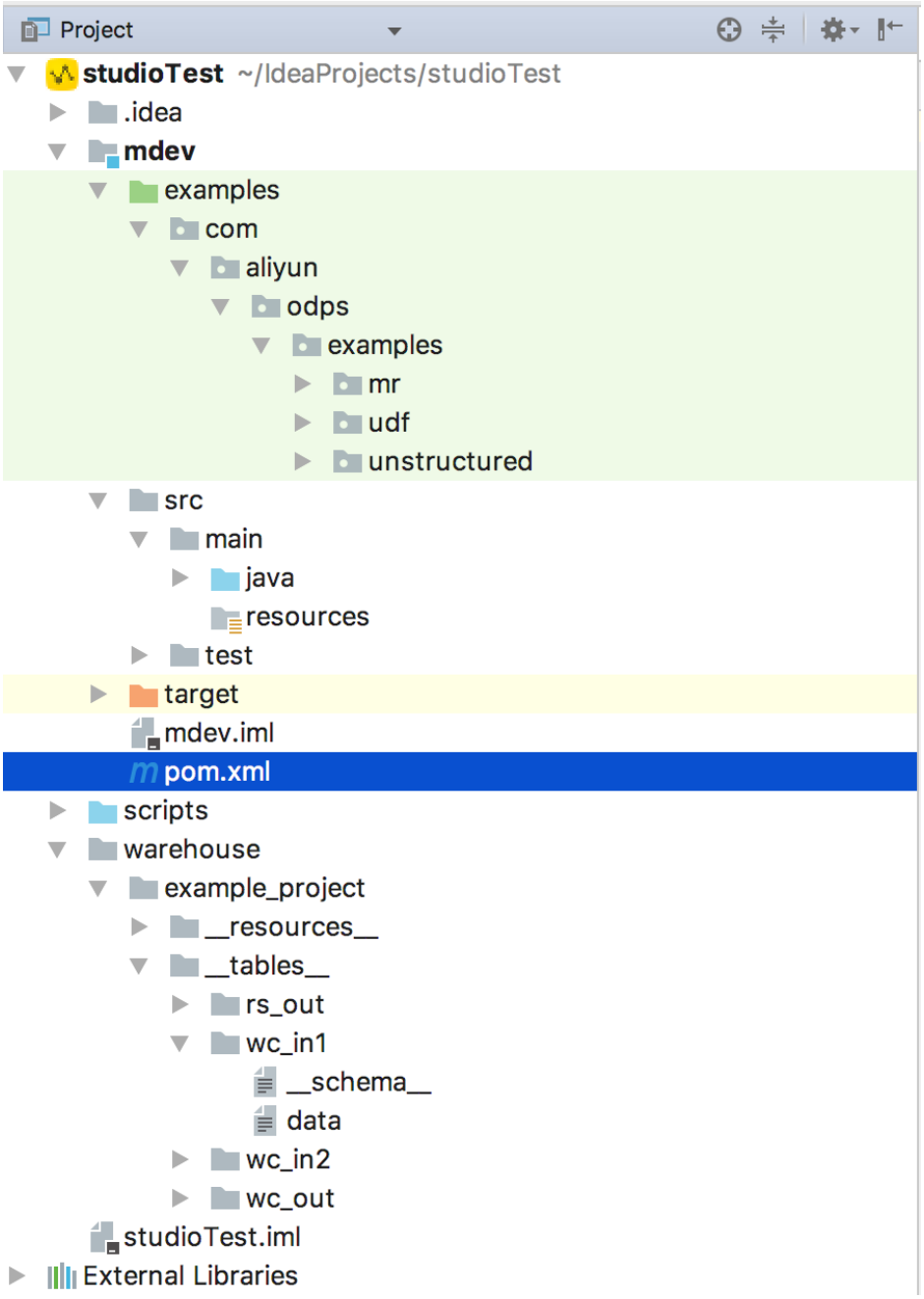
Module structure

So far, a module for developing a MaxCompute Java program has been established, that is the mDev shown in the following figure. Its main directories include:

src/main/java: Source code for Java program development.

examples: Sample code, including unit test (UT) examples. You can see the examples to develop or compile UT.

warehouse: Schema and data required for running locally.



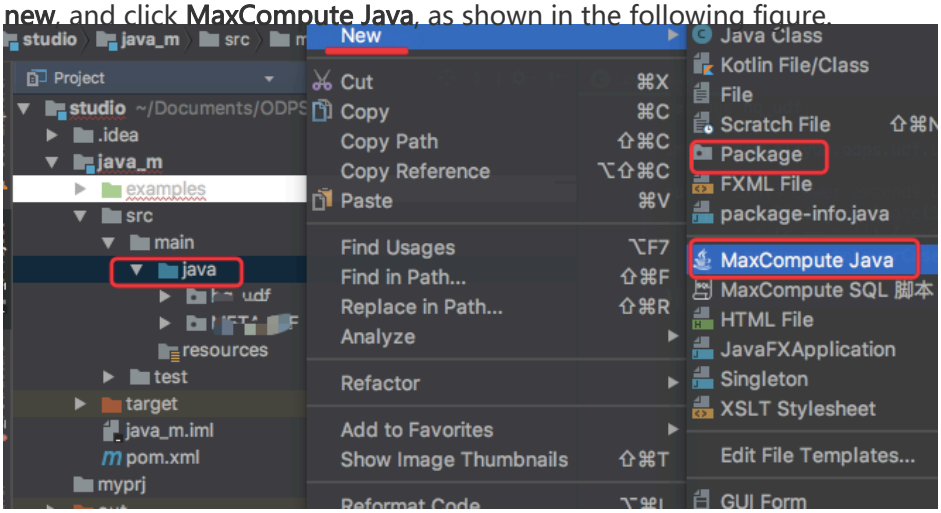
Develop and debug UDF

After the MaxCompute Java module is created, the UDF program can be developed.

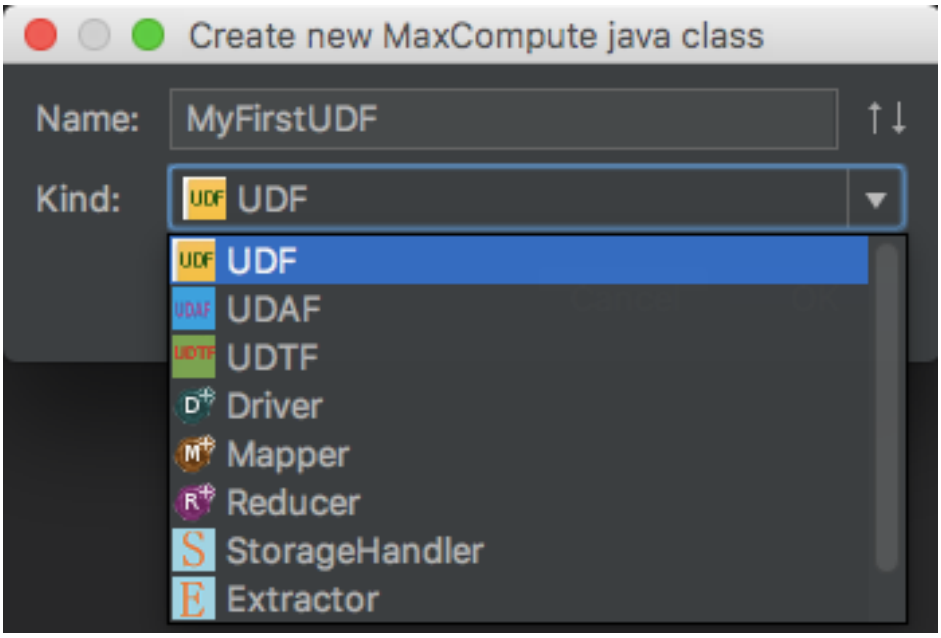
Procedure

1. Unfold the created MaxCompute Java Module directory, navigate to **src > main > java >**

new, and click **MaxCompute Java**, as shown in the following figure.



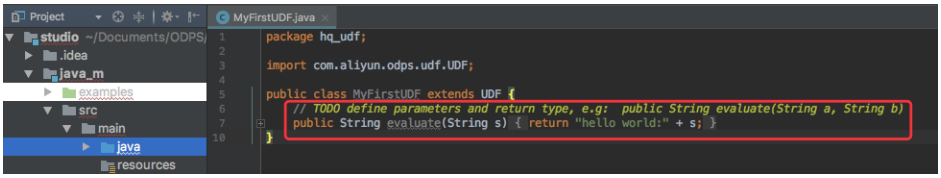
Set **Name** and **Kind**, and click **OK**, as shown in the following figure.



Name: Specifies the name of the MaxCompute Java Class. If you have not created a package, you can enter packagename.classname to automatically create a package.

Kind: Specifies the type. Supported types include custom functions (UDF/UDAF/UDTF), MapReduce (Driver/Mapper/Reducer), and non-structural development (StorageHandler/Extractor).

After the creation is successful, the Java program can be developed, modified, and tested.

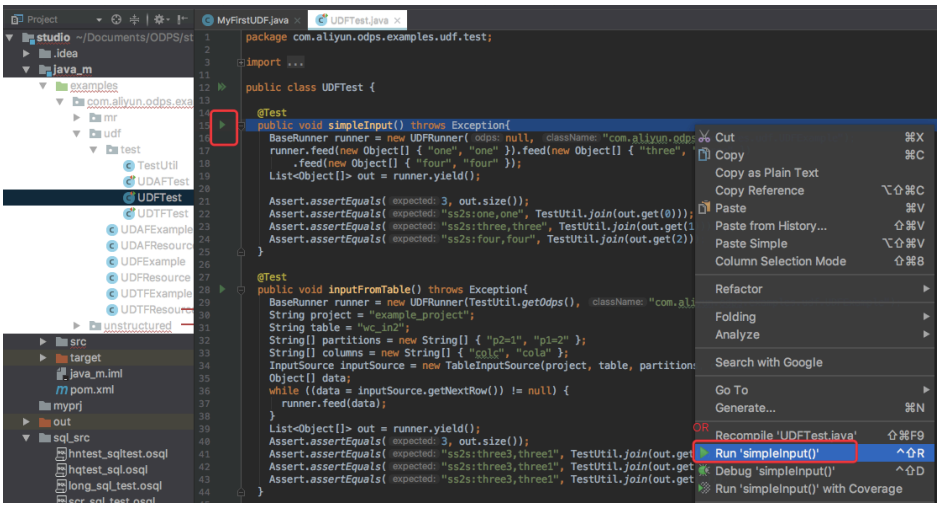


Debug the UDF program

After the UDF program is developed, it can be tested using UT or local running to check whether it meets expectations.

UT

There are various UT examples in the examples directory and you can refer to them to compile your UT.



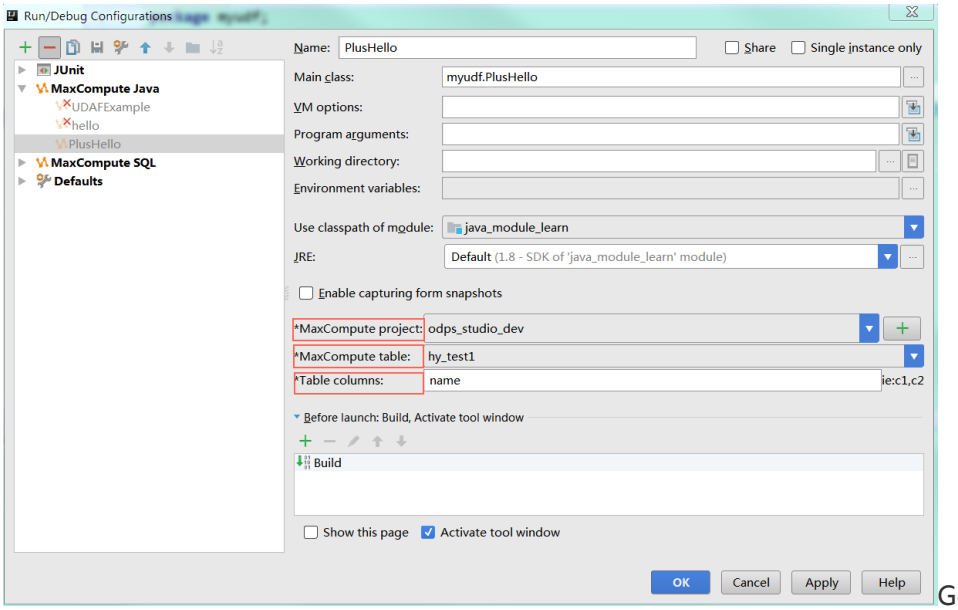
Local running

During local running of the UDF program, the running data source must be specified. The following two methods are provided to set the test data source:

> MaxCompute Studio uses the Tunnel Service to automatically download table data of a specific project to the warehouse directory.
>
> The mock project and table data are provided. You can see example_project in warehouse to set it by yourself.

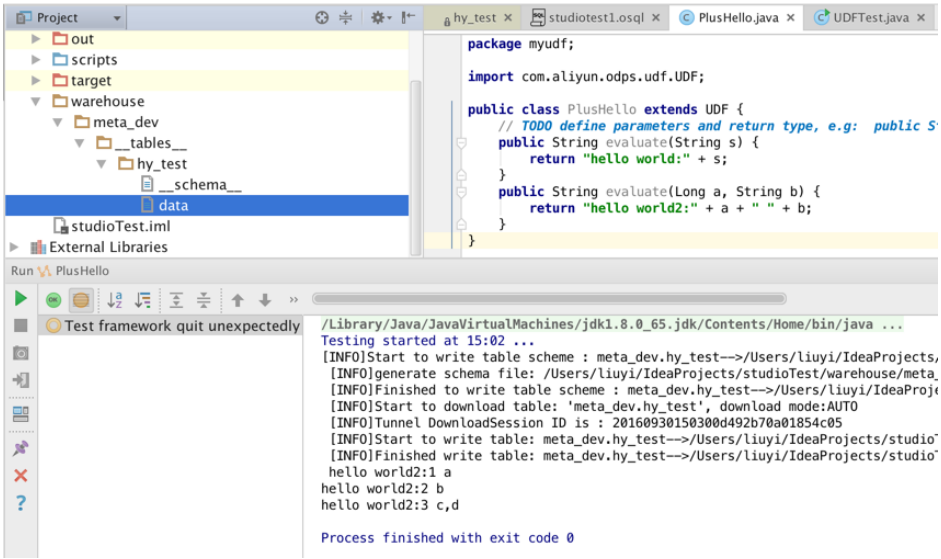
Procedure

Right-click **UDF Class** and select **Run UDF class.main()**. The **Run Configuration** dialog box is displayed.



Generally, UDF/UDAF/UDTF data is used as columns in tables of a select substatement. The MaxCompute project, table, and column must be configured. (**The metadata is from the mock project under project explorer and warehouse.**)

Click **OK**.



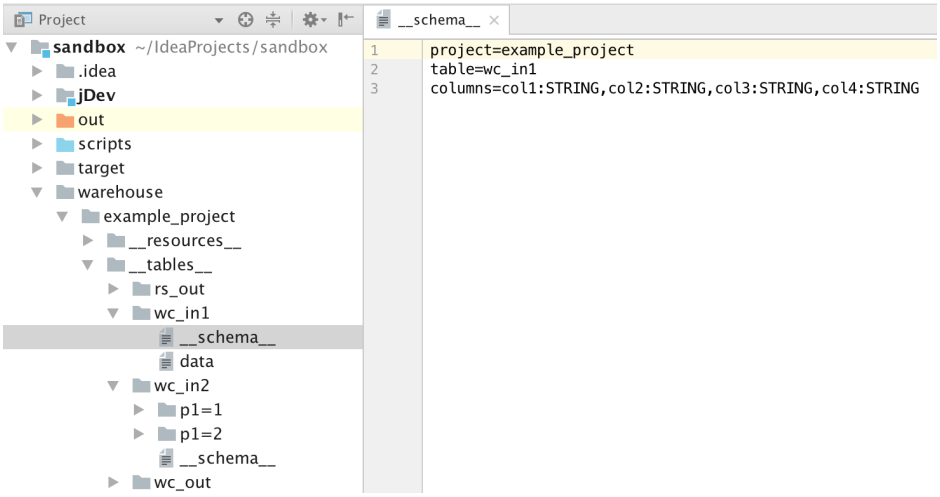
NOTE:

- If table data of a specific project is not downloaded to warehouse, download the data first. By default, 100 data records are downloaded. If more data is required, use the Tunnel Command of the console or table downloading function of Studio.

- If the mock project is used or the table data is downloaded, directly run the program.
- The UDF local run framework uses data in the specific columns in warehouse as the UDF input and runs the UDF program locally. You can view log output and result display on the console.

Local warehouse directory

The local warehouse directory is used to store tables (including meta and data) or resources for local UDF or MR running. The following figure shows the warehouse directory.



NOTE:

- The project name, tables, table name, table scheme, and sample data are under the warehouse directory in sequence.
- The schema file is configured with the project name, table name, and column name and type (separated by a colon) in sequence. For a partition table, the partition column also must be configured. (For a non-partition table, see wc_in1. For a partition table, see wc_in2.)
- The data file uses the standard CSV format to store table sample data.
  - Special characters include comma, double quotation marks, and line feed (\n or \r\n).
  - The column separator is comma and the line separator is \n or \r\n.
  - If the column content includes special characters, double quotation marks (“) must be added before and after the column content. For example, if the column content is 3,No, it is changed to “3, No” .
  - If the column content includes double quotation marks, each double quotation mark is converted to two double quotation marks. For example, if the column content is a” b” c, it is changed to “a” ” b” ” c” .
  - \N indicates that a column is null. If the column content (string type) is \N, it must be converted to “” ” \N” ” ” .

- The file character code is UTF-8.

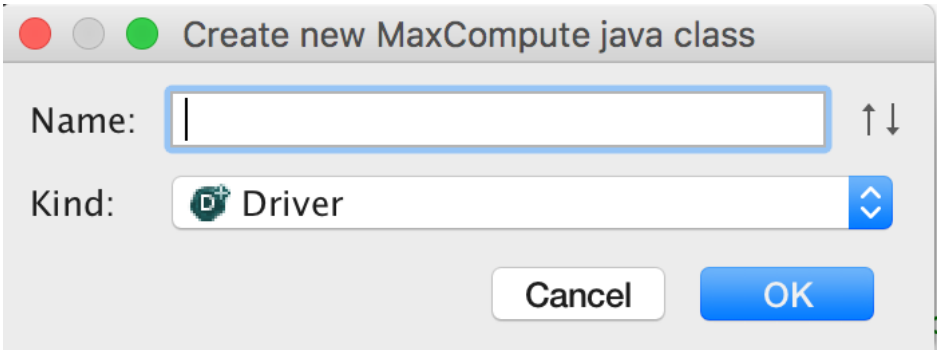
Develop MapReduce

After the MaxCompute Java module is created, MR can be developed.

Develop the MR program

Right-click the module source code directory **src > main**, select **New**, and select **MaxCompute Java**.

Create Driver, Mapper, and Reducer.



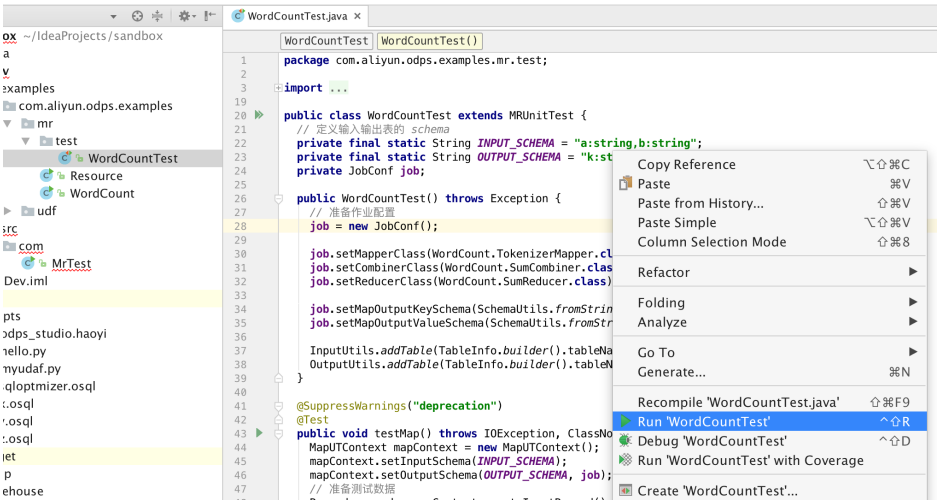
1. Set the input/output table and Mapper/Reducer class. The framework code is automatically filled in the template.

```java
package mymr.myudf;

import ...

public class HelloDriver {

    public static void main(String[] args) throws OdpsException {

        JobConf job = new JobConf();

        // TODO: specify map output types
        job.setMapOutputKeySchema(SchemaUtils.fromString(_?));
        job.setMapOutputValueSchema(SchemaUtils.fromString(_?));

        // TODO: specify input and output tables
        InputUtils.addTable(TableInfo.builder().tableName(_?).build(), job);
        OutputUtils.addTable(TableInfo.builder().tableName(_?).build(), job);

        // TODO: specify a mapper
        job.setMapperClass(_?);
        // TODO: specify a reducer
        job.setReducerClass(_?);

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

    }

}
```

Debug the MR program

After the MR program is developed, test your code and check whether it meets the expectations. The following two methods are supported:

Unit test (UT): There are WordCount UT examples in the examples directory. You can refer to them to compile your UT.



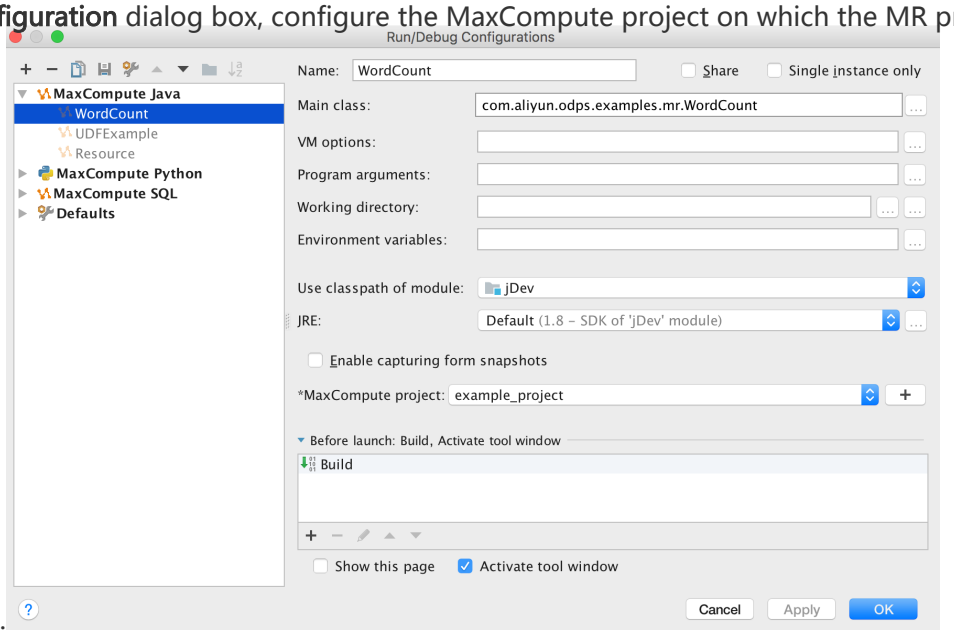
Local MR running: During local running, the running data source must be specified. The following two methods are provided to set the test data source:

> MaxCompute Studio uses the Tunnel Service to automatically download table data of a specific MaxCompute project to the warehouse directory. By default, 100 data records are

downloaded. If more data is required for testing, use the Tunnel Command of the console or table downloading function of MaxCompute Studio.

Provide the mock project (example_project) and table data. You can see example_project in warehouse to set it by yourself.

Run the MR program. Right-click the Driver class and select **Run**. In the displayed **Run Configuration** dialog box, configure the MaxCompute project on which the MR program runs.



Click **OK**. If table data of the specified MaxCompute project is not downloaded to warehouse, download data first. If a mock project is used or the MaxCompute project table data is downloaded, skip this step. Then, the MR local run framework reads specified table data in warehouse as the MR input and runs the MR program locally. You can view log output and result display on the console.

Run the MR program in the production environment

After local debugging is complete, release the MR program to the server and run it in the

MaxCompute distributed environment.

Package the MR program to a JAR package and release it to the server. For more information, see How to package and release MR.

Use the MaxCompute console integrated with MaxCompute Studio in seamless mode, that is, in the **Project Explorer** window, right-click **Project** and select **Open in Console**, and input the commands similar to the following JAR command in the console command line:

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

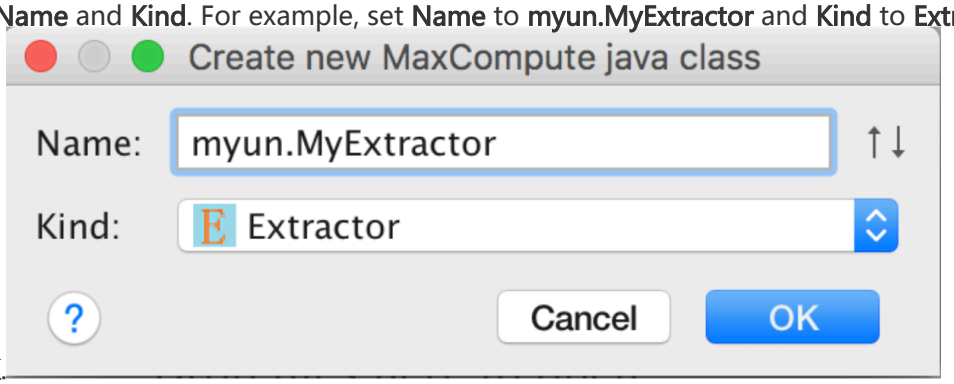
Unstructured development

An unstructured data processing framework is added for MaxCompute 2.0, supporting access to the OSS and Table Store using external tables. MaxCompute Studio provides code templates for the framework, facilitating fast development.

Compile StorageHandler/Extractor/Outputter

1. Create a MaxCompute Java module. (Sample code is provided in the unstructured folder of the examples directory for your reference.)
2. Right-click the module source code directory, that is, **src > main**, select **New**, and select **MaxCompute Java**.

   Specify **Name** and **Kind**. For example, set **Name** to **myun.MyExtractor** and **Kind** to **Extractor**.

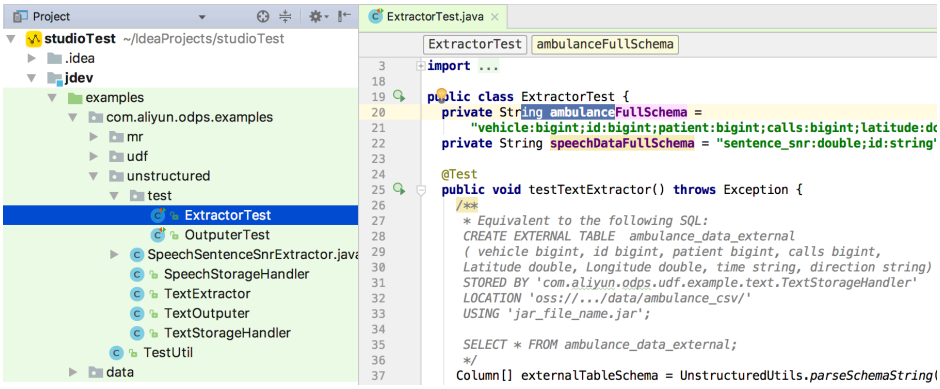
   

   Click **OK**.

   The framework code has been automatically filled in the template. Compile your logic code.
5. Compile Outputter and StorageHandler according to the preceding steps.

UT

You can compile the unit test (UT) by following the examples in the examples directory to test your Extractor/Outputter.

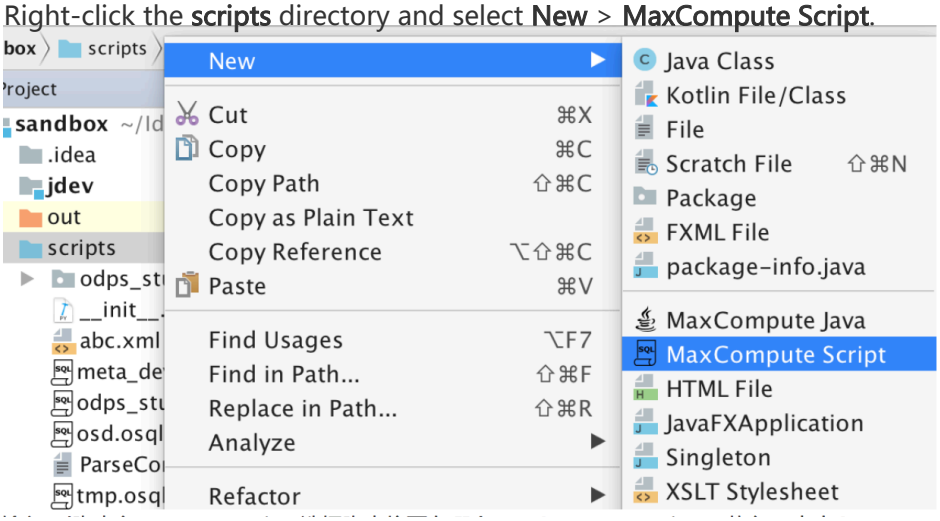


Package and upload

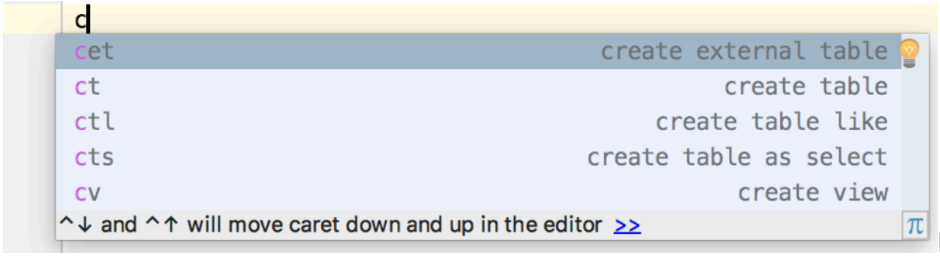
After StorageHandler/Extractor/Outputter is compiled, compress the completed Java program to a JAR package, and upload the package as a resource to the server, see Package and release.

Create external tables

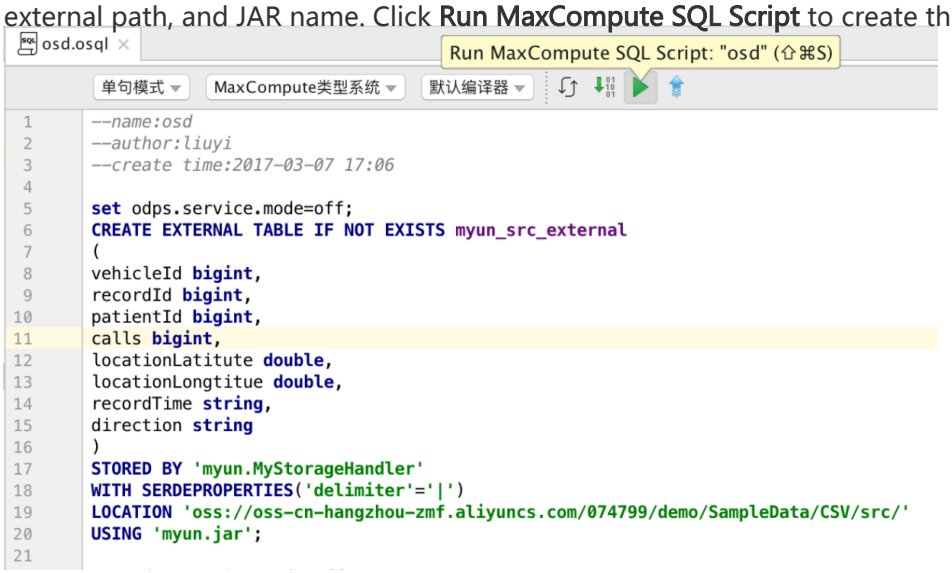
1. Right-click the **scripts** directory and select **New** > **MaxCompute Script**.



2. Enter the SQL script name. Select the MaxCompute project in which the script is to be executed for **Target Project** and click **OK**.
3. Select **create external table live template** in the editor to rapidly insert the script template for creating an external table.



Modify the external table name, column, type, StorageHanlder class path, configuration parameter, external path, and JAR name. Click **Run MaxCompute SQL Script** to create the external table.



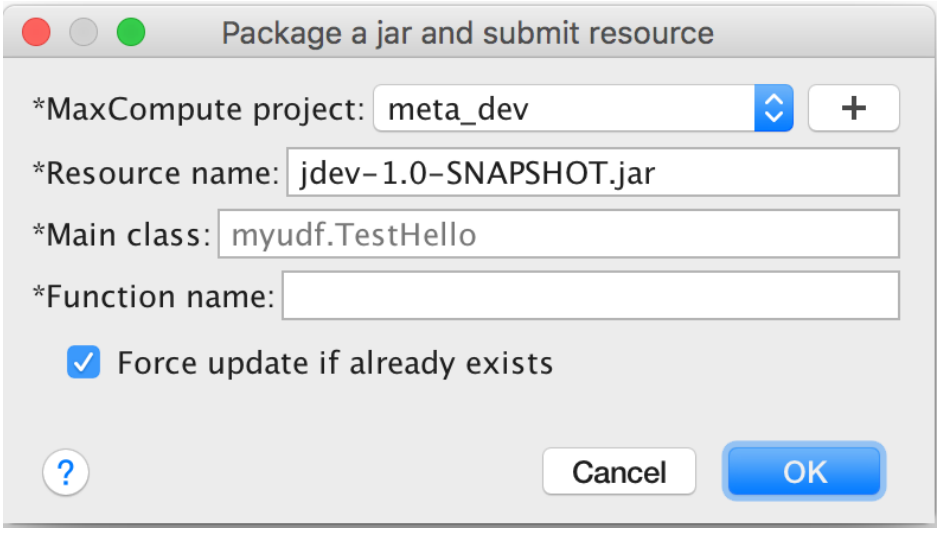
4. Query the created external table.

Package/Upload/Register

After a user-defined function or MapReduce is developed, you must package and release it to the MaxCompute system.

Package a UDF or MapReduce

To release a UDF or MapReduce to the MaxCompute server for production use, you must complete **packaging, uploading, and registration** in sequence. You can use the one-click release function to complete these procedures. MaxCompute Studio runs the **mvn clean package** command, uploads a JAR package, and registers the UDF in one stop. To use this function, right-click the UDF or MapReduce and select **Deploy to server...**. Make sure that the target class is in the **src > main > java** subdirectory and is successfully compiled on the Maven module. The dialog box shown in the following figure appears. Select the MaxCompute project to be deployed and enter a resource name and a function name. Click **OK** and wait until the operation in the background is complete.



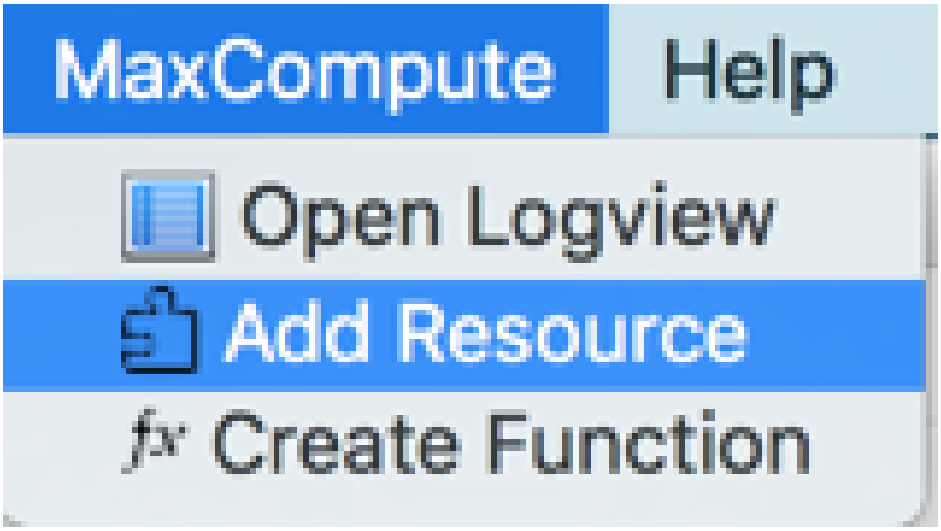
NOTE:

If you require special packaging, you can modify relevant settings in the pom.xml file. After packaging, follow these steps to upload the JAR package and register the UDF.

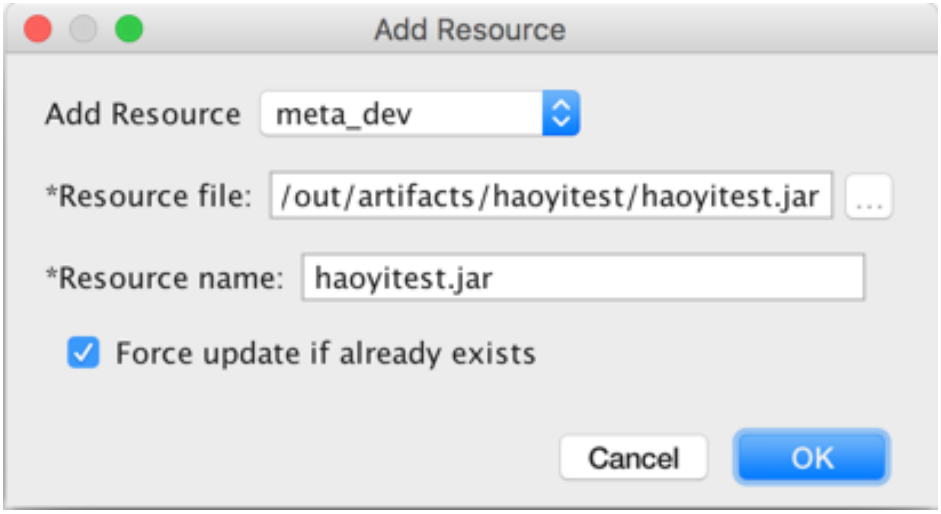
Upload the JAR package

After the JAR package is prepared, upload it to the MaxCompute server.

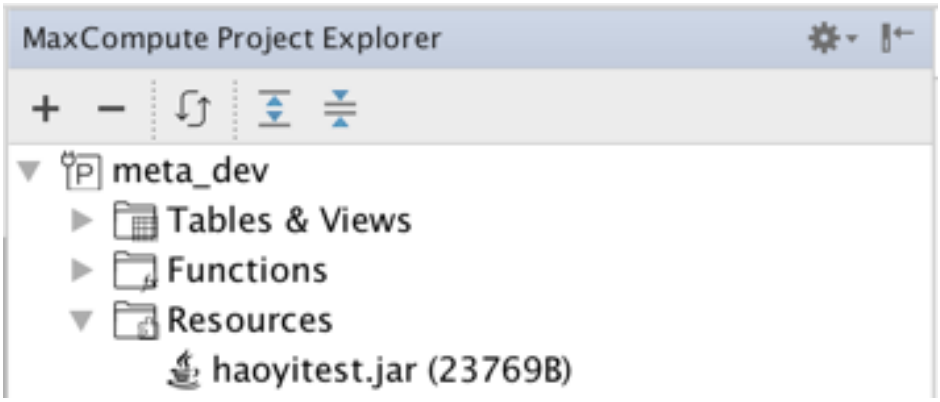
Select **Add Resource** from the MaxCompute menu.



Select the MaxCompute project you want to upload the resource to, the JAR file path, and the resource name you want to register. Determine whether to force update when the resource or function already exists. Then click **OK**.



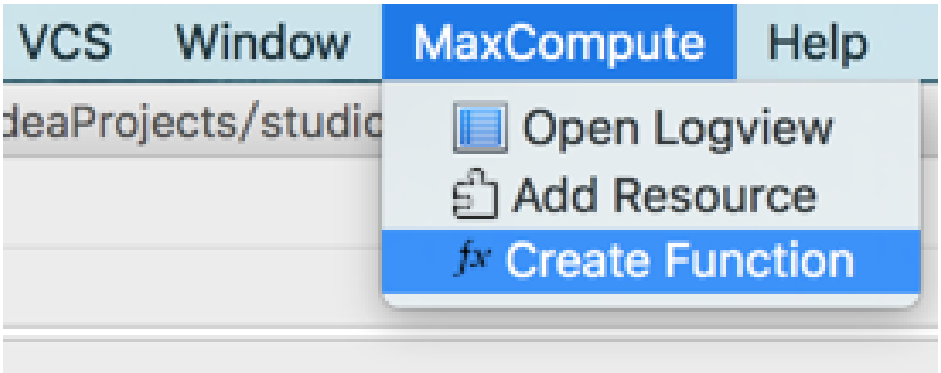
After uploading is successful, you can view the resource under the **Resources** node of the **Project Explorer** window.



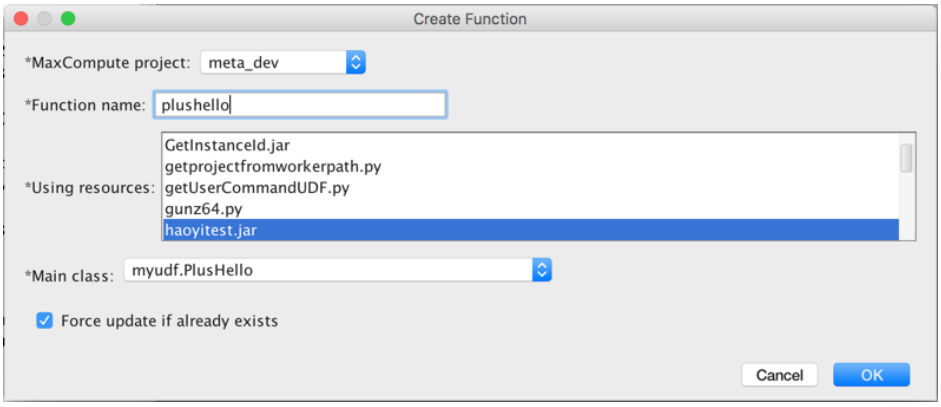
Register the UDF

After the JAR package is uploaded, register the UDF.

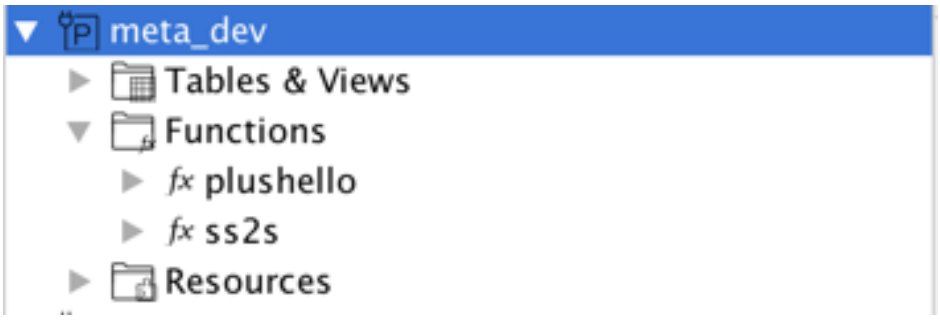
Select **Create Function** from the MaxCompute menu.



Select the required resource JAR and JAR main class, and enter the function name. Click **OK**.

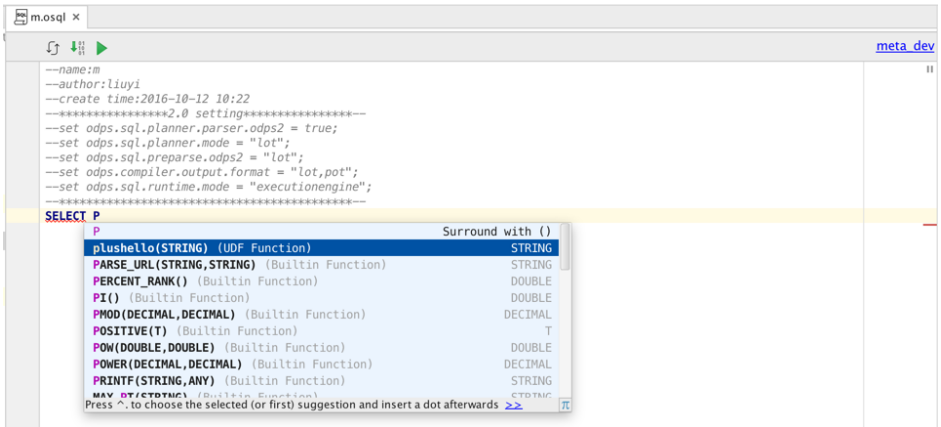


After the registration is successful, you can view the function under the **Functions** node of the **Project Explorer** window.



Apply the UDF

Apply the UDF in SQL to complete subsequent development.



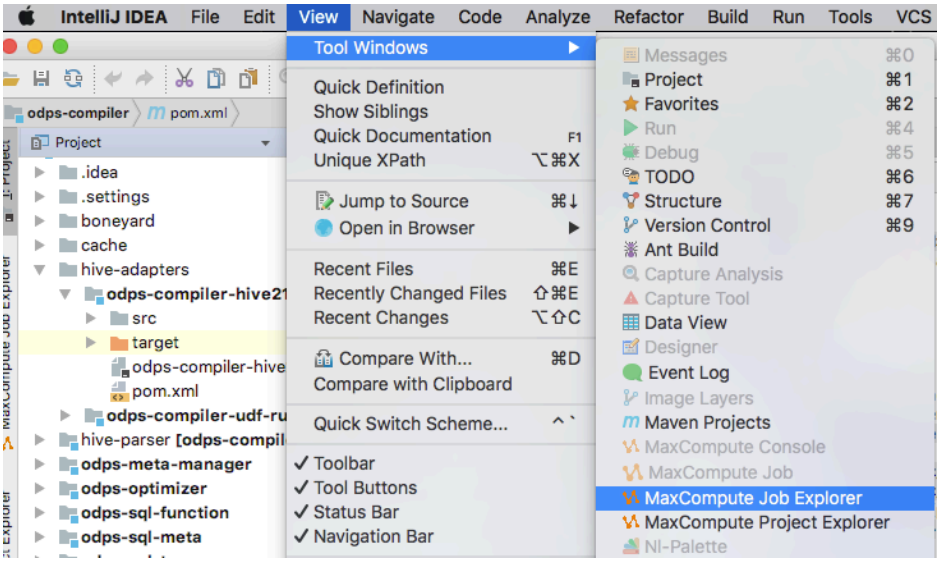
Manage MaxCompute jobs

Job viewing

MaxCompute Studio supports viewing information of MaxCompute running instances submitted by the **current user**, including the running status, job type, and start and stop time.

Open Job Explorer

If Job Explorer View is not displayed on Dock on the left, open Job Explorer by choosing View > Tool > Windows > MaxCompute Job Explorer.



View all job instances in a project

Job Explorer allows you to query submitted job lists by status.

Click the date drop-down box to select another date.

Click **Refresh** to obtain the job list.

> Note:
>
> By default, only the first 1,000 jobs that meet the conditions are displayed. If more than 1,000 jobs meet the conditions, update the filtering conditions.

Sort the job list

You can click the column name in the job list to sort the jobs.

Job queue

If a job in running status is waiting for scheduling in a queue, the job' s location in the queue and global priority is displayed in the job list.

> Note:
>
> The job status and queue location on the **Running Instances** tab are automatically updated. After a job finishes, it is removed from the list.

Save job logs

Currently, Logview logs of a job are saved for seven days by default. If you want to save some important Logview logs for a longer period and view them in the future, you can save them locally.

Double-click a job in the list to display the job details on the right. Click **Save** on the toolbar to save the logs to your local host.

You can set the path for saving the log file on the **Setting** tab of MaxCompute Studio.

Job instance

View a job instance

MaxCompute Studio supports the following two ways to view MaxCompute job instances:

Use a Logview URL or locally stored offline Logview file to open details of a job in read-only mode.

Using Logview to view details of a job is the way familiar to MaxCompute users. By using Logview, you can also view the status of tasks submitted by other users in other projects. You can view details of any job by entering a valid Logview URL using Studio.

Select **MaxCompute** > **Open Logview** from the menu bar. Valid Logview URLs in the

Clipboard will be automatically copied to the displayed dialog box. Alternatively, you can select exporting the locally stored offline Logview file.

In Job Explorer, double-click a MaxCompute instance or right-click a MaxCompute instance and select **Open** to view the instance details.

Job details view

The job details page includes the following five views and two tool windows:

Visualization view: Displays overall information of a job in graphic mode. You can view the subtask dependency and details of subtask instances.

Summary (JSON) view: Displays running details of a job in JSON format.

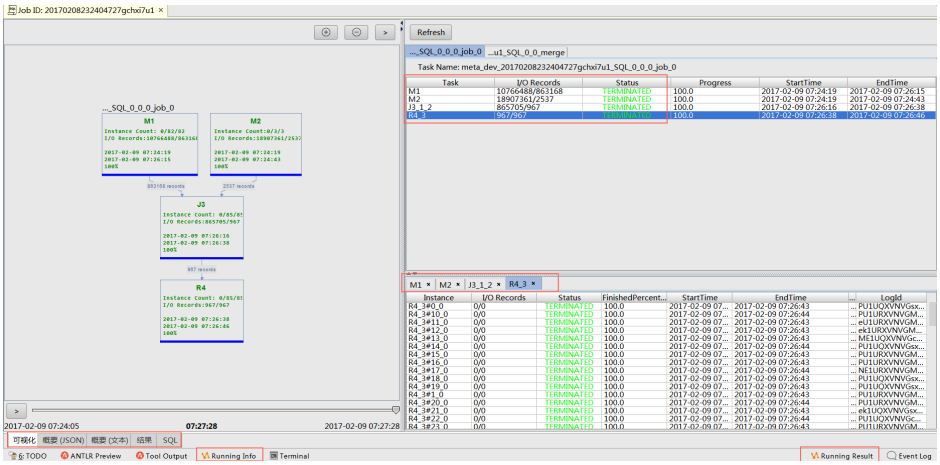
Summary (text) view: Displays running information of a subtask in text mode.

Result view: Displays running results of a job.

SQL view: Displays the corresponding SQL statement when a job is submitted.

Running Info: Tool window that displays the running Logview URL. You can click the URL to switch to Logview.

Running Result: Tool window that displays the running result, which is consistent with the **Result** view.



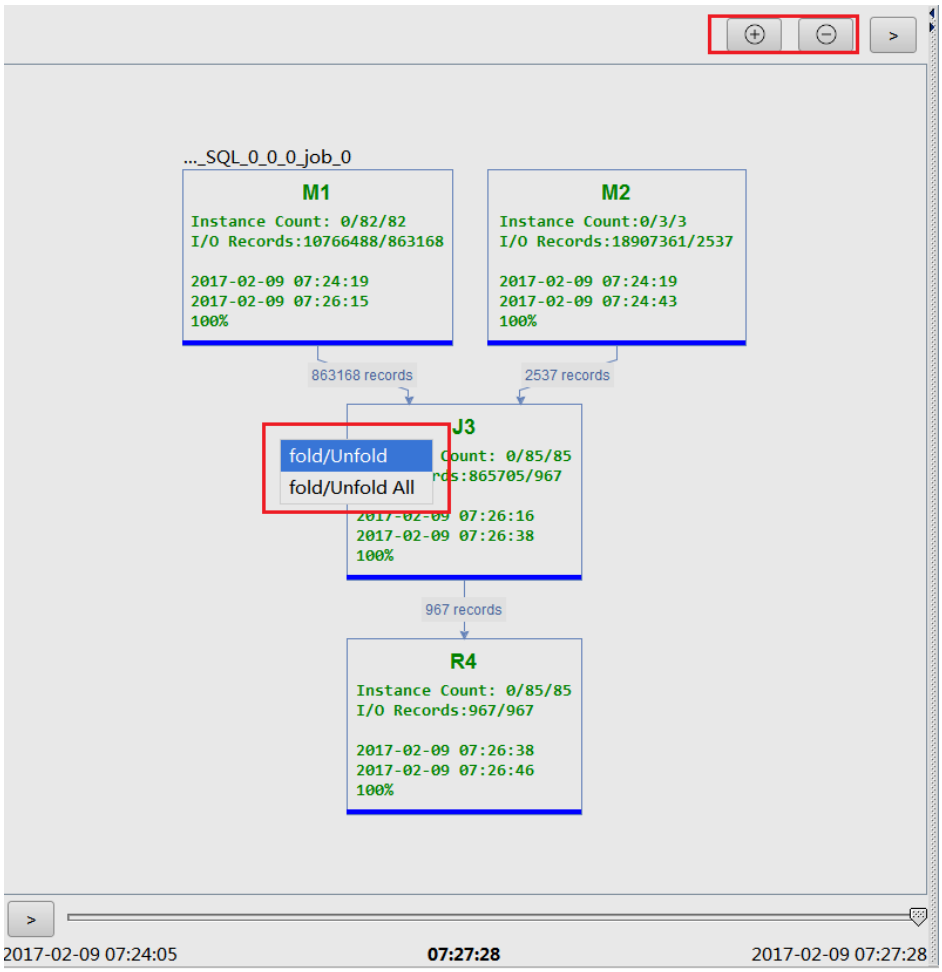
Visualization view details

The following describes details of the **Visualization view**, which is a commonly used tool. You can click the name of another view to switch to the view. The left side of the view displays the execution relationship diagram of a job, and the right side of the view displays the subtask list and the Worker list corresponding to each subtask. You can click **Refresh** to refresh a job and obtain the job details.

Relationship diagram: Illustrates the execution logic among subtasks. You can double-click a subtask or right-click a subtask and select **Fold/Unfold** to view the **POT diagram** of the subtask. You can also click **+** or **-** to zoom in or out the diagram.

TIPS:

You can press **Ctrl+scroll wheel** to zoom in or out the diagram.



Information about the relationship diagram:

Instance Count: a/b/c: Indicates the number of running subtask instances (a),

number of completed instances (b), and total number of instances (c) in a subtask at a time point.

I/O records: Indicates the numbers of input and output records at a time point. **(Note that there is no total number of records because the total number of records cannot be obtained if the subtask is running.)**

Percentage and blue progress bar: Indicate the subtask running status. The proportion is obtained by analyzing instances run in the subtask.

The number displayed on the line between subtasks indicates the number of output records. The arrow indicates the data flow direction.

Subtask list: Lists information about subtasks started by the current job, including the status, number of I/O records, execution progress, and start and end time.

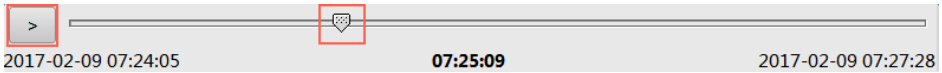
Worker list: Lists information about processes corresponding to each subtask.

Job playback

Studio supports the job playback function. The history of a job can be reviewed within 12s, similar to playing a media file. This function helps you understand the running status of a MaxCompute instance at different time points, rapidly determine the subtask-level running sequence and time consumed, master the key path for executing a job, and accordingly optimize subtasks that run slowly.

Click **>** to play the job, and click **>** again to pause. You can also manually drag the progress bar.

The job start time, playing time, and end time are displayed on the left, in the middle, and on the right of the progress bar, respectively. If a job is in running state, the current time is displayed on the right of the progress bar.

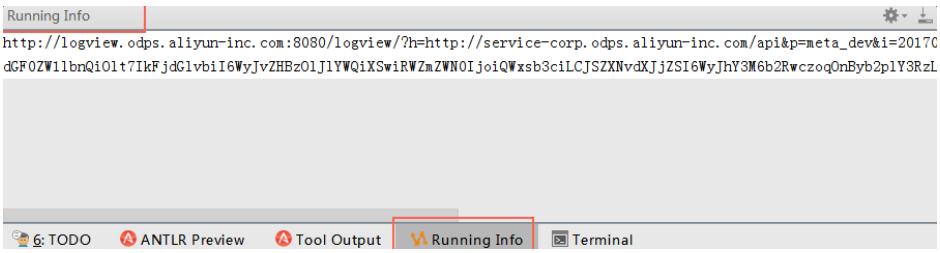


Note:

The playback function only estimates the I/O data volume at a time point by time to determine the completion progress. The estimated I/O data volume does not represent the actual I/O data volume.

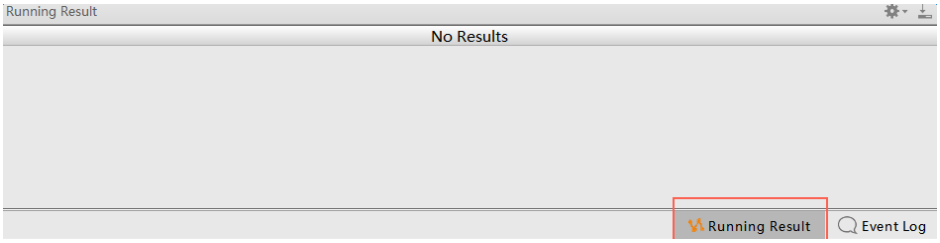
Running Info

Click **Running Information** to view the job running output information, and click the Logview URL to open corresponding Logview using the browser.



Running Result

Click **Running Result** to view the job running data result. (Not all jobs have output data.)



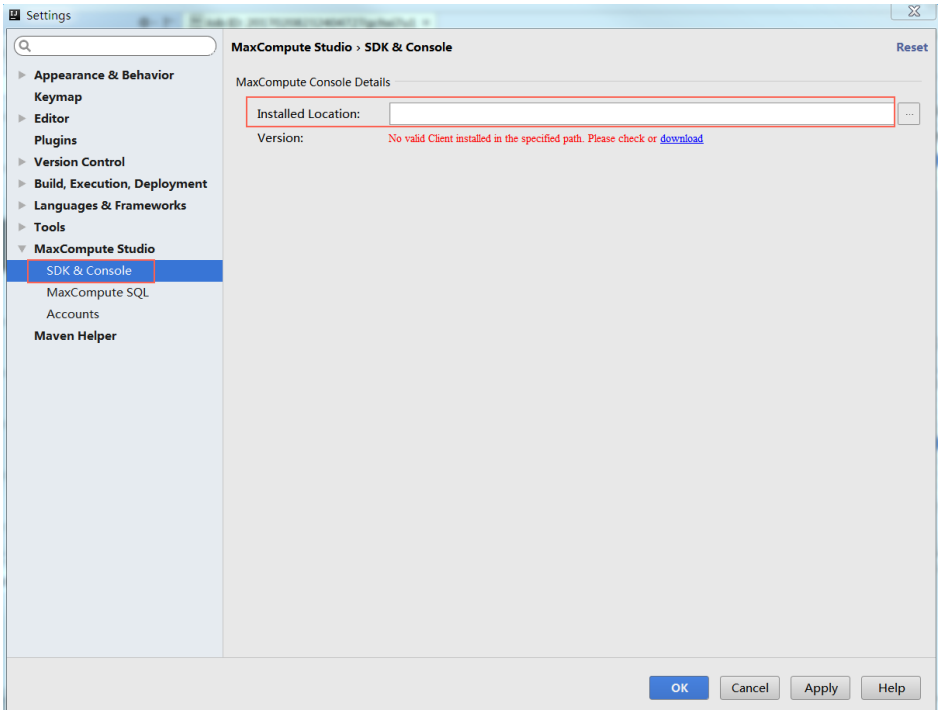
Tool integration

Integrate with MaxCompute client

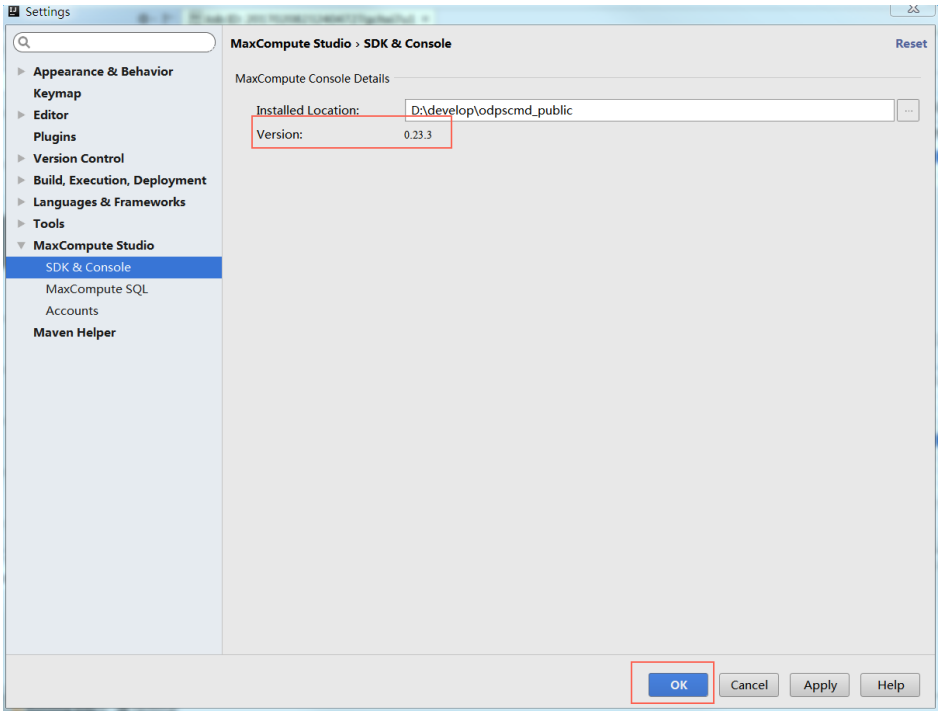
MaxCompute Studio is integrated with the MaxCompute client program. You can open the client on MaxCompute Studio.

Configure the client installation path

> MaxCompute Studio contains the MaxCompute client of the latest version, which is specified as the default client. You can also install the client of another version by selecting **Settings** > **MaxCompute Studio** > **SDK & Console** on IntelliJ IDEA and adding the client program and path. Console download address



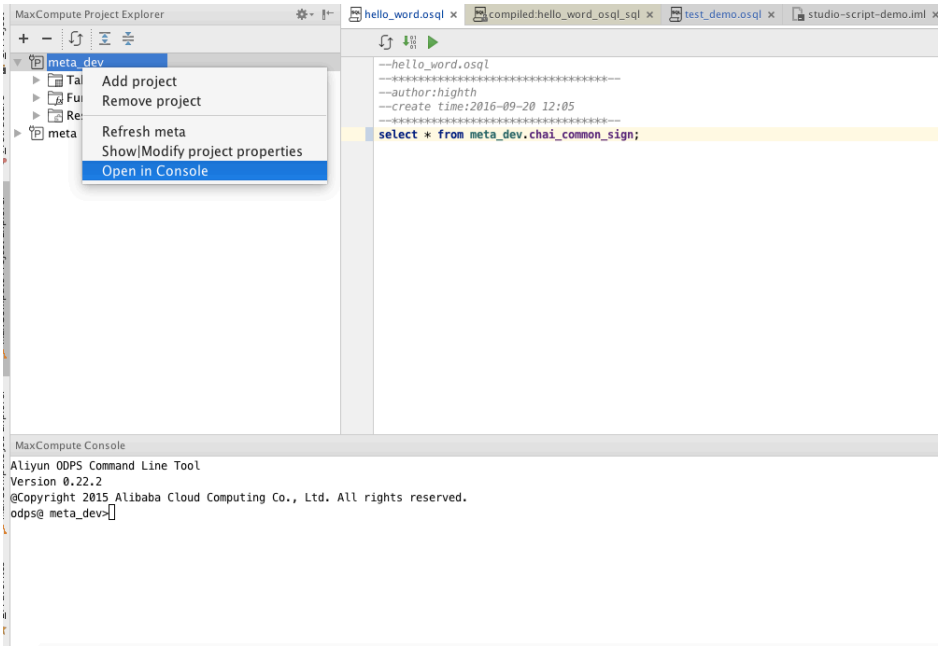
After setting is successful, the MaxCompute client version is displayed.



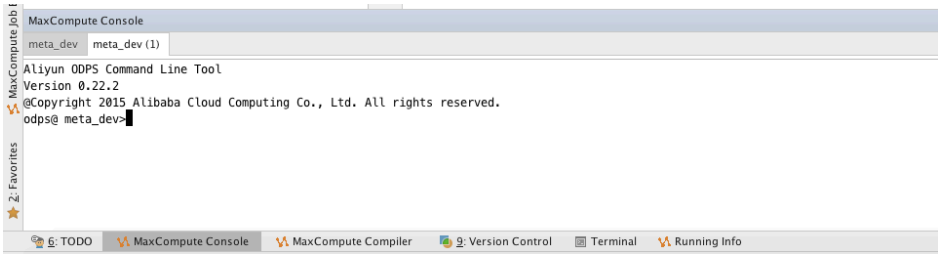
Open the MaxCompute client

After the MaxCompute client installation path is set, you can open the client program on MaxCompute Studio.

In the project browsing list, right-click a project to be opened and select **Open in the console**.



You can open multiple client programs by following the preceding steps.



Configuration items

Configure MaxCompute Studio

After the MaxCompute Studio plug-in is installed, you can find configuration items of MaxCompute Studio on the left bar of the **Settings** page of IntelliJ IDEA.

For more information about how to open the IntelliJ IDEA configuration page, see IntelliJ IDEA

Documentation.

MaxCompute Studio configuration option page

The MaxCompute Studio configuration option page provides the following configuration items:

Path for storing the local metadata base
Specifies the path for locally storing metadata of a MaxCompute project. On MaxCompute Studio, the metadata is stored in the hidden directory .odps.studio/meta of the local user directory by default.

Version update options

- You can use the **Automatically checks for new version** check box to control whether MaxCompute Studio automatically checks for new version updates.
- You can use the **Check new versions** button to manually check new versions. After you click this button, if a new version is available, the **Install new version** button is displayed. You can click this button to install the new version, and restart IntelliJ IDEA after the installation is complete.

SDK and Console configuration option page

The SDK and Console configuration option page provides the following configuration items:

Path for installing a MaxCompute client
Specifies the path for local installation of MaxCompute client. MaxCompute Studio detects the version of the MaxCompute client installed in the path. If detection fails, an error message is prompted.

NOTE:

MaxCompute Studio later than the 2.6.1 version provides the latest MaxCompute client. You do not need to specify the path. If you must use a MaxCompute client of a specific version, you can specify the path.

MaxCompute SQL configuration option page

The MaxCompute SQL configuration option page provides the following configuration items:

Enable syntax coloring

Select **Enable syntax coloring** to enable the syntax highlighting feature.

Enable code completion
Select **Enable code completion** to enable the automatic code complementing feature.

Enable code formatting
Select **Enable code formatting** to enable the code formatting feature.

Compiler options
These are global default compiler options. The following options can be separately set for each file on the toolbar of the SQL compiler.

Compiler Mode

- Statement Mode: In this mode, the compiler compiles and submits a single statement of an SQL file as a unit.

  Script Mode: In this mode, the compiler compiles and submits an entire SQL file as a unit.

  NOTE:

  Script Mode enables the compiler and optimizer to optimize the execution plan and improve the overall execution efficiency. This mode is in the test phase now.

- **Type System**
  - Legacy TypeSystem: Indicates the type system of original MaxCompute.
  - MaxCompute TypeSystem: Indicates the new type system introduced by MaxCompute 2.0.
  - Hive Compatible TypeSystem: Indicates the type system in Hive compatibility mode introduced by MaxCompute 2.0.
- **Compiler Version**
  - Default Version: Indicates the default version of the compiler.
  - Flighting Version: Indicates the experimental version of the compiler, which includes new features of the compiler being tested.

Account configuration option page

You can add or manage accounts used to access MaxCompute on the Account configuration option page. For more information, see User authentication.

You must specify an account on MaxCompute Studio to access a MaxCompute project and run or submit jobs. MaxCompute Studio currently supports the following account type:

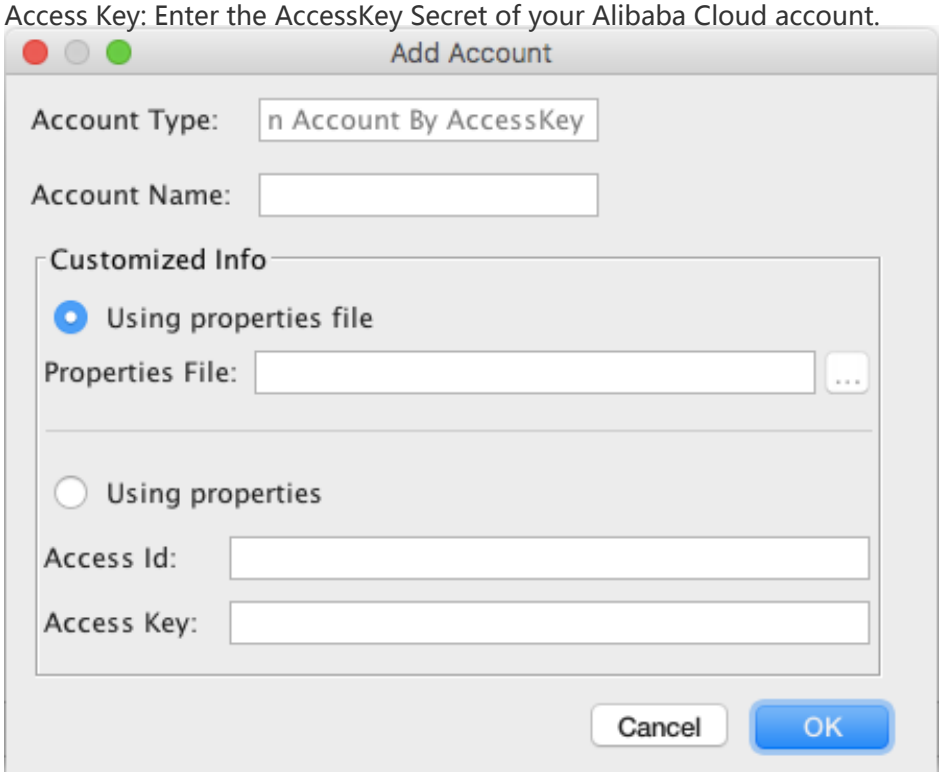
- Alibaba Cloud account (AccessKey)

Add an account

On the Account configuration option page, follow these steps:

1. Click + or press Ctrl-N.
2. Select the account type **Alibaba Cloud Account by AccessKey**.
3. In the displayed **Add Account** window, set the following items:

- Account Name: Indicates the name of the account on MaxCompute Studio.
- Using properties file: Read the AccessKey ID and AccessKey Secret from the configuration file.
    - Select the configuration file conf/odps_config.ini after you process User authentication.
- Using properties: Manually enter the AccessKey ID and AccessKey Secret.
    - Access Id: Enter the AccessKey ID of your Alibaba Cloud account.
    - Access Key: Enter the AccessKey Secret of your Alibaba Cloud account.



1. Click **OK** to complete addition. Then, the account will be displayed in the Account list on the Account configuration option page.

Delete an account

On the Account configuration option page, follow these steps: (This operation only deletes the account configuration on Studio configuration, which does not affect your account.)

1. Select the account to be deleted in the Account list.
2. Click -.
3. In the displayed dialog box, click **OK**.

Modify the AccessKey of an account

On the Account configuration option page, follow these steps:

1. Select the account to be deleted in the Account list.
2. Click the pencil icon.
3. In the displayed **Edit Account** window, modify the account information. The content is similar to that in the preceding section **Add an account**.

FAQ

How to develop UDF using Studio

How to develop the MaxCompute Java UDF using MaxCompute Studio?

1. Add a module. For more information, see Create a MaxCompute Java module. The UDF code is stored in the module.

2. Develop the UDF. For more information, see UDF development. MaxCompute Studio provides the UDF development template. You can complete UDF development based on the template.

3. Perform a test. MaxCompute Studio provides the mechanism for local UDF debugging. You can compile your own test cases based on the UDF test template.

4. Package the UDF source code. You can use the packaging function provided by Data IDE to package the UDF source code to a .jar package.

5. Register and release the UDF. After the .jar package is prepared, add resources and register functions. After functions are registered, the UDF can be viewed in the **Functions** node of the **Project Explorer** window of MaxCompute. The UDF can also be used in the script editor.

How to manage MaxCompute metadata using Studio

During routine MaxCompute usage, you must browse and manage metadata (including tables, functions, and resources) of projects. The following describes how to browse and manage metadata using MaxCompute Studio.

1. Add a project connection. For more information, see **Add connections**. Add a MaxCompute project connection in the **Project Explorer** window of MaxCompute. After the connectivity test is successful, the added project node tree can be viewed.

2. View the table list and schema. For more information, see **Browse meta**. Expand **Tables & Views** in **Project** to view the table and view lists. Expand a specific table to view the column and type.

3. Query the function code. Expand **Functions** to view the function list. Expand a specific function to view the method signature. Double-click the function to view the decompiled source code.

4. View the sample data of a downloaded table. For more information, see **Import and export table data**. Double-click a table to view the detailed schema. In the **Data Preview** window, right-click **Export Grid Data** or right-click a table name and select **Export** to preview sample data.

5. Update node metadata. For example, if a column is added to a table, right-click the table and select **Refresh Meta** or click **Refresh** on the toolbar to view the added column. If tables are added in a project, select **Tables & Views** and click **Refresh** on the toolbar to view the newly created tables.

6. Perform the DDL operation on tables. The deletion operation can be performed in batches. Right-click the selected table and select **Delete table from server**. Table addition and edition cannot be performed on the interface. You can compile corresponding DDL statements in the editor to complete table addition and edition.

Eclipse Plugins

Install

To facilitate the development work with Java SDK of **MapReduce** and **UDF**, MaxCompute provides Eclipse Development Plug-in.

This plug-in can simulate the running process of MapReduce and UDF to provide local debugging methods and simple template generation.

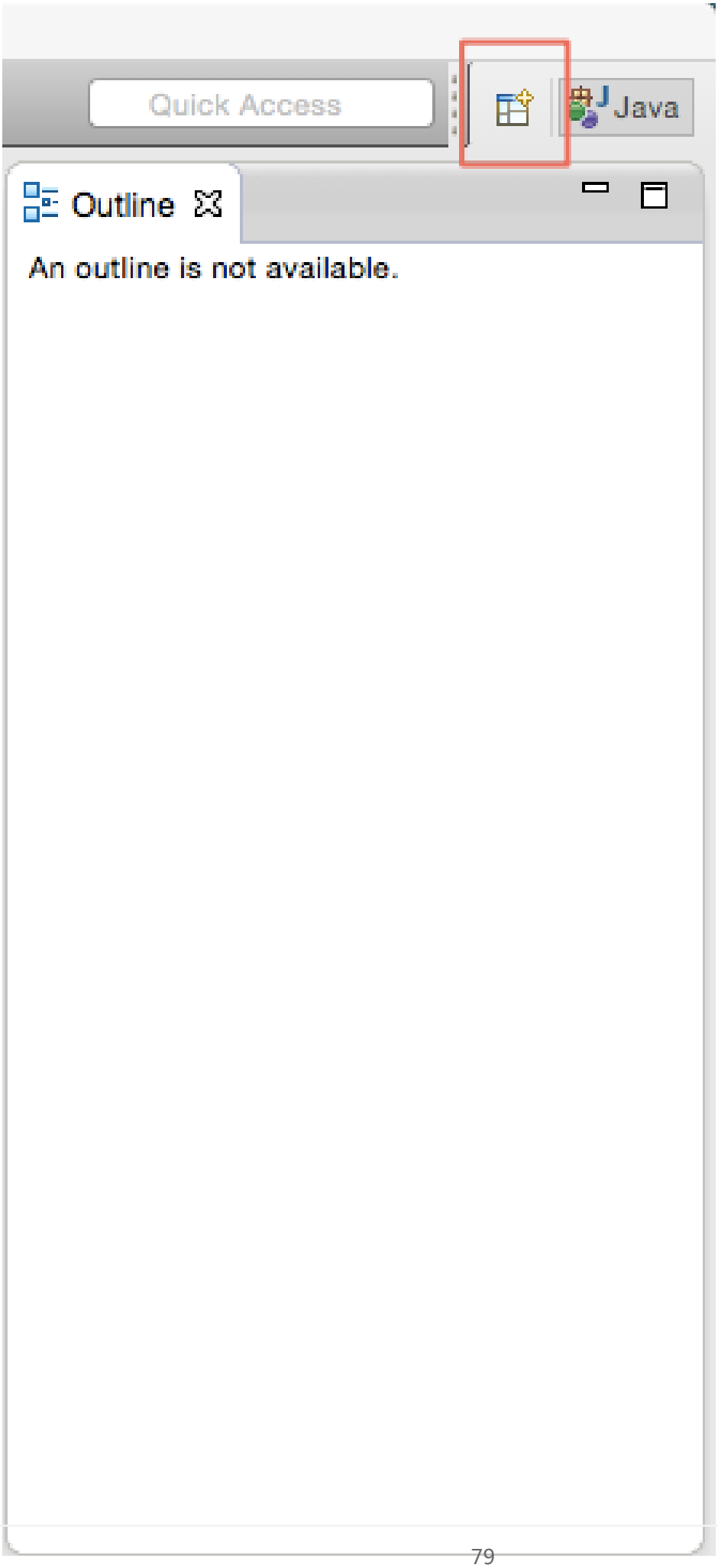
Note:

- To download this plug-in, click Here.
- Unlike the local running mode provided by MapReduce, Eclipse plug-in cannot synchronize data with MaxCompute. Data must be manually copied to the warehouse directory of Eclipse plug-in.

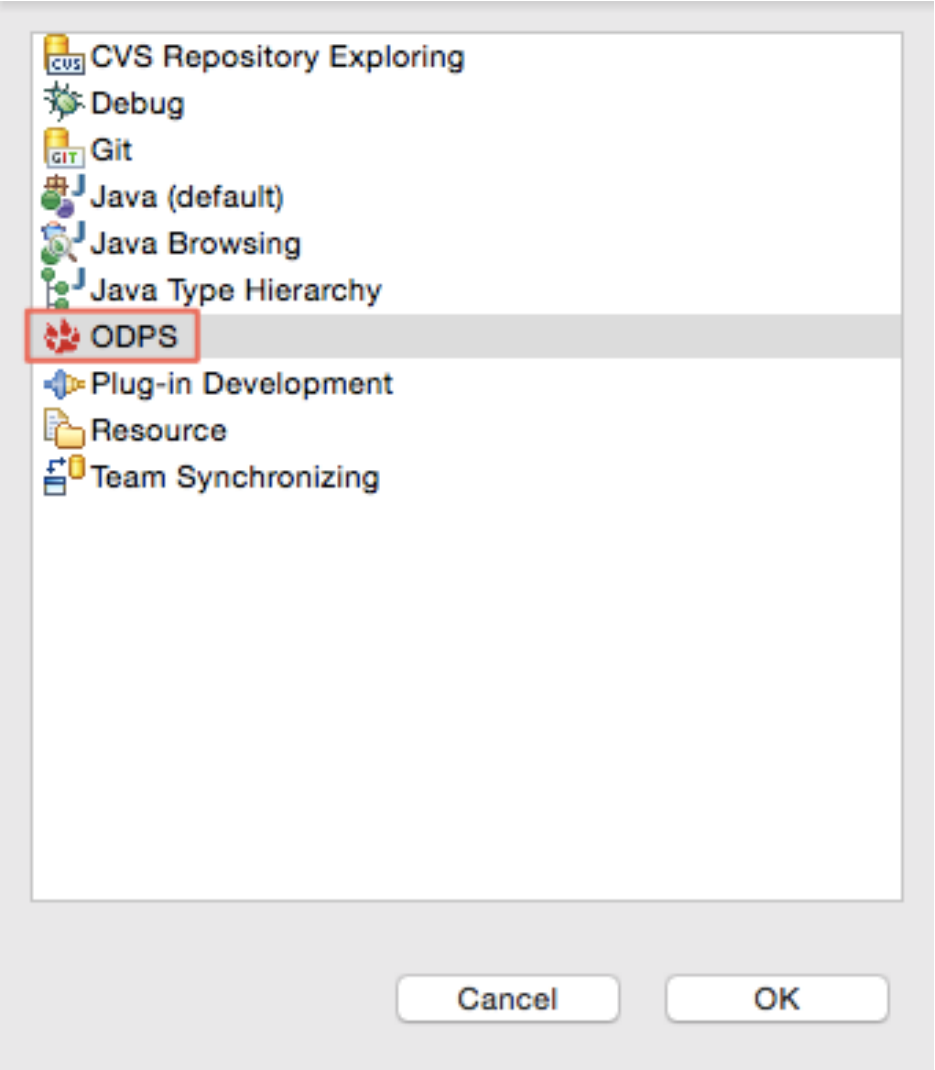
After downloading the Eclipse plug-in, decompress the software package to find the following jar:

```
odps-eclipse-plugin-bundle-0.15.0.jar
```

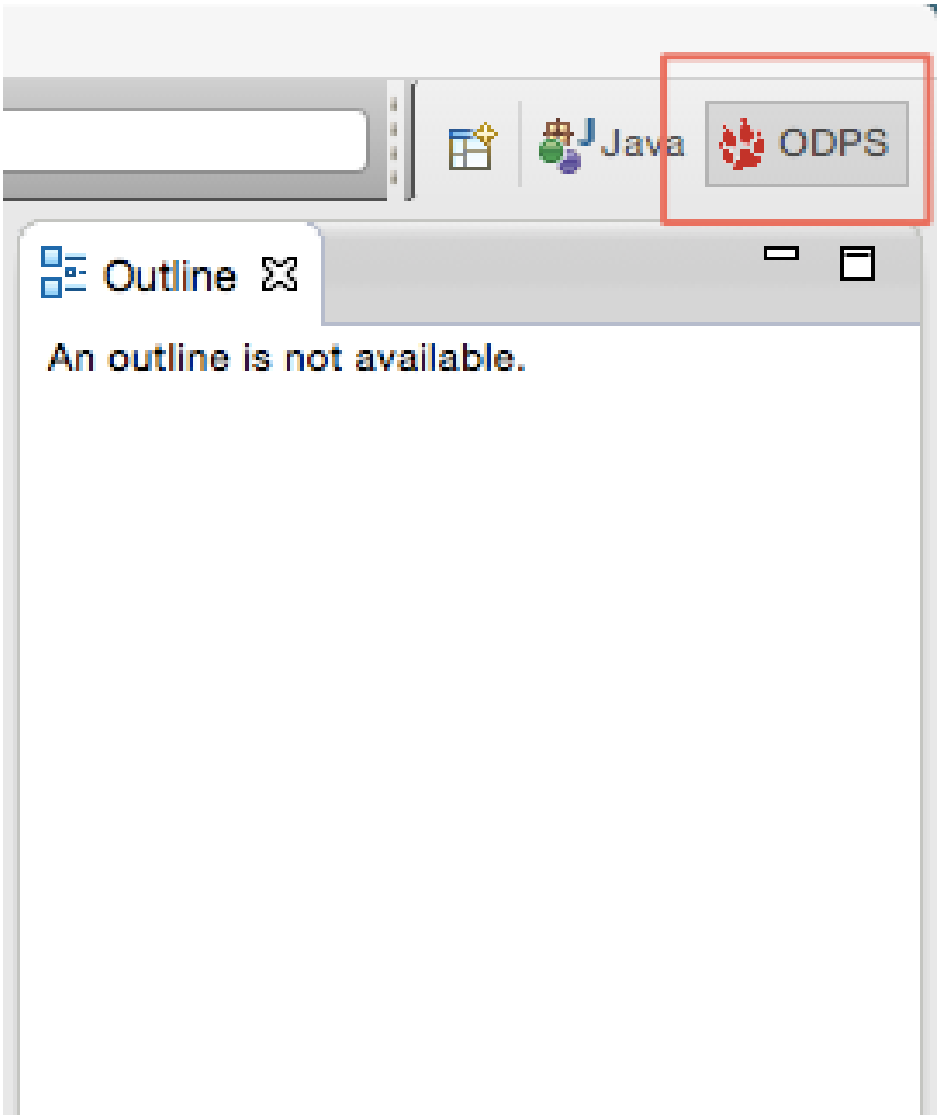
Place the plug-in into the subdirectory **plugins** in Eclipse installation directory. Start the Eclipse plug-in, and click **Open Perspective** in the upper right corner.


Quick Access
Java
Outline
An outline is not available.

After clicking the button, the following dialog box is displayed:



Select **ODPS** and click **OK**. The MaxCompute icon appears in the upper right corner, indicating that the plug-in takes effect.

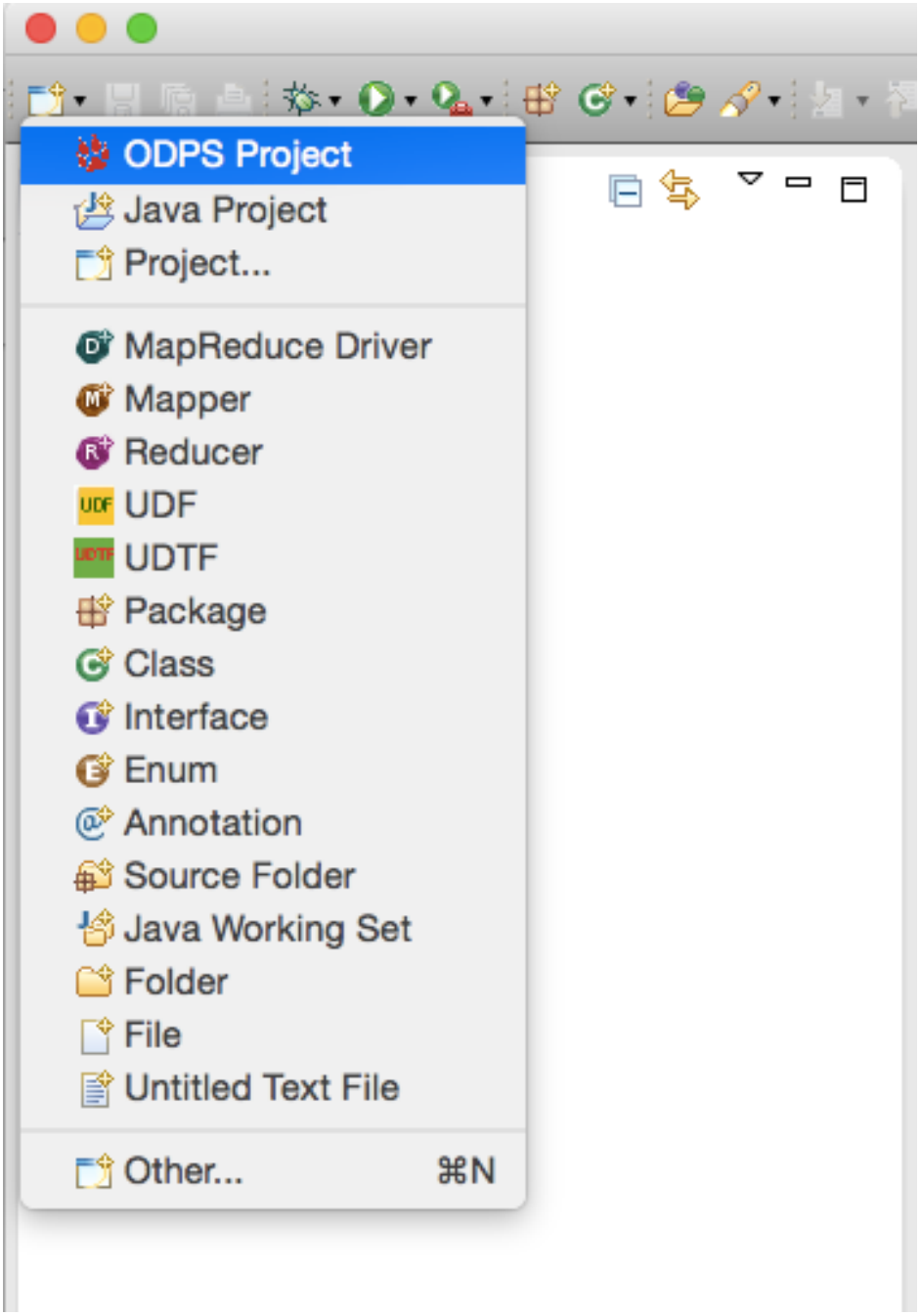


Create Project

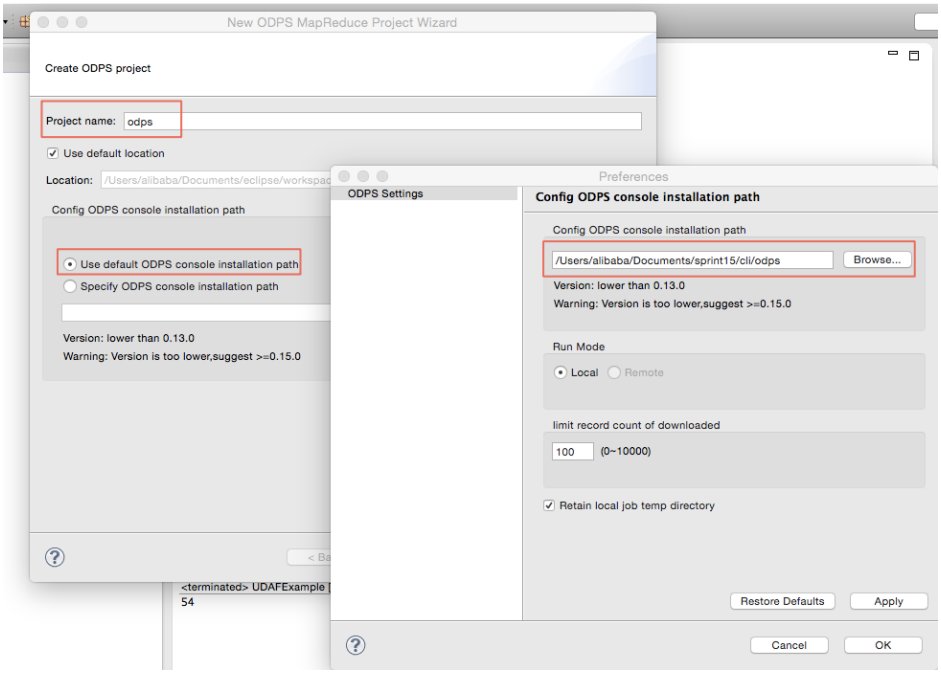
There are two ways to create a MaxCompute project.

Method 1

Select **File** > **New** > **Project** > **MaxCompute** > **MaxCompute Project** to create the project (in the example, use **ODPS** as the project name):



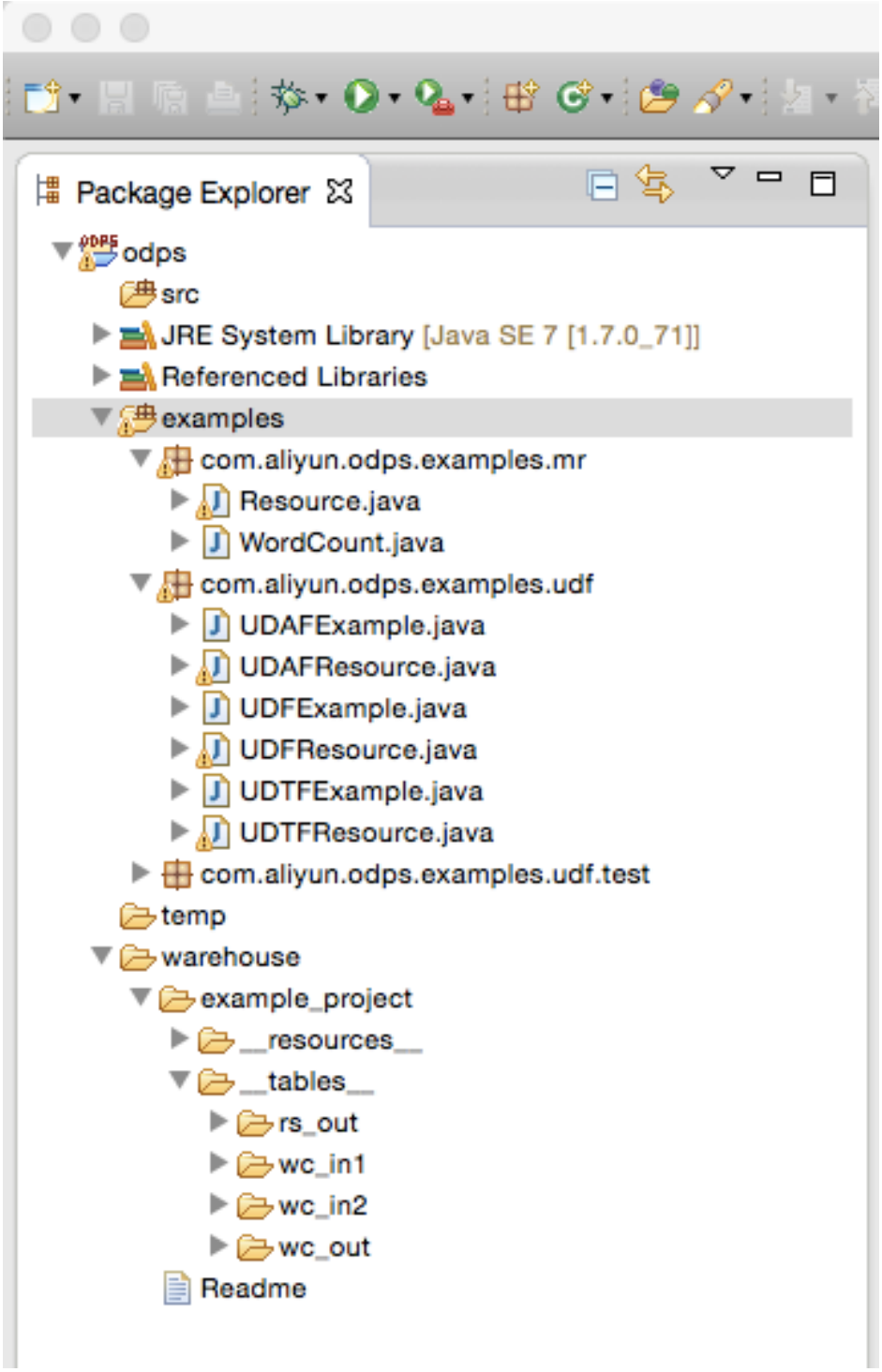
After creating MaxCompute project, the following dialog box is popped up. Input Project name, and select the path of MaxCompute console. (The console must be uploaded first.) Then click **Finish** to confirm.



Note:

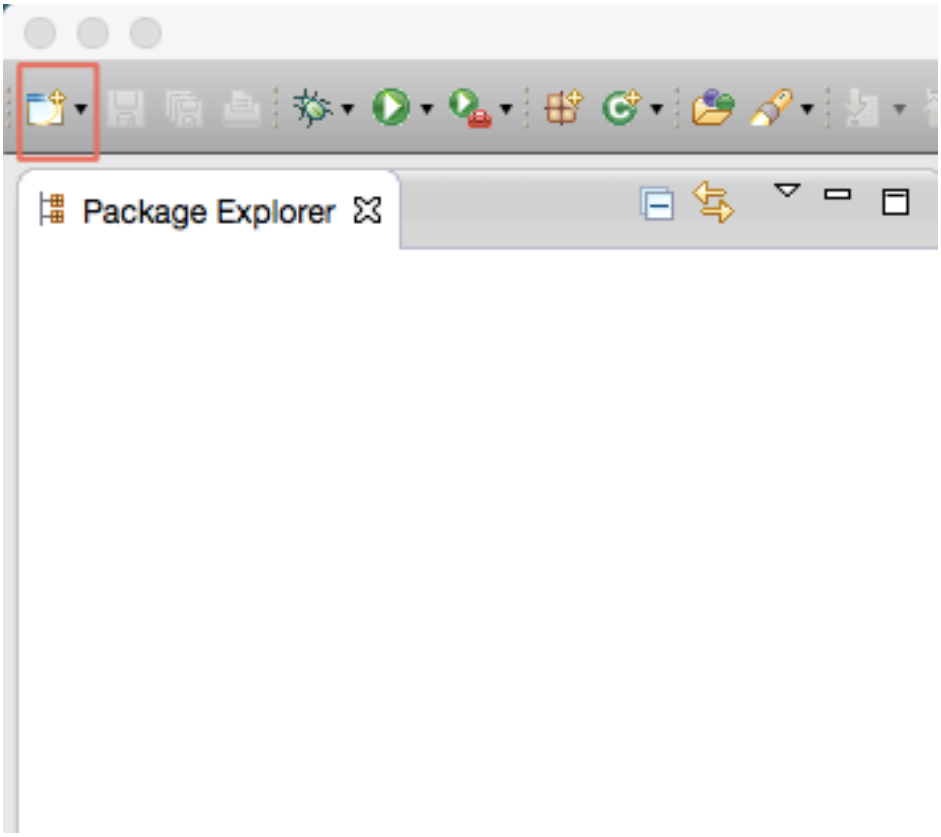
- For the introduction of MaxCompute console, see Console.

After creating the project, the following directory structure is displayed in the left **Package Explorer**:

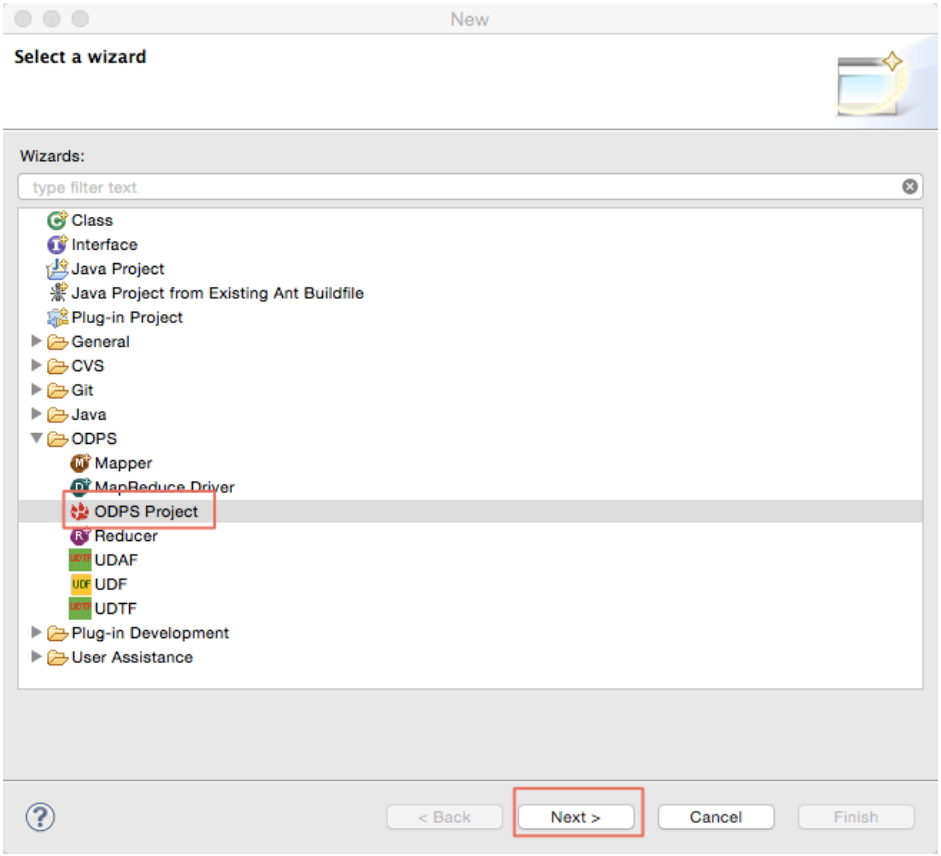


Method 2

Click **New** in the upper left corner:



After the dialog box is popped up, select **ODPS Project** and click **Next**:



The subsequent operations are similar to Method 1.

The installation of MaxCompute Eclipse plug-in is completed. You can use this plug-in to write MapReduce or UDF programs.

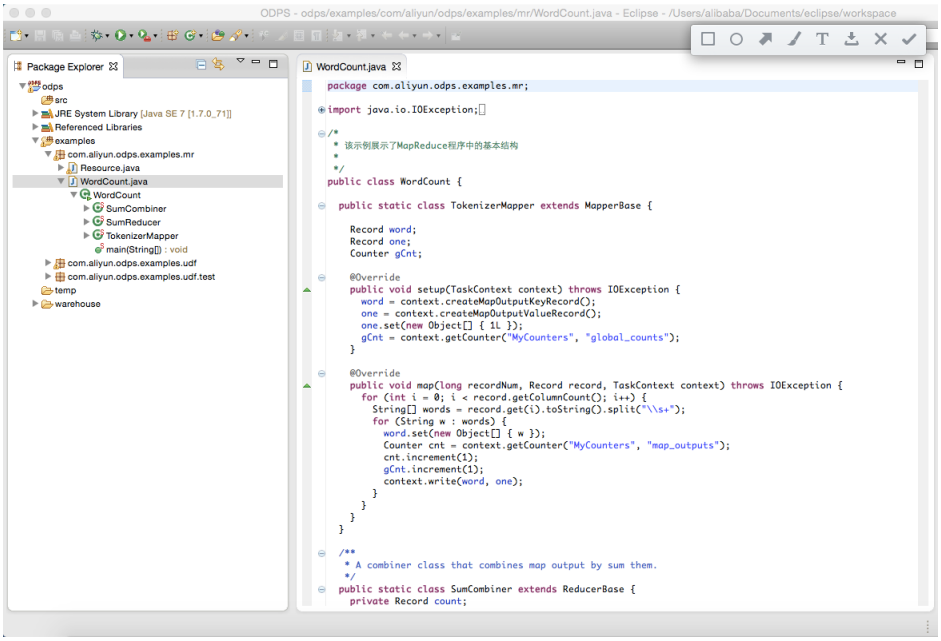
Note:

- For the function introduction of MapReduce in the plug-in, see MapReduce Development Plug-in Introduction.
- For the UDF program example, see UDF Development Plug-in Introduction.

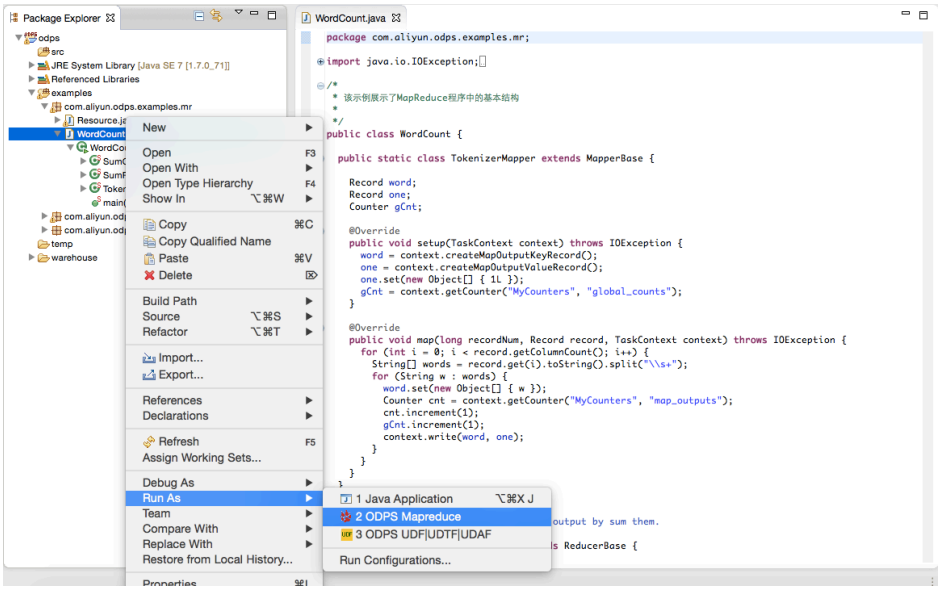
MapReduce

Run WordCount Example Quickly

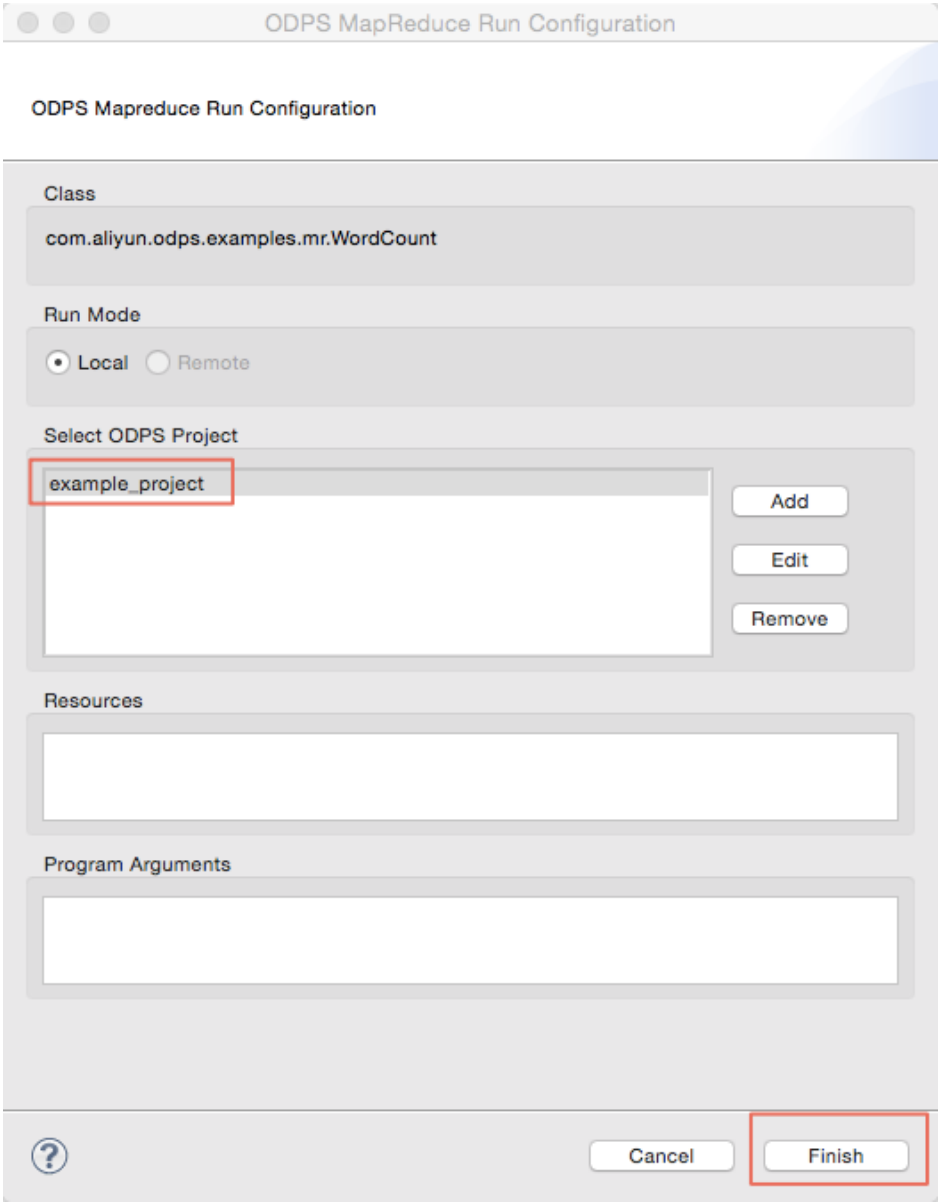
Select **WordCount** example in MaxCompute project:



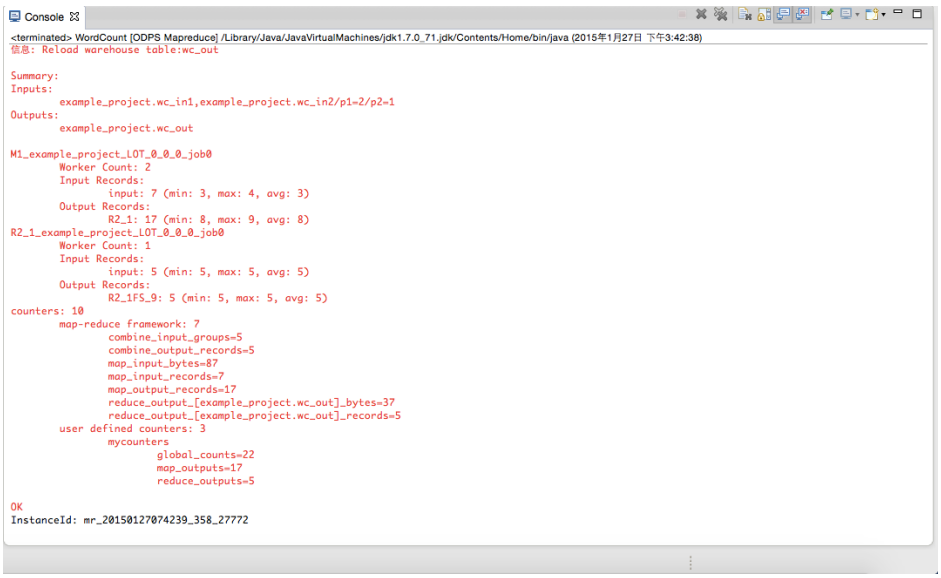
Right-click **WordCount.java** and choose **Run As** -> **ODPS MapReduce**, as follows:



After the dialog box is popped up, select **example_project** and click **Finish**:

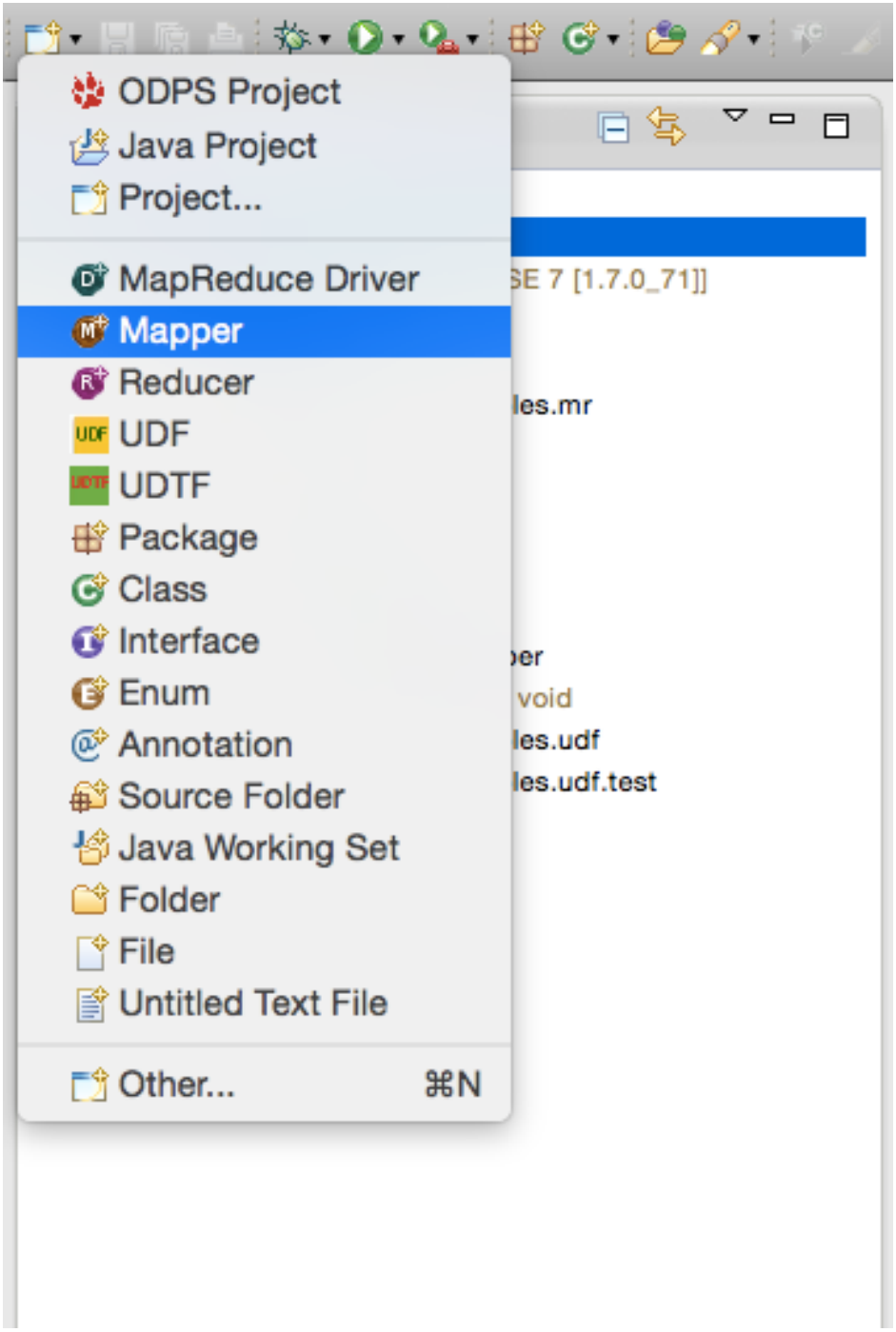


After running is completed, the following result is displayed:

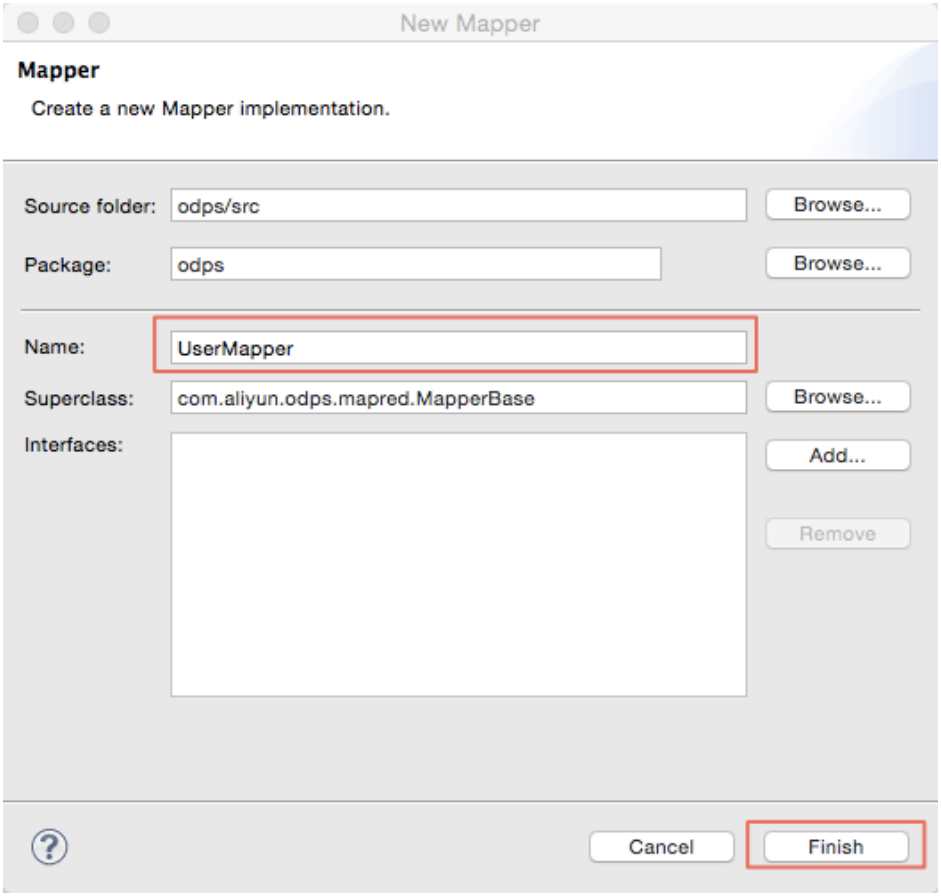


Run User-defined MapReduce Program

Right-click **src** directory. Select **New -> Mapper**:



After selecting **Mapper**, the following dialog box is displayed. Input the name of Mapper class and click **Finish**:



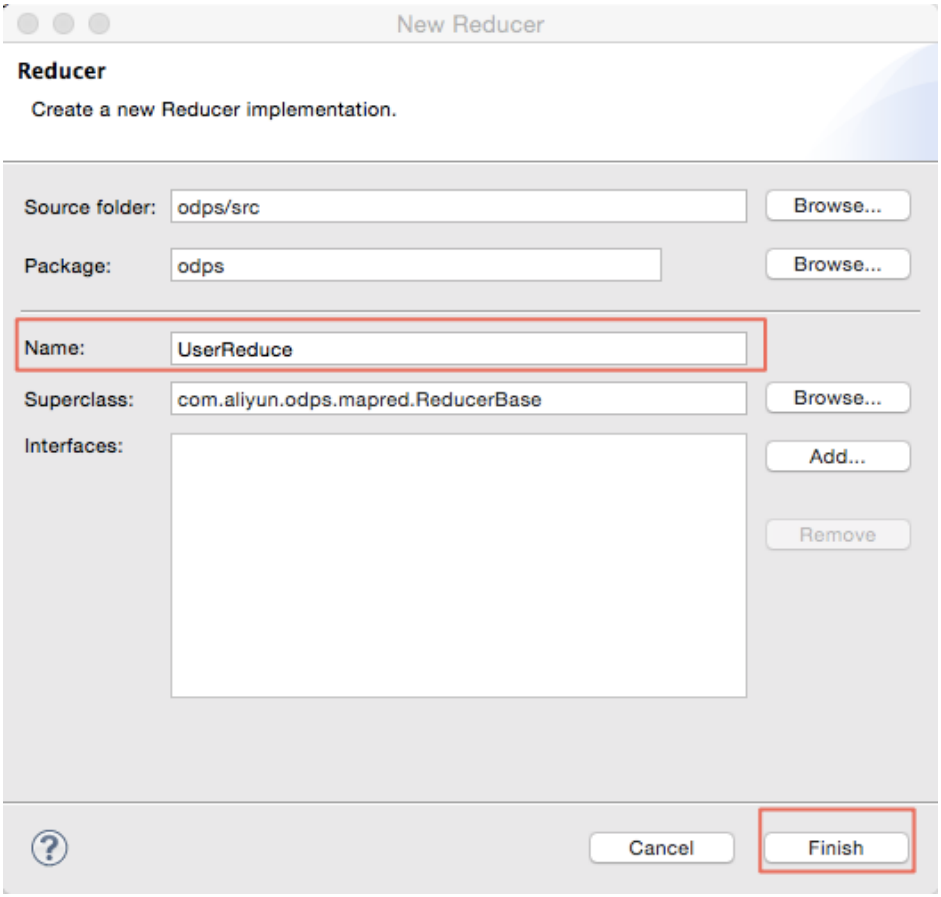
The file **UserMapper.java** is generated in the **src** directory in **Package Explorer**. The content of this file is a template of Mapper class:

```
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 {
}
}
```

In the template, the configured package name defaults to **odps**. You can modify it according to your actual requirement. Write the template content as follows:

```
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 {
}
}
```

Similarly, right-click **src** directory and select **New -> Reduce**:



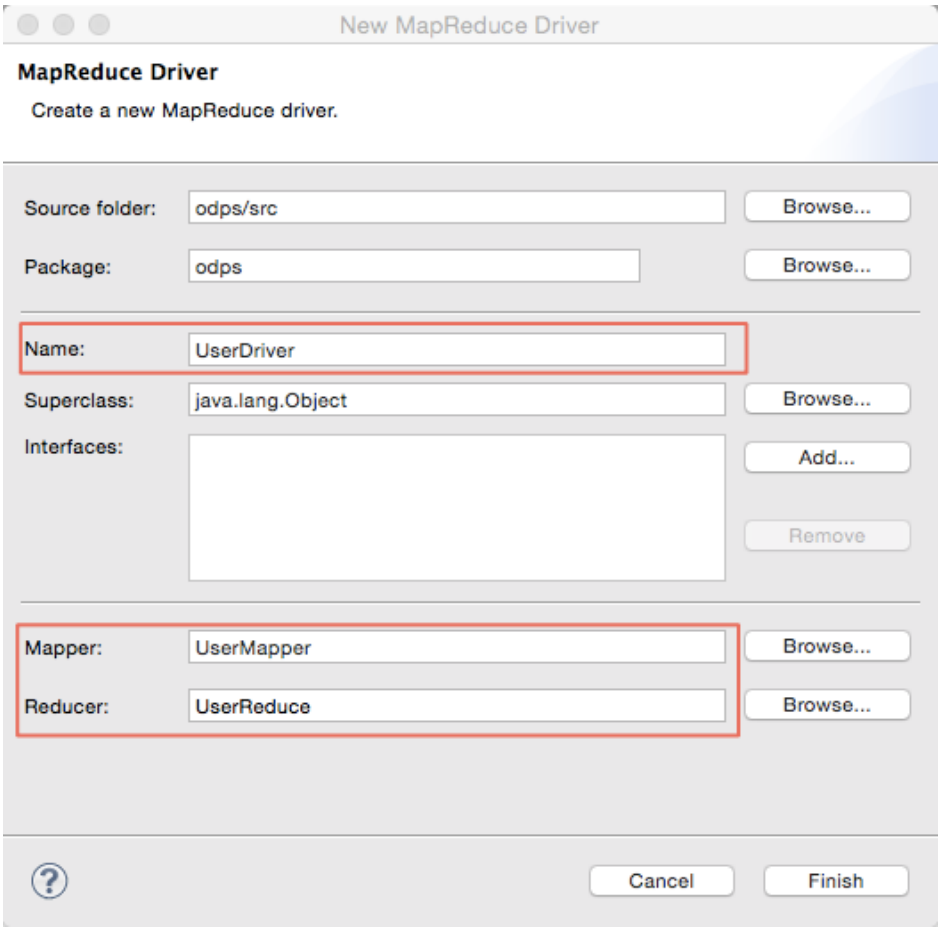
Input the name of Reduce class. (In this example, use **UserReduce** as the class name.)

In **Package Explorer**, a file name **UserReduce.java** is generated in the **src** directory. This file content is a template of Reduce class. Edit the template:

```
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 {
}
}
```

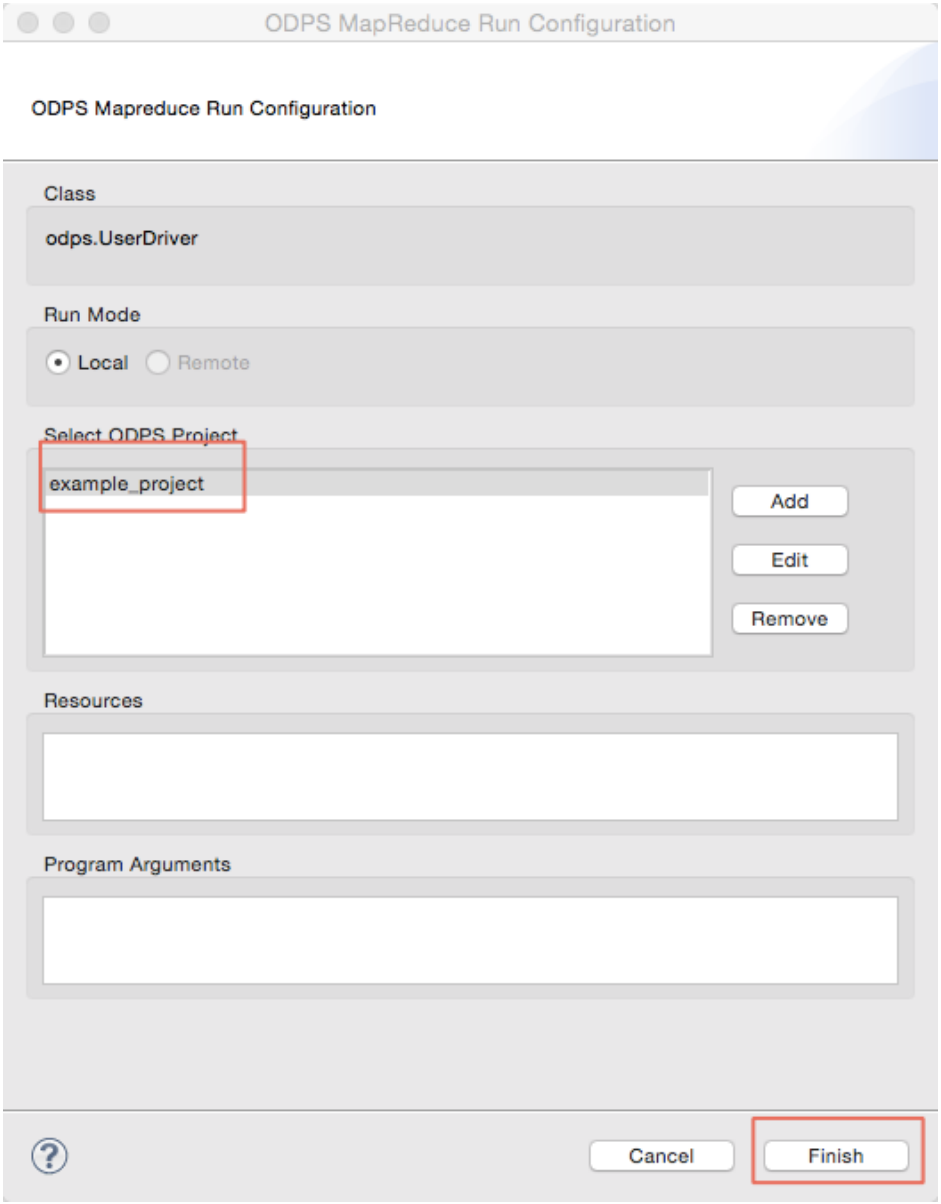
Create **main** function: right-click **src** and select **New -> MapReduce Driver**. Enter Driver Name (in this example, use **UserDriver** as the name), Mapper and Reduce (in this example use **UserMapper** and **UserReduce** as corresponding names) and click **Finish**. The file **MyDriver.java** is also displayed in **src** directory:



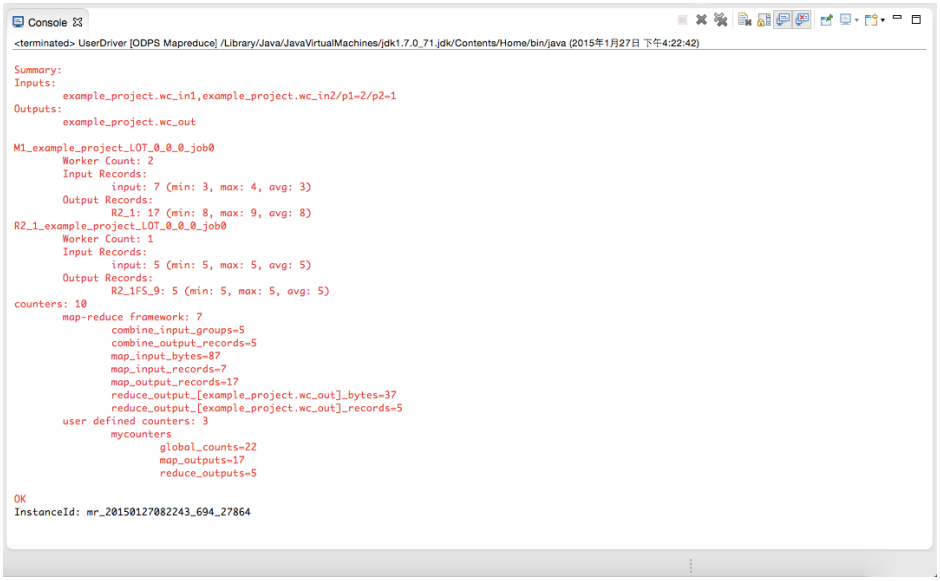
Edit the driver content:

```
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();
}
}
```

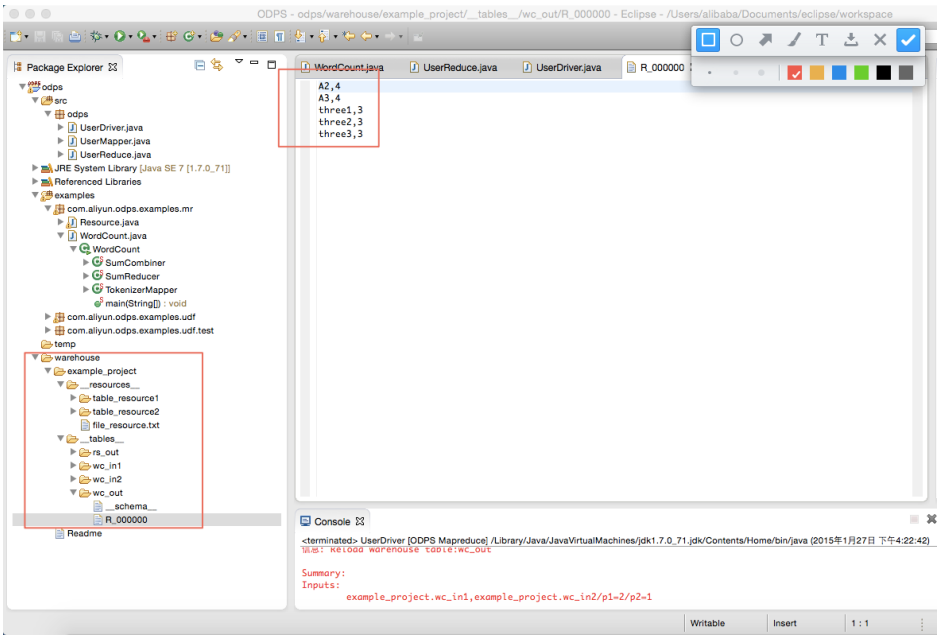
Run MapReduce program. Right-click **UserDriver.java** and select **Run As -> ODPS MapReduce**, the following dialog box is displayed:



Select **example_project** as the MaxCompute Project and click **Finish** to run MapReduce program in the local:



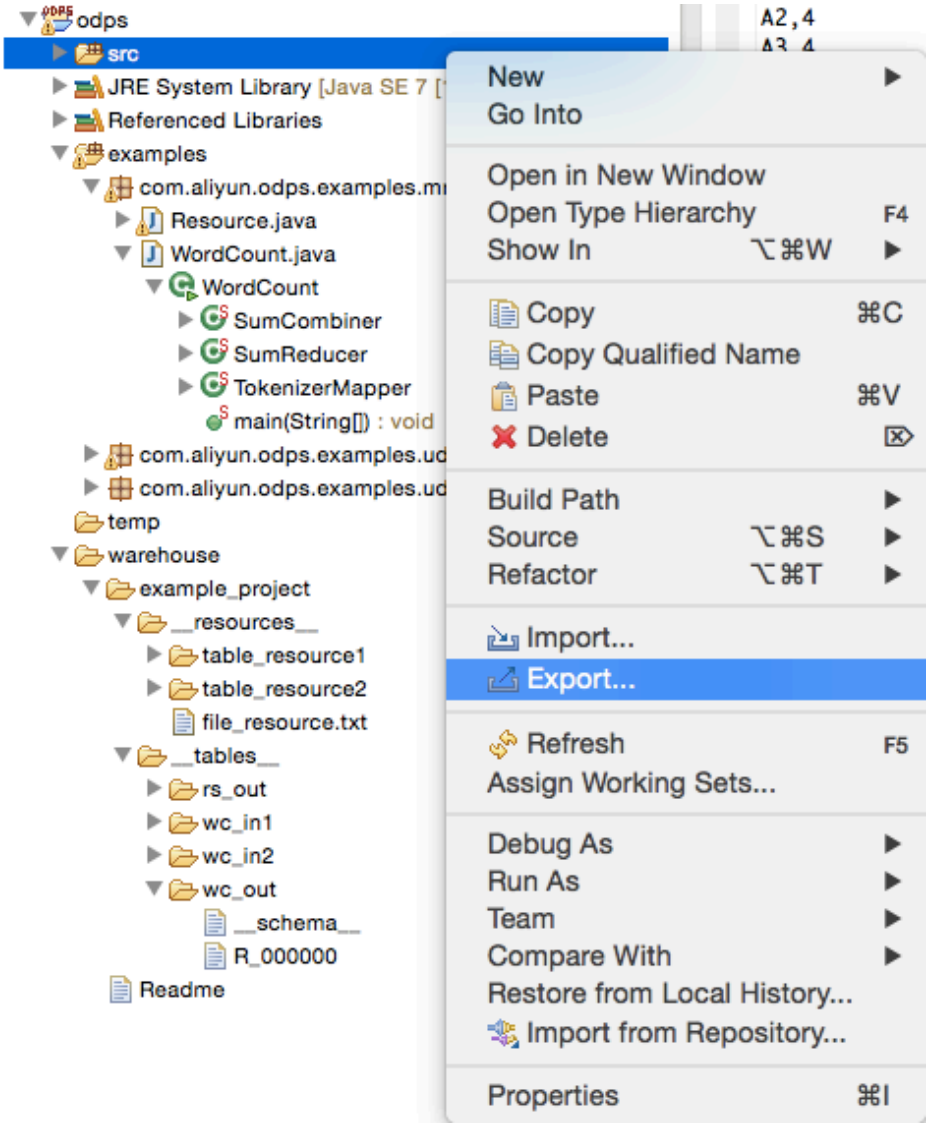
If the output is the same as in the preceding figure, it indicates that local operation runs successfully. The output result is saved in the **warehouse** directory. Refresh MaxCompute project:



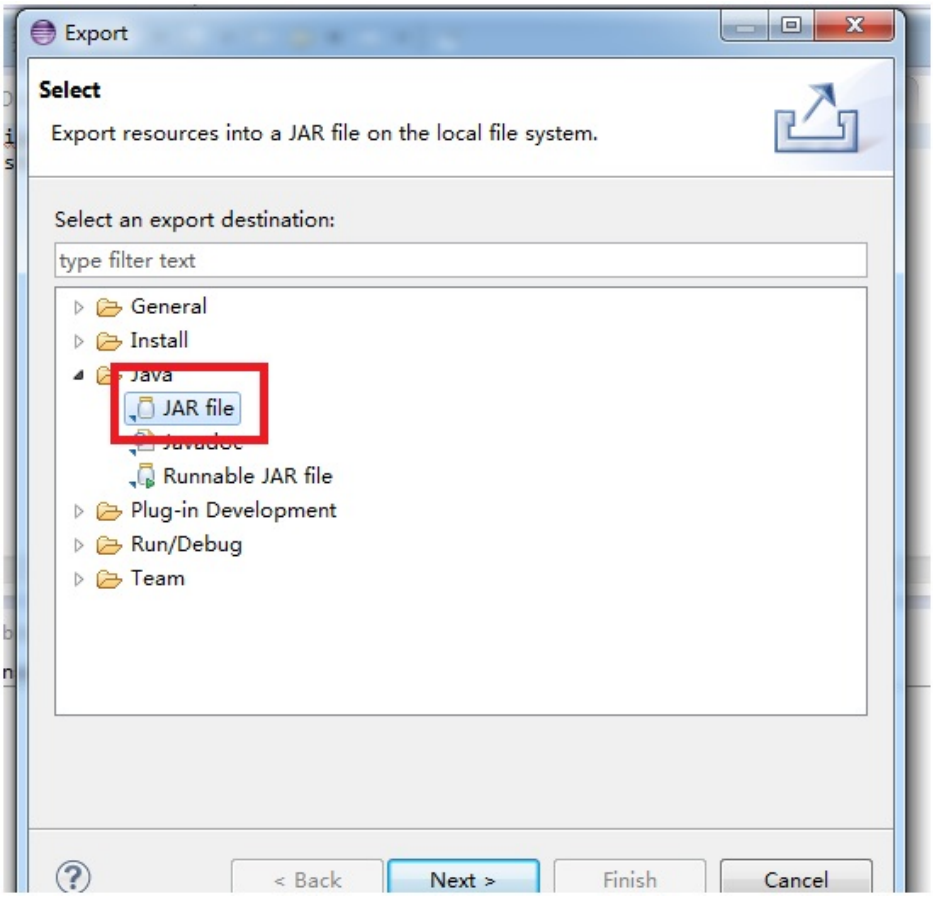
wc_out is the output directory and **R_000000** is the result file. By local debugging, the result is confirmed to be correct and you can package MapReduce program using Eclipse export function. After it is packaged, upload the jar package to MaxCompute. For more information how to run MapReduce in distributed environment, see Quick Start.

After the local debugging is completed, you can package the codes in jar package using Eclipse Export function, provided for subsequent distributed environment. In this example,

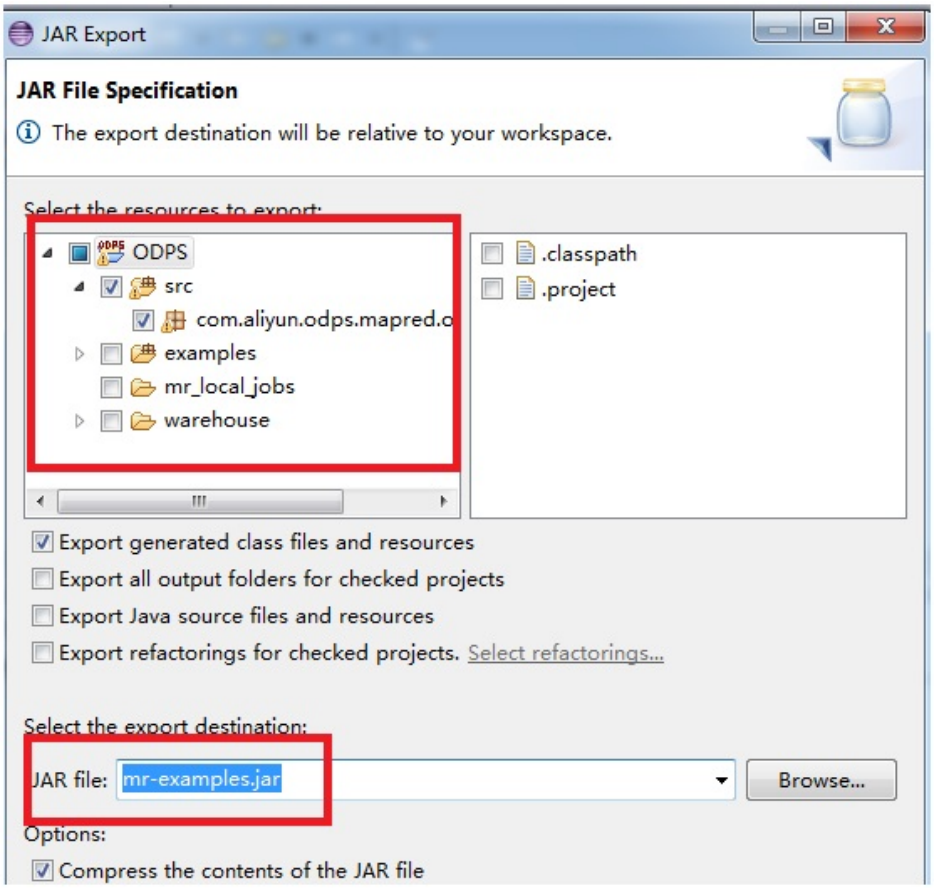
the package name is **mr-examples.jar**. Select the **src** directory and click **Export**:



Select **Jar File** as an export mode:



You must only export the package in **src**. The Jar File name must be specified as **mr-examples.jar**:



Click **Next** to export the jar file.

If you want to simulate new Project creation in the local, you can create a subdirectory (has same level with example_project) in the **warehouse** directory. The directory hierarchy structure is shown as follows:

```
<warehouse>
|___example_project(Project Dirctory)
|___ <__tables__>
| |__table_name1(non-partition table)
| | |___ data(File)
| | |
| | |___ <__schema__> (File)
| |
| |__table_name2(Partition Table)
| |___ partition_name=partition_value(partition directory)
| | |___ data(file)
| |
| |___ <__schema__> (file)
|
|___ <__resources__>
|
|___table_resource_name (table resource)
| |___<__ref__>
|
```

```
|___ file_resource_name(file resource)
```

schema Example:

```
Non-partiton table:
project=project_name
table=table_name
columns=col1:BIGINT,col2:DOUBLE,col3:BOOLEAN,col4:DATETIME,col5:STRING
Partition table:
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
Note:
Currently, the following five data formats are supported: bigint,double,boolean,datetime,string, which correspond to the data types in java: -long,double,boolean,java.util.Date,java.lang.String.
```

data Example:

```
1,1.1,true,2015-06-04 11:22:42 896,hello world
\N,\N,\N,\N,\N
Note:
The time format is accurate to the millisecond level and all types are represented NULL by '\N'.
```

Note:

- If MapReduce program runs in the local, the default is to search corresponding tables or resources from the **warehouse** directory. If the tables or resources do not exist, corresponding data will be downloaded from the server and saved in **warehouse**. Then run MapReduce in the local.
- After running MapReduce is finished, refresh the **warehouse** directory to view the generated result.

UDF

Local Debug UDF Program

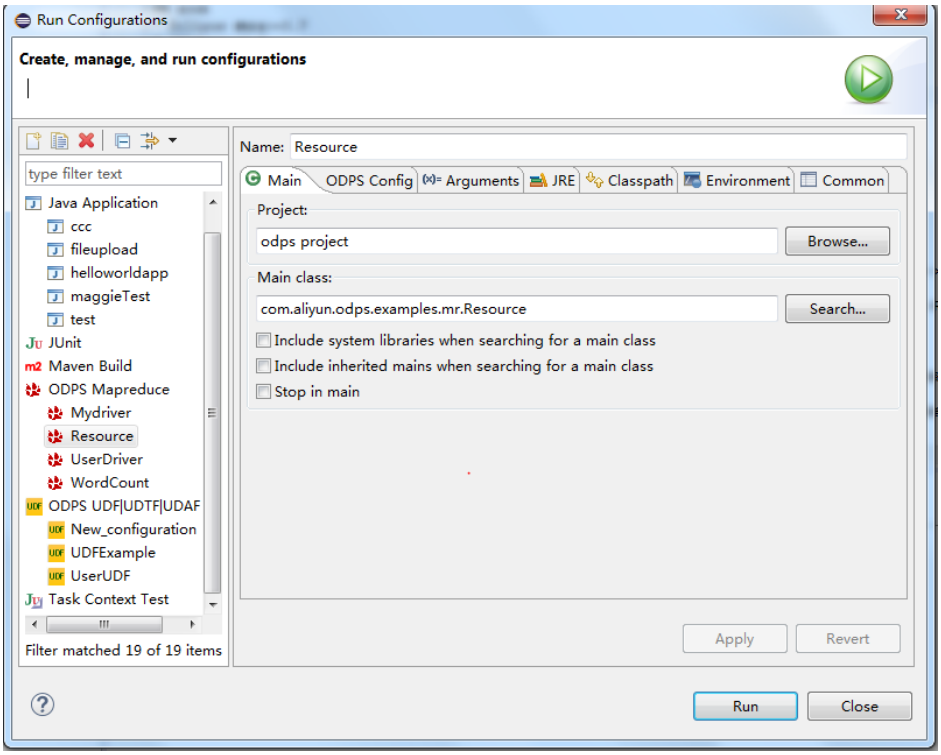
This section describes how to develop UDF with the Eclipse plug-in and how to run UDF on local. The preparation and implement process is similar to UDF. You can see the example of UDF.

MaxCompute Eclipse plug-in provides two methods to run UDF: Menu Bar and run by right-clicking

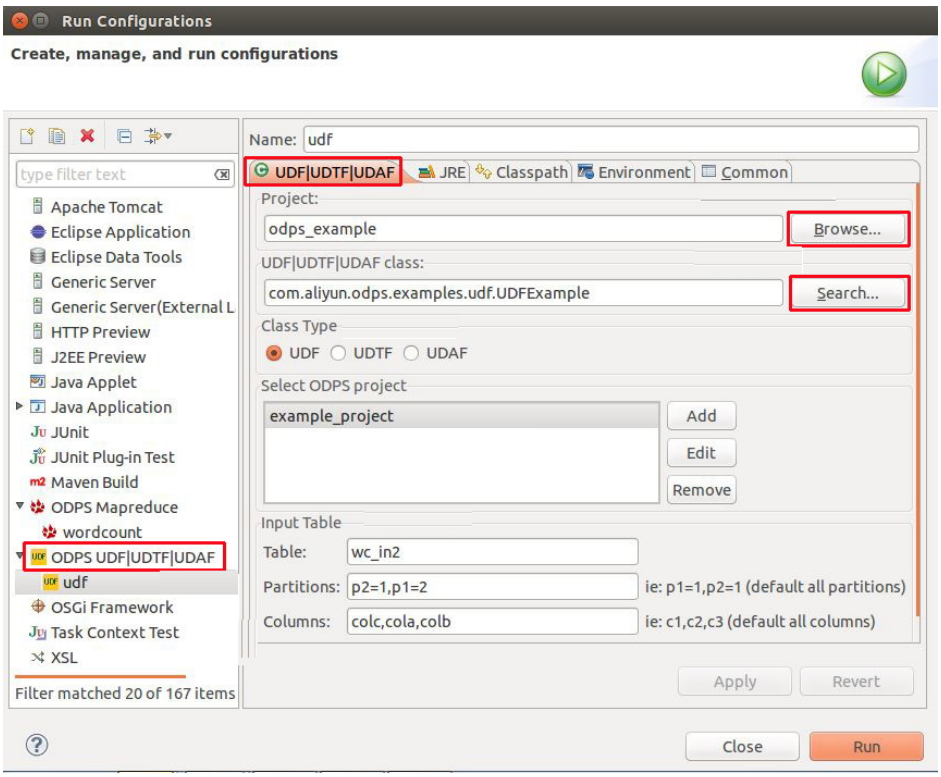
it.

Run UDF using Menu Bar

Select **Run > Run Configurations...** from the menu bar and the following dialog box appears:



You can create a new Run Configuration. Select the UDF class and type to be executed, select MaxCompute Project and enter the information of input table. For example:

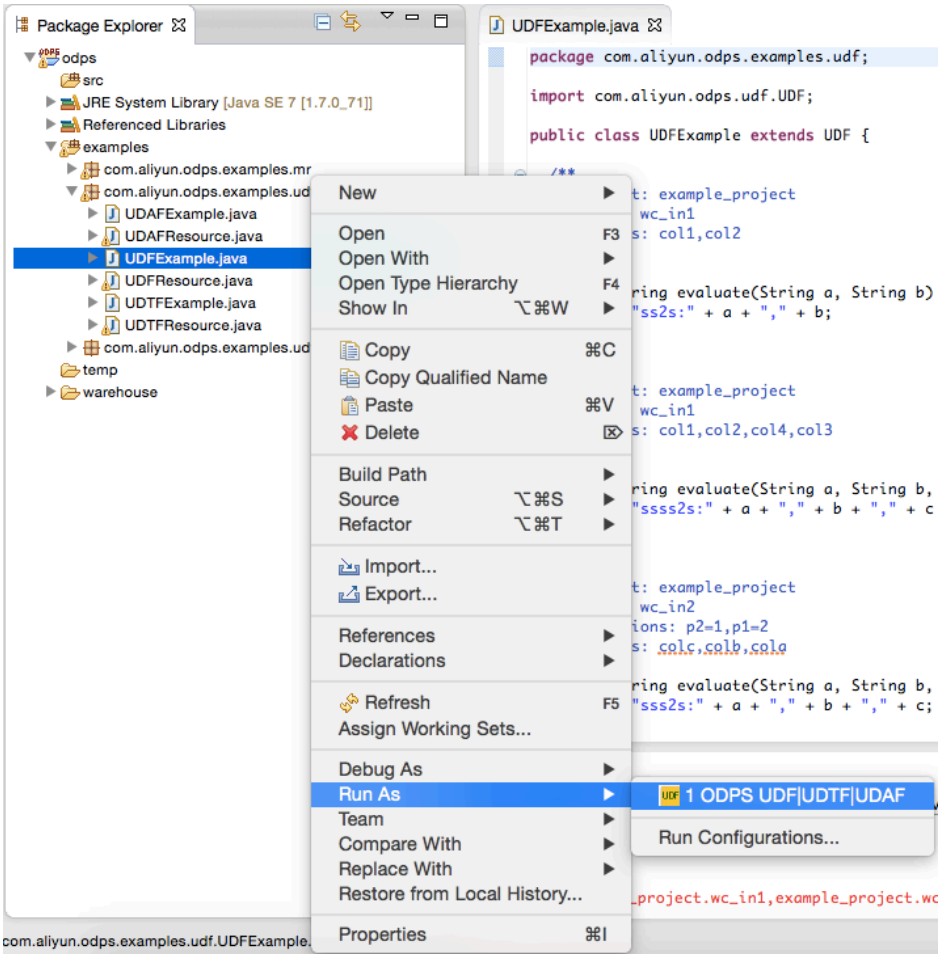


In the preceding configuration, **Table** indicates the input table of UDF. **Partitions** indicates the partitions from which the data is read, separated by commas. **Columns** indicates the columns, which are considered as the parameters of UDF to be introduced. The columns are separated by commas.

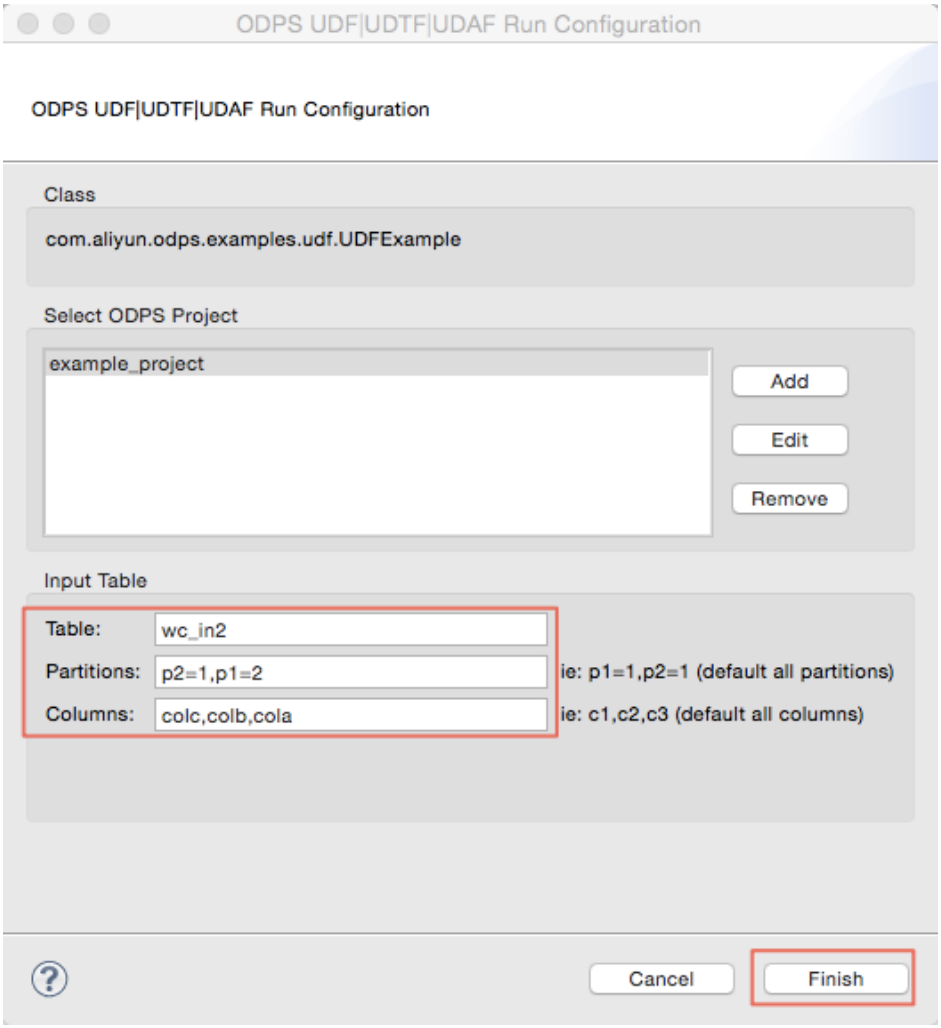
Click **Run** to run the program and the running result is displayed in the console:

Run by right-clicking

Select a udf.java file (such as UDFExample.java) and right-click it. Then select **Run As > Run UDF|UDAF|UDTF**:



The configuration information is shown as follows:



In the preceding configuration, **Table** indicates the input table of UDF. **Partitions** indicates the partitions from which the data is read, separated by commas. **Columns** indicates the columns, which are considered as the parameters of UDF to be introduced. The columns are separated by commas.

Click **Finish** to run UDF and get the output result.

Running customized UDF program

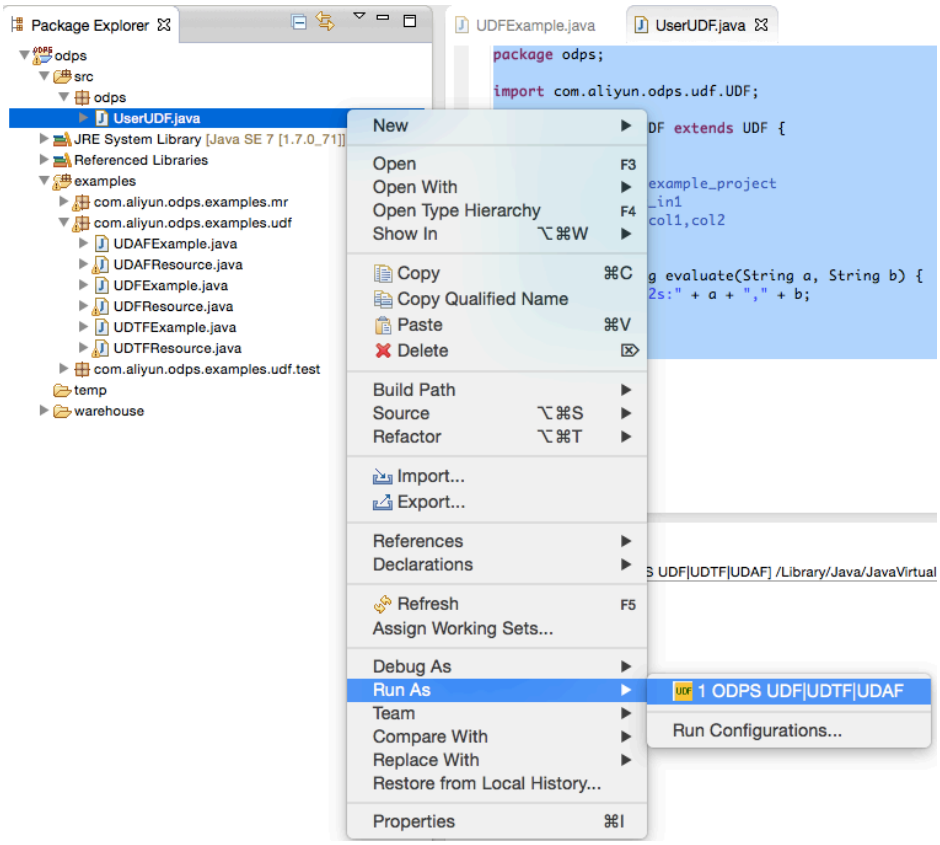
Right-click a project and select **New >UDF** (or select the menu bar **File > New > UDF**).

Enter the UDF class name and click **Finish**. Generate a Java file in corresponding **src** directory with the same name as this UDF class. Edit this java file as follows:

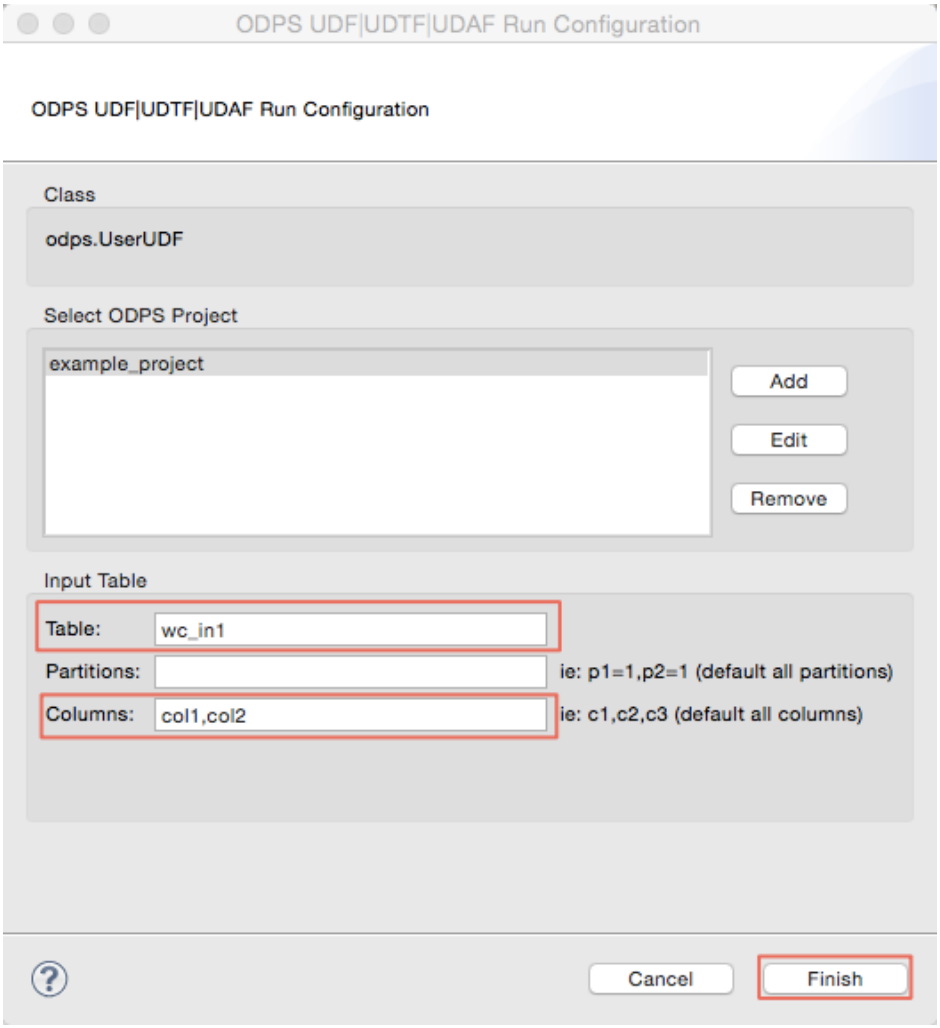
```
package odps;

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;
}

}
```

Right-click this java file (such as UserUDF.java) and select **Run As -> ODPS UDF|UDTF|UDAF**:



Configure the following dialog box:



Click **finish** to get the result:

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

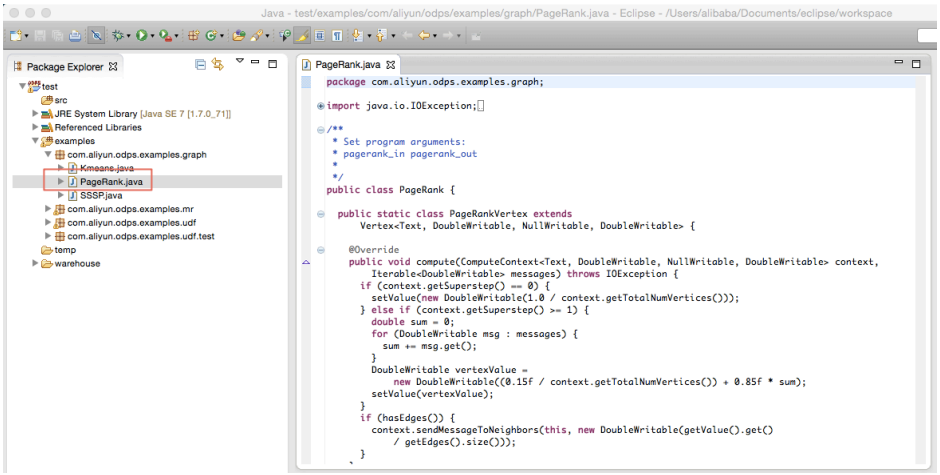
Only the operation instance of UDF is described in this section, and the way of UDTF operating is basically similar to the UDF.

Detailed introduction

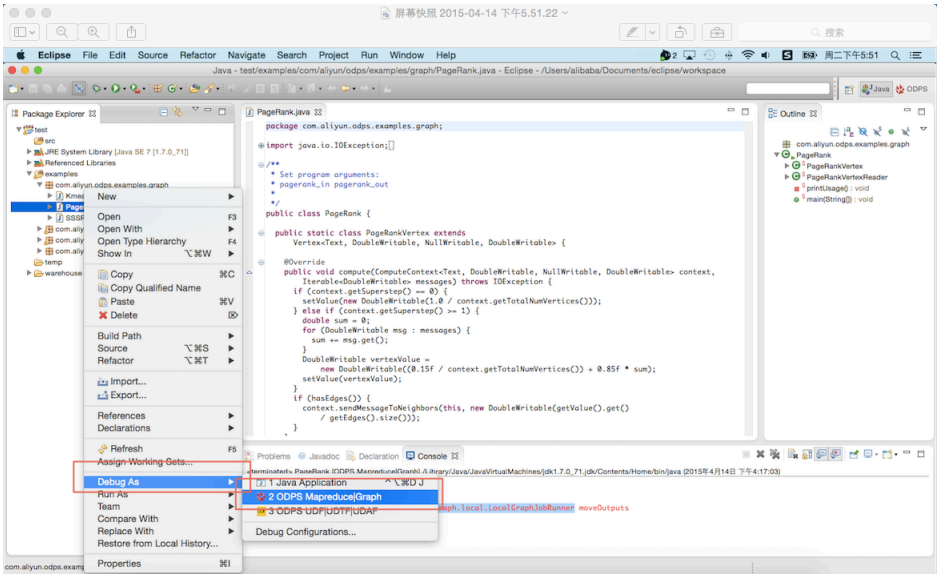
After creating a MaxCompute project, you can write Graph program and complete the local debugging according to the following steps.

In this example, **PageRank.java** provided by the plug-in is selected to complete the debugging.

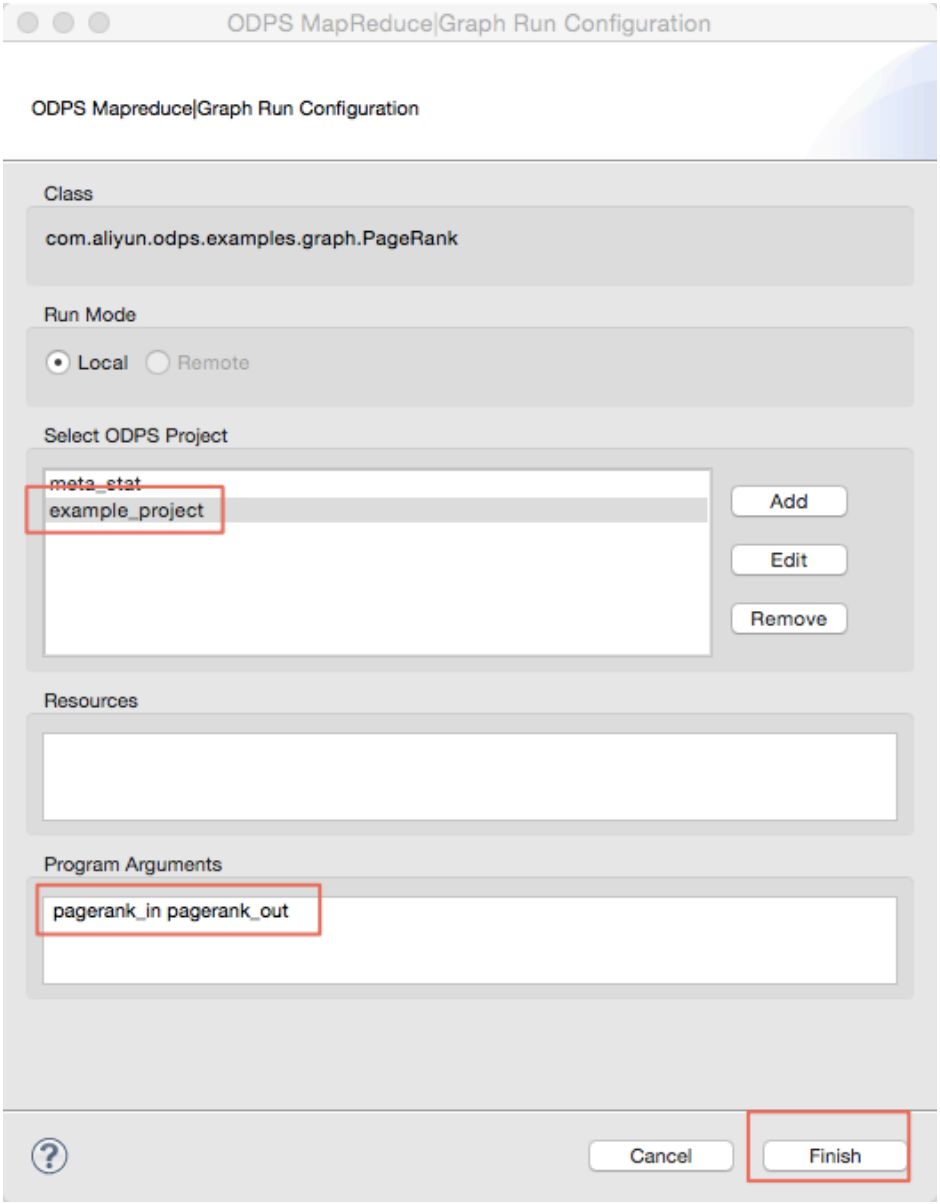
Select **PageRank.java** in **examples**:



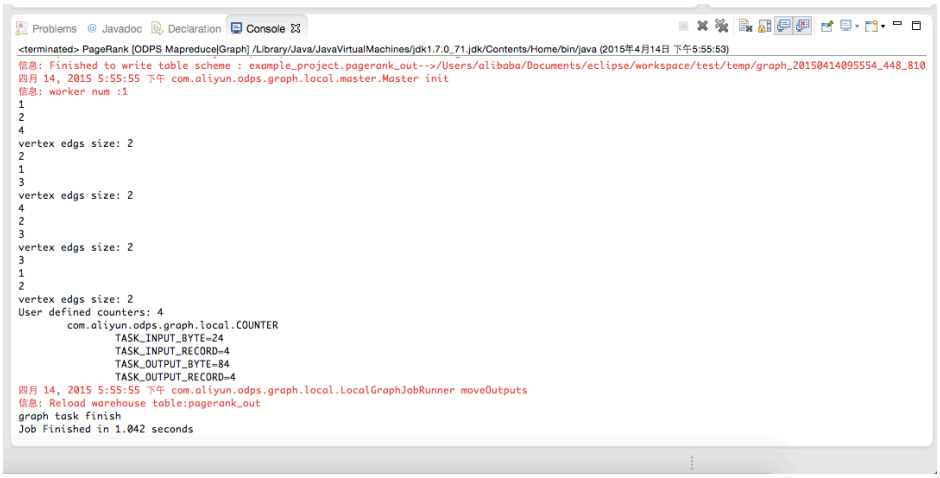
Right-click and select **Debug As -> ODPS MapReduce|Graph**:



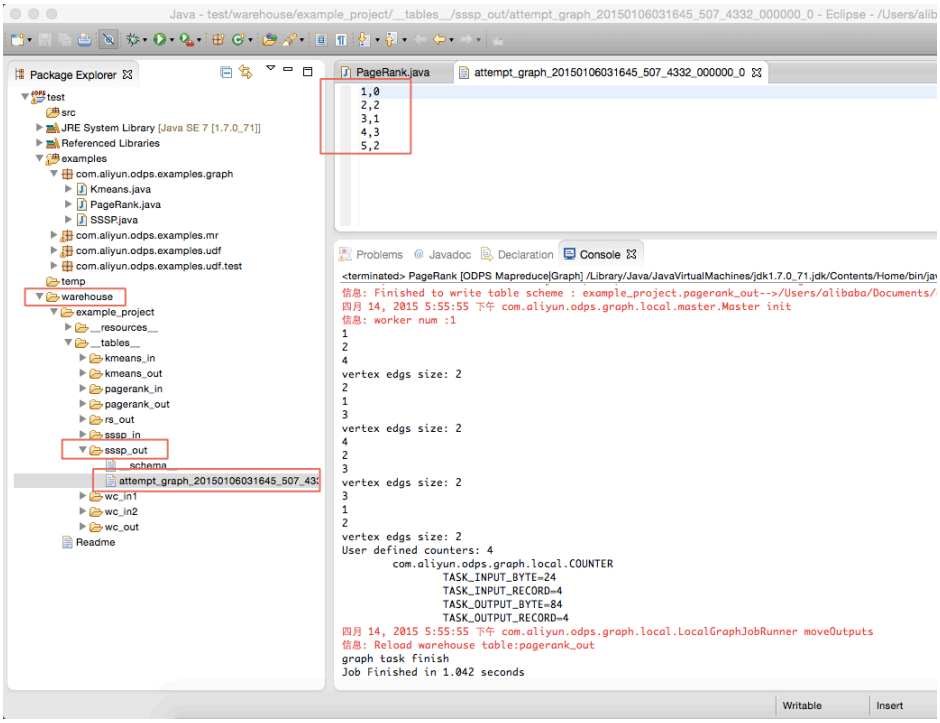
The dialog box appears, and configure it as follows:



View the running result:



You can view the computing result on the local:



After the debugging is complete, you can package the program and upload it to MaxCompute as a Jar resource. Then submit Graph job.

Note:

- For the package process, see MapReduce Eclipse Plug-in Introduction.
- For the structure introduction of local result, see MapReduce Eclipse Plug-in Introduction.
- For the detailed introduction of uploading Jar resource, see Add Resource in Basic Introduction.
- For submitting the Graph job, see Graph Function.

Downloads

- SDK Downloads: Maven users can search **odps-sdk** from Maven library to get different versions of the Java SDK.
- MaxCompute console
- Eclipse plugin
- Intelij plugin

- Studio