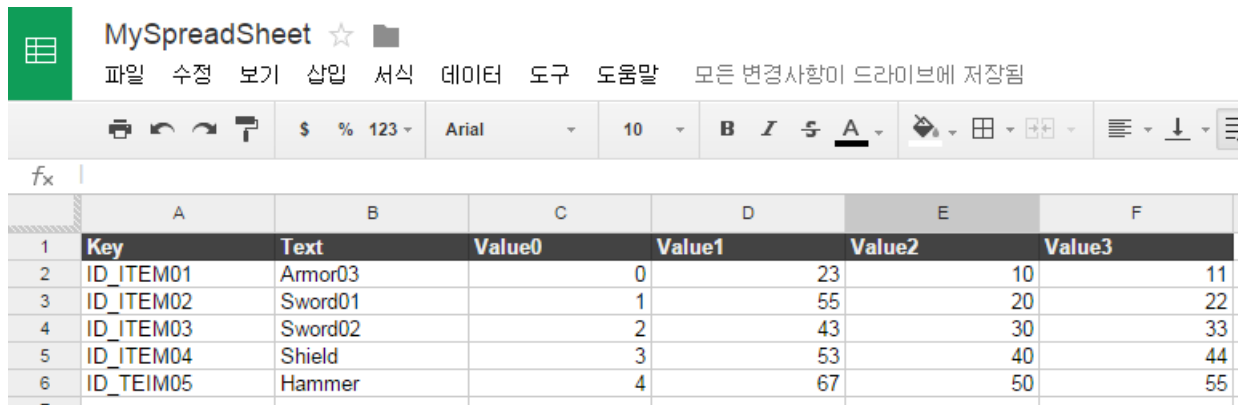


## How to work with Google Spreadsheet

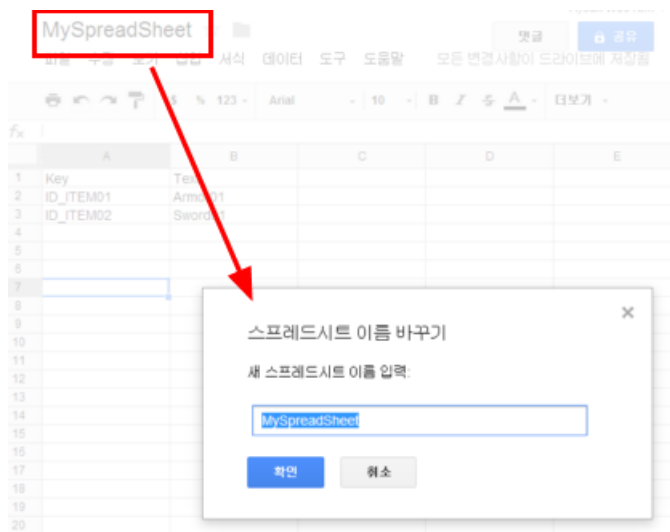
This document shows and helps you how to set up and use [Unity-QuickSheet \(https://github.com/kimsama/Unity-QuickSheet\)](https://github.com/kimsama/Unity-QuickSheet) with Google Spreadsheet.



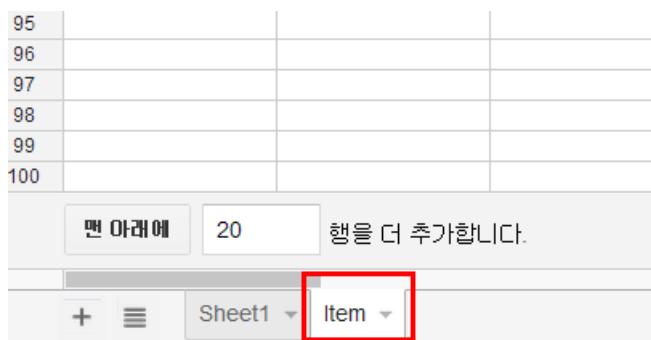
	A	B	C	D	E	F
1	Key	Text	Value0	Value1	Value2	Value3
2	ID_ITEM01	Armor03	0	23	10	11
3	ID_ITEM02	Sword01	1	55	20	22
4	ID_ITEM03	Sword02	2	43	30	33
5	ID_ITEM04	Shield	3	53	40	44
6	ID_ITEM05	Hammer	4	67	50	55

First, you need to create a google spreadsheet on your Google Drive. Login your Google Drive with your google account and create a new spreadsheet.

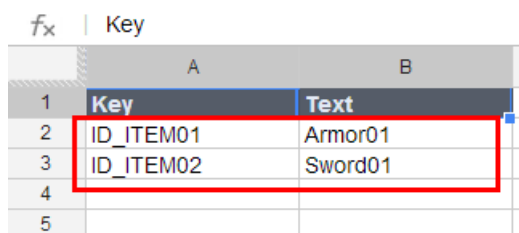
Change the title of the created spreadsheet as 'MySpreadSheet' like the following:



Next, create a new worksheet and rename it to whatever you want to as the following image shows:



Now, it needs to edit cells for spreadsheet. Insert 'Key' and 'Text' at the first row of the created worksheet as like that:

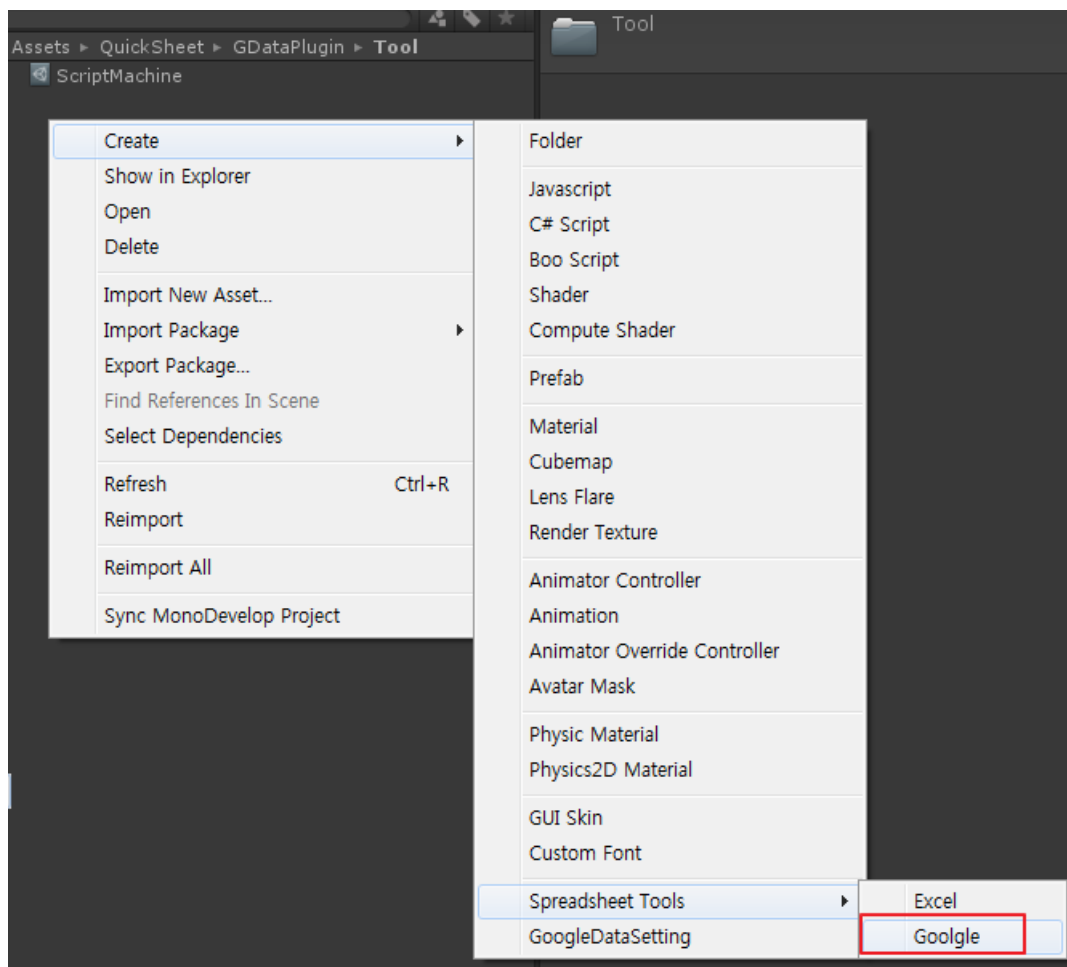


	A	B
1	Key	Text
2	ID_ITEM01	Armor01
3	ID_ITEM02	Sword01
4		
5		

**IMPORTANT** Note that the first row should not contain any values which are used for your class members.

### Step 1) Creating Google Spreadsheet Setting File

First you need thing to do is creating a google spreadsheet setting file. Simply right click on the Project view and select 'Create > Spreadsheet Tools > Google'. It creates a new file which shows various setting to create script files and get data from the specified google spreadsheet.



Select **Google** menu item then it creates setting file. It may be shown like the following:

**GoogleDrive Settings:**

Username:

Password:

**Script Path Settings:**

Template Path:

Runtime Class Path:

Editor Class Path:

Spread Sheet Name:

Work Sheet Name:

Only DataClass: ☐

#### GoogleDrive Setting

1. Specify your google account.
2. Specify your password for the account.

#### Script Path Setting

3. **Template** indicates a path where template files which are necessary to generate script files. In most case you don't need to change it.
4. **Runtime** indicates a path where generated script files which are used on runtime will be put.
5. **Editor** indicates a path where generated script files which are used on editor mode will be put.
6. **Spread Sheet Name** is what the name of the spreadsheet which is created on google drive.
7. **Work Sheet Name** is one of the worksheet name of the spreadsheet you want to get data from.

After doing done with all path setting, press **Import** button then it shows all column headers of the page. That are necessary to let you set the type of the each cells.

**Type Settings:**

Key	<input type="text" value="String"/>
Text	<input type="text" value="String"/>
Value0	<input type="text" value="Int"/>
Value1	<input type="text" value="Int"/>
Value2	<input type="text" value="Int"/>
Value3	<input type="text" value="Int"/>

Set the proper type of the cells.

Currently the following types are supported:

- string
- int
- float
- double
- enum
- bool

## Step 2) Generating Script Files

If you've done all necessary setting, it's time to generate some script files which are needed for reading data in from the sheet page of the excel file and to store that within *ScriptableObject* which is being as an asset file in the *Project View*.

Press **Generate** button.

After generating some script files, Unity Editor starts to compile those. Wait till Unity ends doing compile then check the specified *Editor* and *Runtime* paths all necessary script files are correctly generated.

In **Editor** folder should have contain two files:

- *your-sheetpage-name*AssetCreator.cs
- *your-sheetpage-name*Editor.cs

In **Runtime** folder should have contain two files:

- *your-sheetpag-name*.cs
- *your-sheetpage-name*Data.cs

See the *your-sheetpage-name*Data.cs file. The class members of the file represent each cells of the sheet page.

```
using UnityEngine;
using System.Collections;

///
/// !!! Machine generated code !!!
/// !!! DO NOT CHANGE Tabs to Spaces !!!
///
[System.Serializable]
public class PlayerItemData
{
    [SerializeField]
    string key;

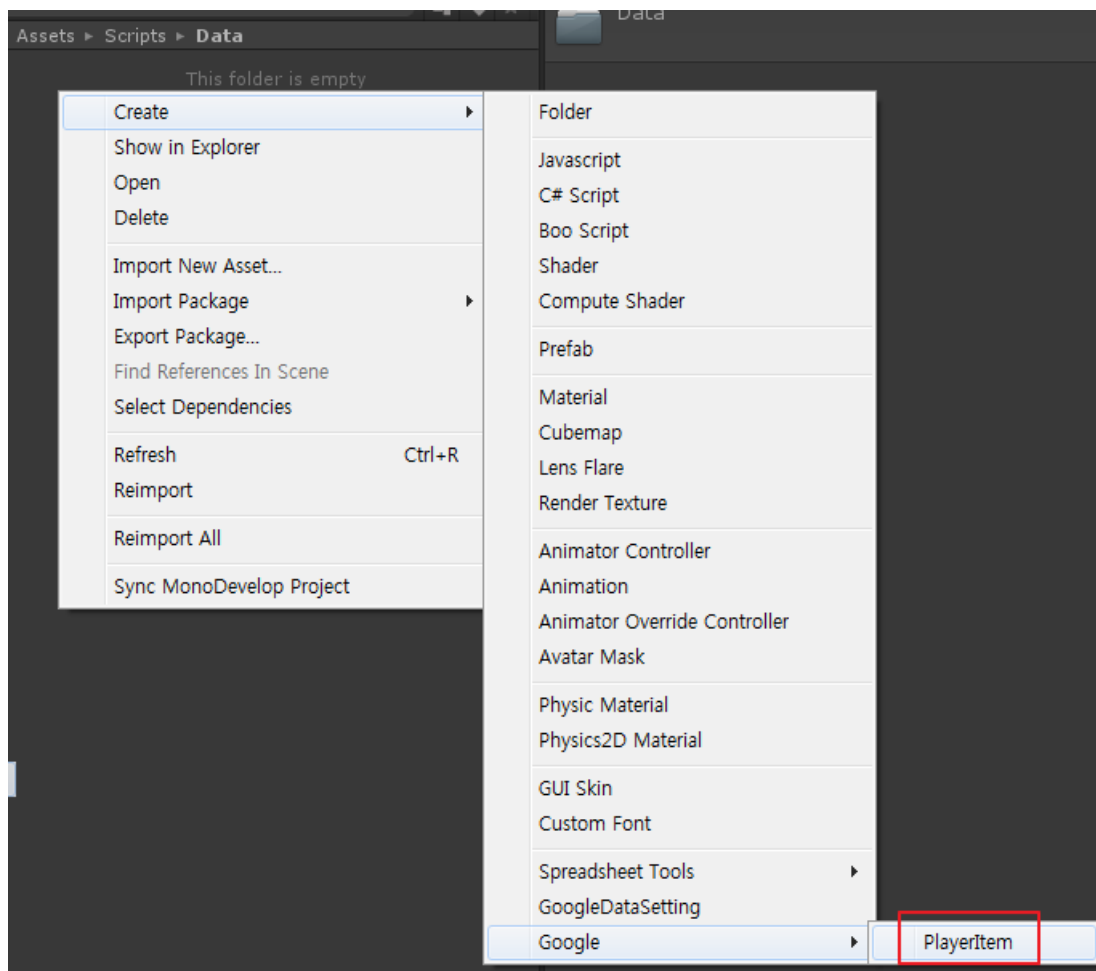
    [ExposeProperty]
    public string Key { get {return key; } set { key = value;} }

    [SerializeField]
    string text;

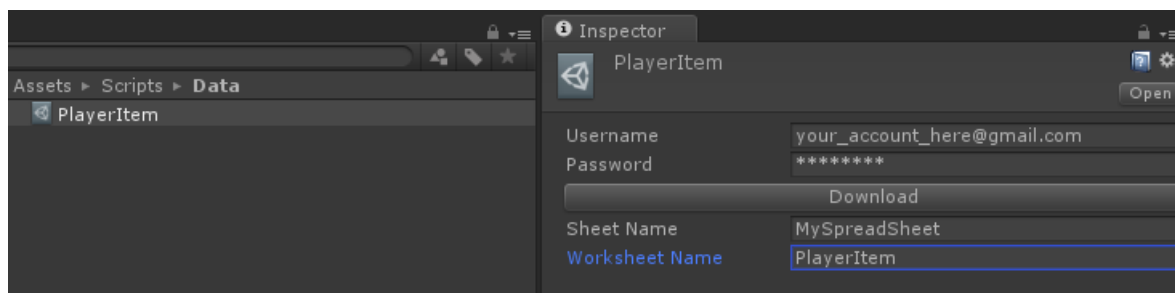
    [ExposeProperty]
    public string Text { get {return text; } set { text = value;} }
```

## Step 3) Importing Spreadsheet Data

Now you need to create an asset file which will imports and stores all data from google drive. Simply right click on the Project view and select 'Create > Google'. Now there is a new menu item which has same name with the worksheet name of the google spreadsheet.



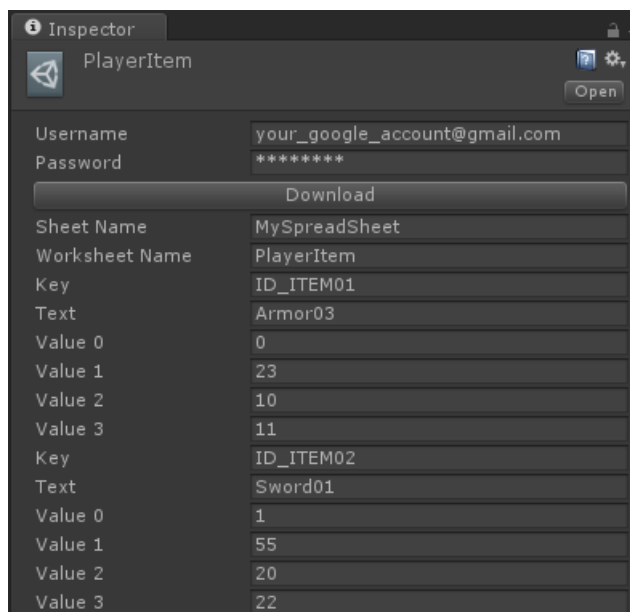
Select the menu item then it creates an asset file. It may be shown like the following:



Note that the created asset file has same file name as the specified worksheet name.

1. Type **Username** and **Password** of your google account.
2. Specify the same Spreadsheet and Worksheet name as the google setting.
3. Press **Download** button then it starts to import data from google drive. (It may takes a few seconds.)

Finished downloading shows the imported data on the Inspector View like the following:



It's done. Hope you enjoy that!

## TroubleShoting

### Invalid Credential Error

If you met an error which is shown as an invalid credentials when you try to get data by clicking 'download' button, check that your google accout page and you have two-stage verification.

If you have Google two-stage verification on, then it doesn't matter what your Google password is, it won't be accepted. You need to generate (on Google) what is called an Application Specific Password (ASP). Go to [Google Account Page \(https://www.google.com/settings/account\)](https://www.google.com/settings/account) and set up an ASP, enter the password you generate as the password in your code, and you're done.

### Security Error

Google Spreadsheet plugin does not work in the Unityweb player's security sandbox. You should change the *Platform* to 'Stand Alone' or something else such as 'iOS' or 'Android' platform in the **Build Setting**.