

目录

介绍 开始了	1.1
Excel如何 谷歌	1.2
豪多 故障排除	1.2.1
— 高强 谷	1.2.2
歌	1.2.3
用法	1.2.3.1
运行时加载 与	1.2.3.2
LINQ一起使用	1.3
支持的细胞类型枚举	1.3.1
类型 数组类型	1.3.2
配方细胞类型 案	1.4
例分析	1.4.1
公式计算：第一部分 公式计	1.4.2
算：第二部分 公式计算：第	1.4.3
III部分	1.5
提示和已知问题 常问问题	1.5.1
附录	1.5.2
Goolge OAuth2设置 代码模	1.5.3
板	1.6
参考	1.7
	1.8.1
	1.8.2
	1.9

团结，QuickSheet中

Unity-QuickSheet使您可以在Unity编辑器中使用Google和Excel电子表格数据。使用Unity-QuickSheet，您可以从电子表格中检索数据并将其另存为资产文件[ScriptableObject](#)即使不编写单行代码也可以格式化。

特征

- **简单**！无需编写任何单行代码。
- **方便**！它可以从excel文件中检索数据。（Windows上支持xls和xlsx格式，OSX上只支持xls。）
- **灵活**！它可以从谷歌电子表格中检索数据。
- **快速**！无需编写解析器来检索数据，它会自动将检索到的数据序列化为Unity3D [ScriptableObject](#)，二进制格式，因此它比使用通常是ASCII格式的XML快。

再说一遍，你甚至不需要编写单行代码！

入门

查看页面 [“Excel HOWTO”](#) 页面如果您使用Excel的电子表格，否则 [“Google Spreadsheet Howto”](#) 页面将以Unity-Quicksheet开头。

还有[常问问题](#)任何其他信息的页面。

您还可以找到找到的Unity论坛页面[这里](#)。如果您有任何问题或建议，请将其发布在论坛上或将其作为问题留在论坛上[github项目页面](#)。欢迎任何反馈！

用法

您可以在任何需要的地方使用“Unity-QuickSheet”将电子表格数据导入Unity编辑器而不会感到痛苦。它简单快捷。

- 在客户端将其用作数据库。与json或xml相比，它简单快捷。与LINQ一起使用。您可以
- 更轻松的处理数据。
- 支持各种类型，不仅是原始类型like int, float, 还有枚举和数组。本土化。
- 公式计算的自动化。请参阅案例研究部分。

执照

此代码根据MIT许可证的条款和条件分发。一些代码是从中借来的[格达达布](#)。跟随他们的许可证。版权所有 (c) 2013 Kim, Hyoun Woo

如何使用Excel

这篇文章显示并帮助您设置和使用[团结](#)，[QuickSheet](#)中与excel。

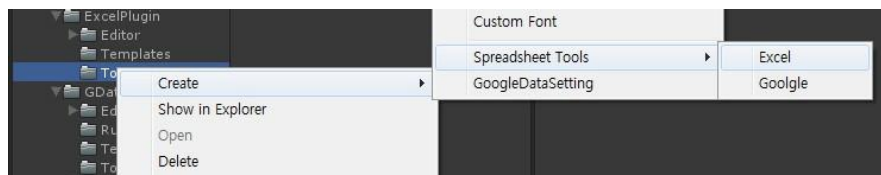
Excel版本97/2000 / XP / 2003支持'.xls'（由于这个原因）[恩波伊](#)，用于读取Unity-QuickSheet的excel电子表格的开源项目不支持'.xls'的Excel版本5.0 / 95。这与'.xlsx'无关

	A	B	C	D	E	F	G
1	MeleeSkillLevel	STR	DEX	INTL	TotalDamage	TotalArmor	Health
2	0	10	10	10	8	16	56
3	1	13	11	11	10	20	66.6
4	2	16	12	12	12	24	77.2
5	3	19	13	13	14	28	87.8
6	4	22	14	14	16	32	98.4
7	5	25	15	15	18	36	109
8	6	28	16	16	20	40	119.6
9	7	31	17	17	22	44	130.2
10	8	34	18	18	24	48	140.8
11	9	37	19	19	26	52	151.4
12	10	40	20	20	28	56	162
13	11	43	21	21	30	60	172.6
14	12	46	22	22	32	64	183.2
15	13	49	23	23	34	68	193.8
16	14	52	24	24	36	72	204.4

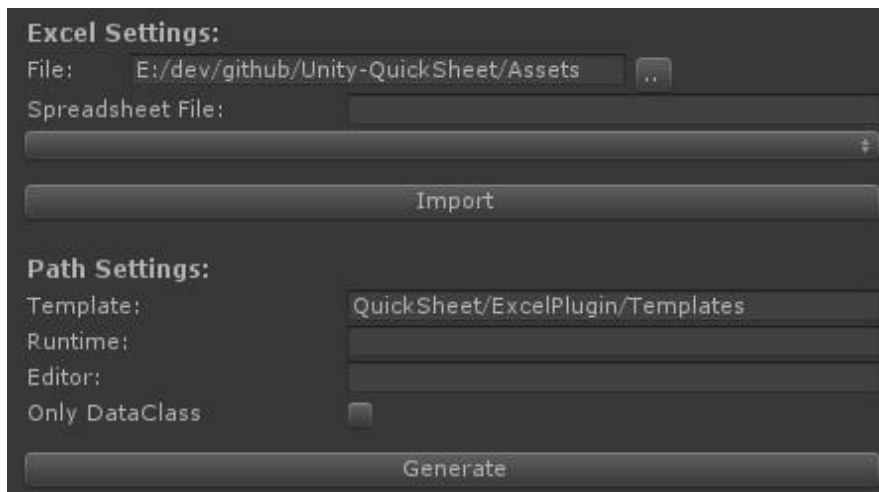
在开始之前，请再次检查您的电子表格页面。它应该从没有空行开始，这意味着第一行不应该是空行。

步骤1) 创建Excel设置文件

首先，你需要做的是创建一个excel设置文件。只需右键单击项目视图，然后选择“创建>电子表格工具> Excel”。它创建一个新文件，显示各种设置以创建脚本文件并从指定的excel文件中获取数据。



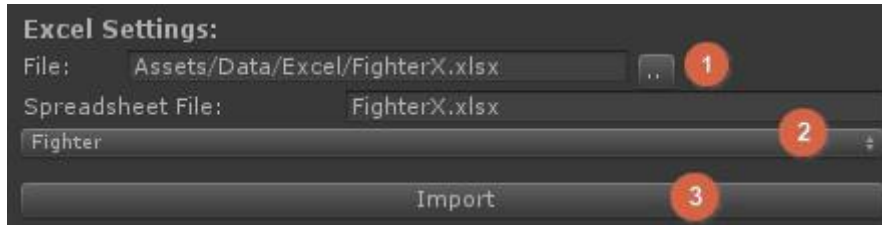
选择Excel菜单项，然后创建设置文件。它可能会显示如下：



让我们开始做setting。

文件设置

文件设置正在设置要导入的excel文件及其工作表页面。



首先，您应指定要导入的excel文件。

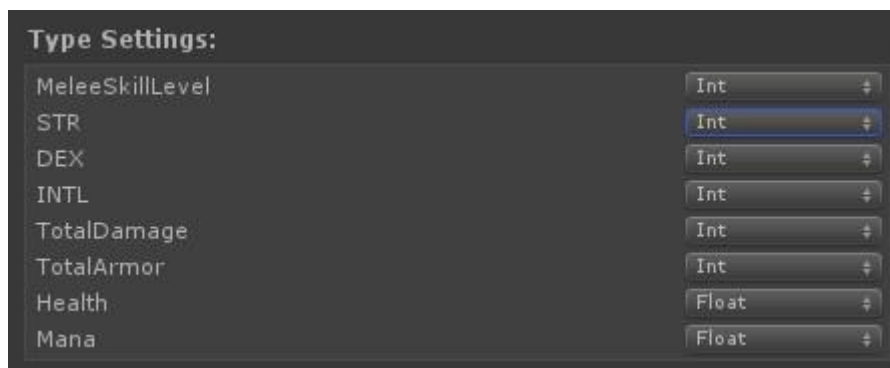
1. 使用“文件打开”对话框选择要导入的Excel文件。

excel文件可以包含一个或多个工作表页面，因此您需要决定选择哪个工作表并从中检索数据。

1. 选择要导入的工作表页面。
2. 按导入按钮导入指定的Excel文件并显示所有列标题。

类型设置

导入指定的工作表页面会显示页面的所有列标题。您需要设置每个单元格的类型。



设置正确的单元格类型。

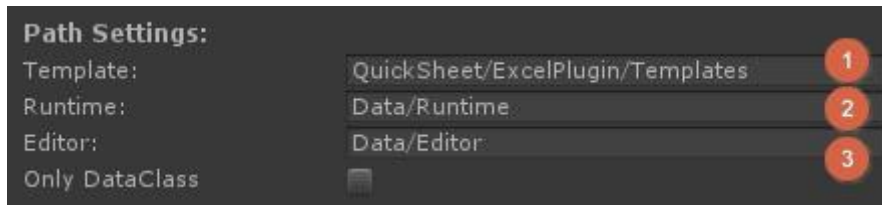
目前以下类型是supported:

- String
- int浮
- 动双枚
- 举布尔
-
-

路径设置

路径设置涉及指定生成的脚本文件的放置路径。

注意：所有路径都应该是相对的，没有'Assets /'。



1. **模板**表示生成脚本文件所需的模板文件的路径。在大多数情况下，您无需更改它。
2. **运行时**表示将放置在运行时使用的生成脚本文件的路径。
3. **编辑器**会在一个路径中显示生成的编辑器模式下使用的脚本文件。

步骤2) 生成脚本文件

如果您已完成所有必要的设置，则需要生成一些脚本文件，这些脚本文件是从Excel文件的工作表页面读取数据所需的，并将其存储在ScriptableObject中，该文件作为项目视图中的资产文件。

按Generate按钮。

生成一些脚本文件后，Unity Editor开始编译它们。等到Unity结束编译然后检查指定的编辑器和运行时路径是否正确生成了所有必需的脚本文件。

在Editor文件夹中应该包含两个文件：

- *你的片名*
-

在运行时，foller应包含两个文件：

- *您-sheetpag-name.cs*
- *您-sheetpage-nameData.cs*

请参阅your-sheetpage-nameData.cs文件。该文件的类成员表示工作表页面的每个单元格。

```

使用UnityEngine;
使用System.Collections;

[System.Serializable]公共
类FighterData
{
    [序列化字段] int
    MeeSekLILL;

    [曝光特性]
    公共int Meleeskilllevel {
        得到{return
            meleeskilllevel;} {MeleSkiLealSt=
            值}; }
    }

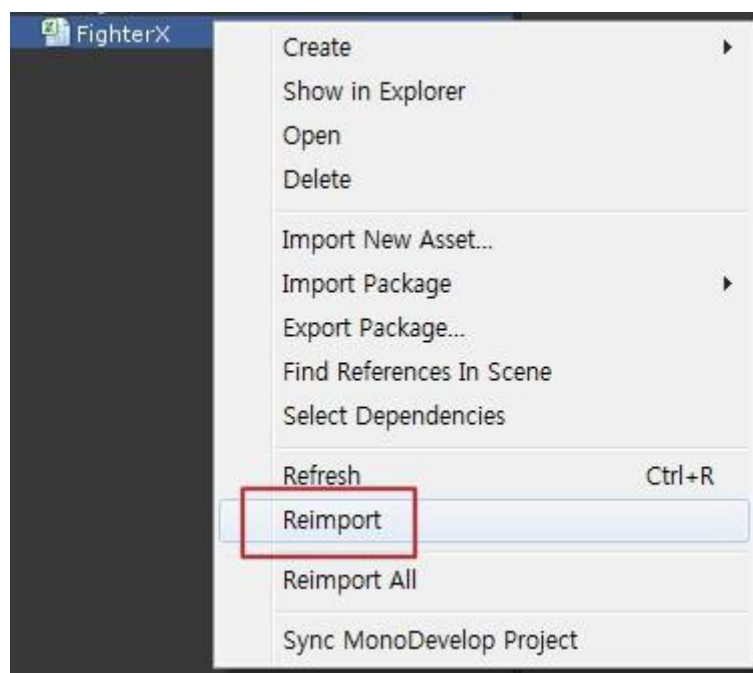
    [ SerializeField ]
    int STR;

    [曝光特性]
    公共int STR{get {STR STR; }设置 {STR= Value}; }

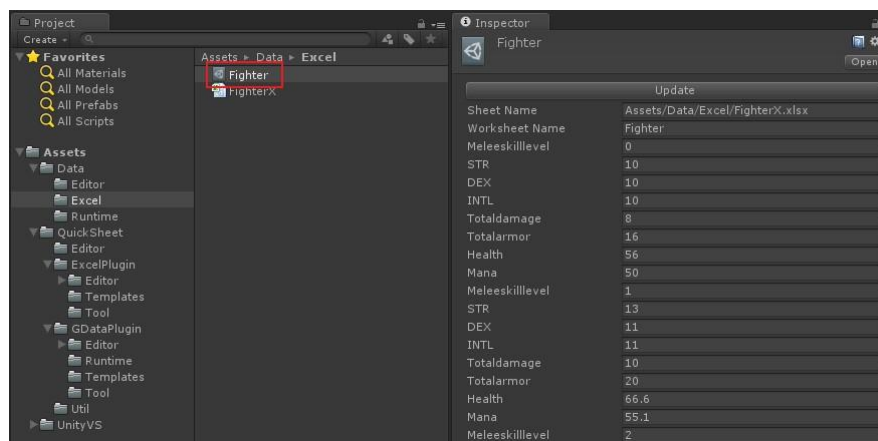
```

步骤3) 导入电子表格数据

只需在Project视图中重新导入任何xls或xlsx文件，即可创建资产文件并将数据从电子表格文件导入到创建的资产文件中。



重新导入xls或xlsx文件将自动创建一个资产文件，该文件具有与工作表页面名称相同的文件名（不是excel文件名），并自动将数据从excel文件的工作表页面导入到创建的资产文件中。



完成。希望你喜欢！

如何使用Google电子表格

该文档的这一页描述并帮助您设置和使用[团结, QuickSheet](#)中同'Google电子表格'。

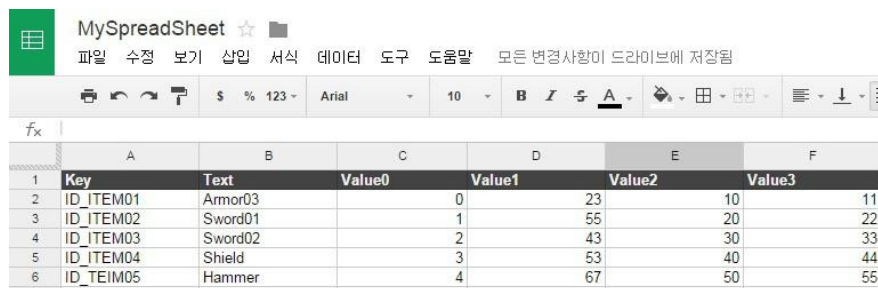
设置OAuth2以访问Google云端硬盘

自2015年5月5日起, Google已更改了身份验证方案。现在, 要访问Google电子表格, 您需要设置OAuth2。要进行此设置, 请访问[谷歌开发者控制台](#)然后创建一个新项目, 启用Drive API, 创建“服务帐户”类型的新客户端ID并下载json文件。

看到[这一页](#)用于设置凭据并获取OAuth2' client_ID' 和' client_secret' 那些是设置谷歌电子表格设置文件所必需的。

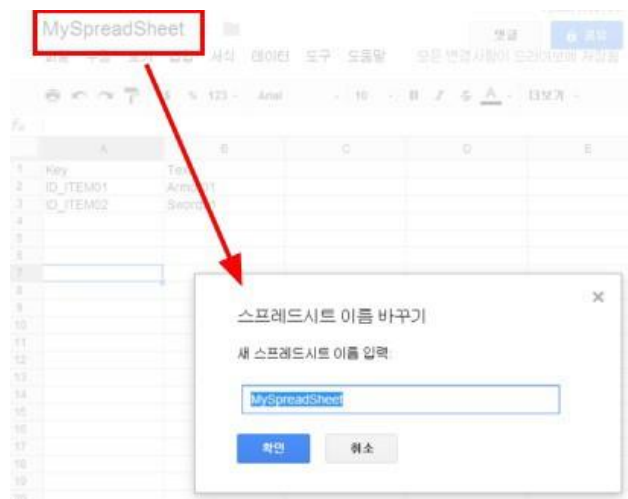
创建电子表格和工作表

使用您的帐户登录“Google云端硬盘”后, 在“Google云端硬盘”上创建一个Google电子表格。

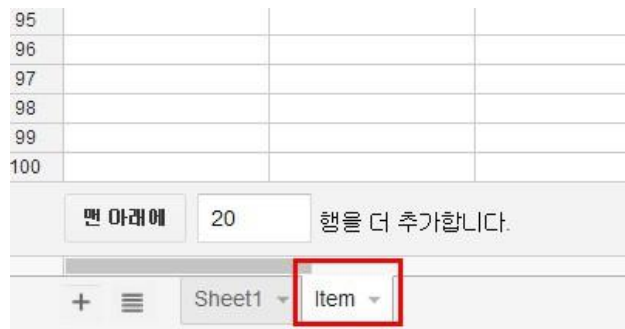


	A	B	C	D	E	F
1	Key	Text	Value0	Value1	Value2	Value3
2	ID_ITEM01	Armor03	0	23	10	11
3	ID_ITEM02	Sword01	1	55	20	22
4	ID_ITEM03	Sword02	2	43	30	33
5	ID_ITEM04	Shield	3	53	40	44
6	ID_ITEM05	Hammer	4	67	50	55

将创建的电子表格的标题更改为“MySpreadSheet”, 如下所示:



接下来, 创建一个新工作表并将其重命名为您想要的任何内容, 如下图所示:



现在，它需要编辑电子表格的单元格。在创建的工作表的第一行插入“Key”和“Text”，如下所示：

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

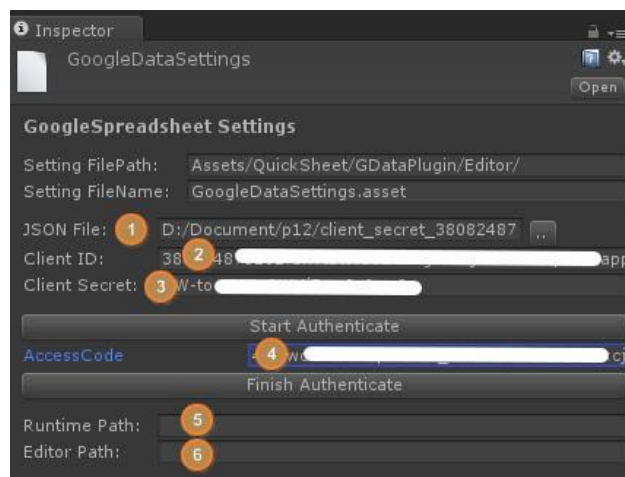
重要说明请注意，第一行上的单元格名称将用作生成的类的成员字段名称，稍后您将在此页面上看到。因此，请注意不要使用像 'int', 'string' 这样的名称，它们是C#的关键字。

Google OAuth2服务帐户

在进一步开始之前，您需要创建“Google OAuth2帐户”以验证您的帐户并使您可以在Google电子表格中访问Unity。

如果您尚未设置“OAuth2 Service Account”，请参阅[Google API上的OAuth2设置](#)在此页面上找到的页面。在将数据导入Unity编辑器之前，必须首先进行设置。

如果您成功获取了json私钥，则选择“GoogleDataSettings.asset”文件，该文件位于“Assets / QuickSheet / GDataPlugin / Editor”文件夹下。



1. 首先，将下载的json私钥设置为“JSON文件”。
2. 现在，您可以看到2) '客户ID'和3) '客户秘密' 将自动分类
3. 单击“开始验证”按钮。它将使用以下图像启动您的浏览器：



选择“允许”按钮，然后转到下一步。

现在您将看到“访问代码”。将其复制并粘贴到Unity的“访问代码”设置中。



最后一步是单击“完成验证”按钮以验证您的凭据。

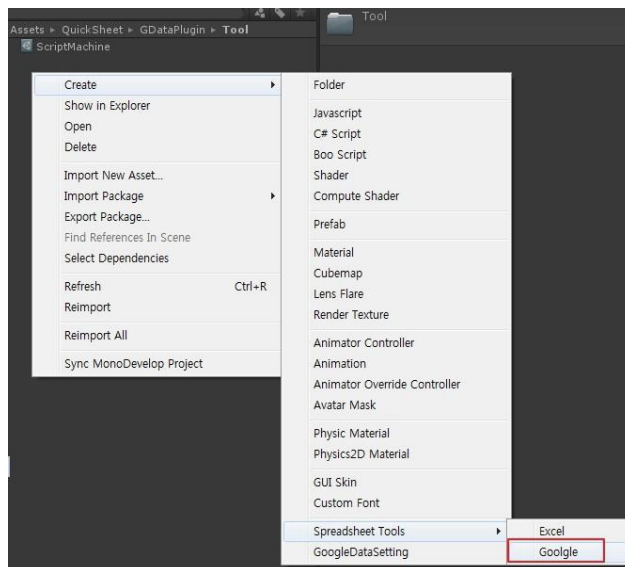
并为您的项目设置其他设置，如“运行时路径”和“编辑器路径”。它可能足以设置如下：

运行时路径：数据/运行时编辑
器路径：数据/编辑器

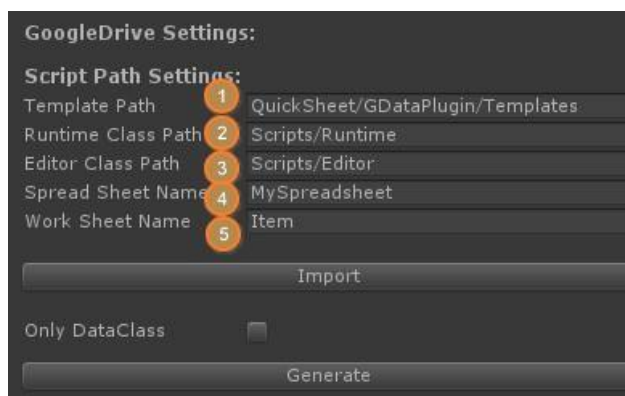
它假定' Data' 文件夹位于' Assets' 文件夹下。

步骤1) 创建Google电子表格设置文件

首先，你需要做的是创建一个谷歌电子表格设置文件。只需右键单击项目视图，然后选择“创建>电子表格工具> Google”。它会创建一个新文件，显示各种设置以创建脚本文件并从指定的Google电子表格中获取数据。



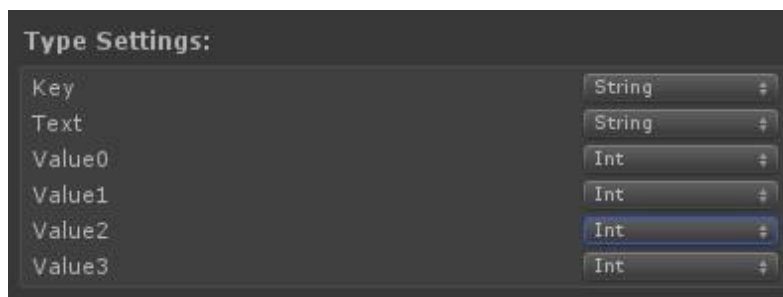
选择Google菜单项，然后创建设置文件。它可能会显示如下：



脚本路径设置

1. **模**板表示生成脚本文件所需的模板文件的路径。在大多数情况下，您无需更改它。
2. **运**行时表示将放置在运行时使用的生成脚本文件的路径。
3. **编**辑器会在一个路径中显示生成的编辑器模式下使用的脚本文件。
4. **S**pread Sheet Name是在Google驱动器上创建的电子表格的名称。
5. **工**作表名称是您要从中获取数据的spreadwheet的工作表名称之一。

完成所有路径设置后，按导入按钮，然后显示页面的所有列标题。您需要设置每个单元格的类型。



设置正确的单元格类型。

目前以下类型是supported:

- String
- int浮
- 动双枚
- 举布尔
-
-

步骤2) 生成脚本文件

如果您已完成所有必要的设置，则需要生成一些脚本文件，这些脚本文件是从Excel文件的工作表页面读取数据所需的，并将其存储在ScriptableObject中，该文件作为项目视图中的资产文件。

按Generate按钮。

生成一些脚本文件后，Unity Editor开始编译它们。等到Unity结束编译然后检查指定的编辑器和运行时路径是否正确生成了所有必需的脚本文件。

在Editor文件夹中应该包含两个文件：

- 您的*SueTPAGE-NAMESETCREATOR*。
-

在运行时，foller应包含两个文件：

- 您-*sheetpag-name.cs*
- 您-*sheetpage-nameData.cs*

请参阅your-sheetpage-nameData.cs文件。该文件的类成员表示工作表页面的每个单元格。

```

使用UnityEngine;
使用System.Collections;

///
/// !!!机器生成的代码!!!
/// !!!请勿将标签更改为空格!
//[系统可串行化]
公共类PlayerItemData
{
    [SerializeField]字符串
    键;

    公共字符串键 {get {返回键;} 设置 {key = 值; }}

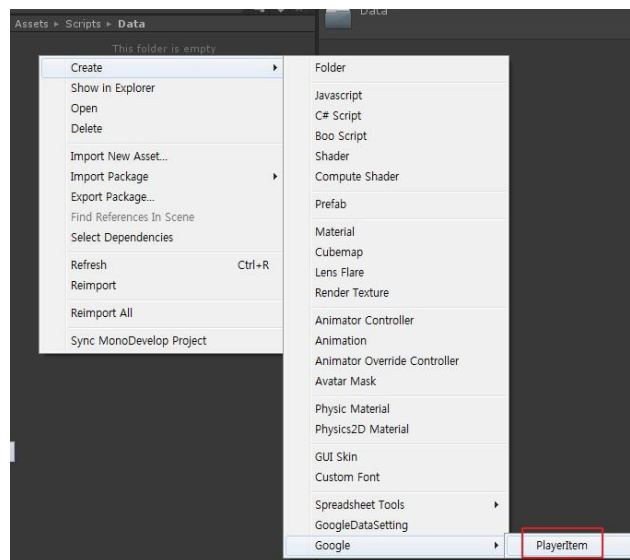
    [SerializeField]字符串
    文本;

    公共字符串文本 {GET {返回文本;} 设置 {Text = Value}; }

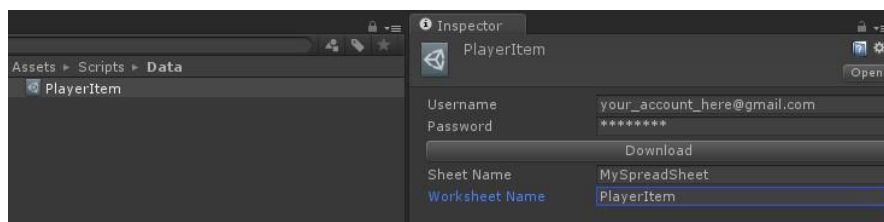
```

步骤3) 导入电子表格数据

现在，您需要创建一个资产文件，该文件将导入并存储来自Google云端硬盘的所有数据。只需右键单击项目视图，然后选择“创建> Google”。现在有一个新的菜单项，其名称与谷歌电子表格的工作表名称相同。



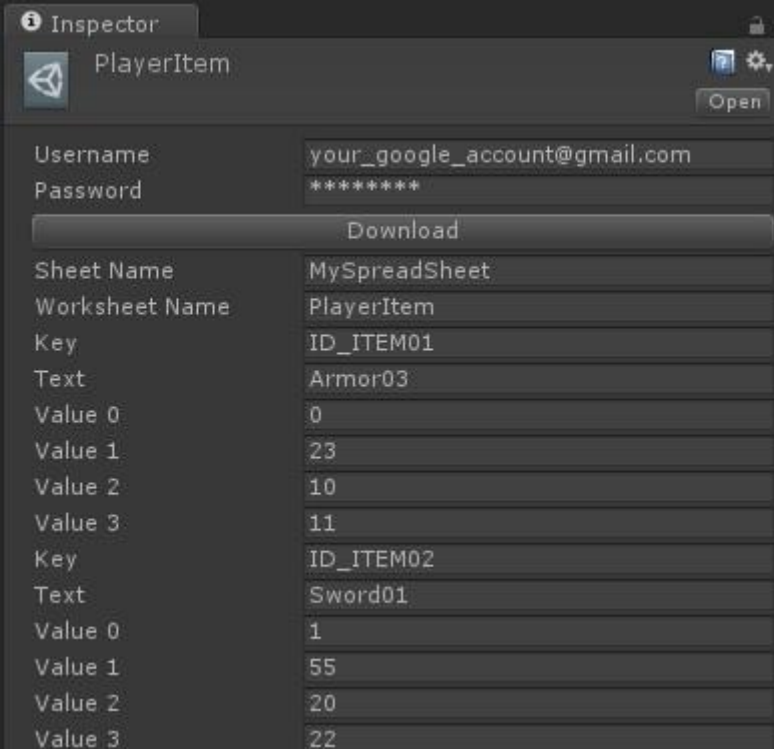
选择菜单项，然后创建资产文件。它可能会显示如下：



请注意，创建的资产文件与指定的工作表名称具有相同的文件名。

1. 输入您的Google帐户的用户名和密码。
2. 指定与Google设置相同的电子表格和工作表名称。
3. 按下“下载”按钮，然后开始从谷歌硬盘导入数据。（可能需要几秒钟。）

完成下载会在Inspector视图中显示导入的数据，如下所示：



The Inspector window displays the 'PlayerItem' component. It includes fields for 'Username' (your_google_account@gmail.com) and 'Password' (masked with asterisks). A 'Download' button is present. Below, the imported data is shown in a list format:

Sheet Name	MySpreadSheet
Worksheet Name	PlayerItem
Key	ID_ITEM01
Text	Armor03
Value 0	0
Value 1	23
Value 2	10
Value 3	11
Key	ID_ITEM02
Text	Sword01
Value 0	1
Value 1	55
Value 2	20
Value 3	22

完成。希望你喜欢！

如果您遇到任何麻烦或问题，请参阅此处的“故障排除”页面。

故障排除

高强

‘无法阅读内容类型部分！’Mac上的错误

在Mac计算机上，打开.xlsx文件会导致错误“无法读取内容类型部分！”。将其保存为

“.xls”，然后再次打开。它解决了这个问题。

支持的’.xls’文件版本

Excel版本97/2000 / XP / 2003支持’.xls’（由于这个原因）[恩波伊](#)用于读取Unity-QuickSheet的Excel电子表格的开源项目不支持“.xls”的Excel版本5.0 / 95但是你不关心“.xlsx”。

Excel反序列化异常异常错误

Unity控制台上可能存在异常错误，例如“Excel反序列化异常：对象引用未设置为objectRow [5]的实例，单元格[6]该单元格是否为空？”如果电子表格的单元格为空。

检查具有错误的给定行和单元索引的单元格。然后看到单元格被定义为字符串类型而不是空的。如果单元格为空，请填写适当的数据。

谷歌

凭据错误无效

如果您在尝试通过单击“下载”按钮获取数据时遇到显示为无效凭据的错误，请检查您的google accout页面并进行两阶段验证。

如果您使用Google两阶段验证，则无论您的Google密码是什么，都不会被接受。您需要生成（在Google上）所谓的特定于应用程序的密码（ASP）。去[Google帐户页面](#)并设置一个ASP，在代码中输入您生成的密码作为密码，然后就完成了。

安全错误

Google Spreadsheet插件无法在Unity网络播放器的安全沙箱中使用。您应该在构建设置中将平台更改为“独立”或其他内容，例如“iOS”或“Android”平台。

运行时加载

在运行时将导出的.asset scriptableObject文件加载到场景中的方法与Unity3D中的任何其他内容没什么不同。

我们假设您有一个简单的电子表格，如下图所示：

	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

并且您已将其导出为.asset scriptableObject。

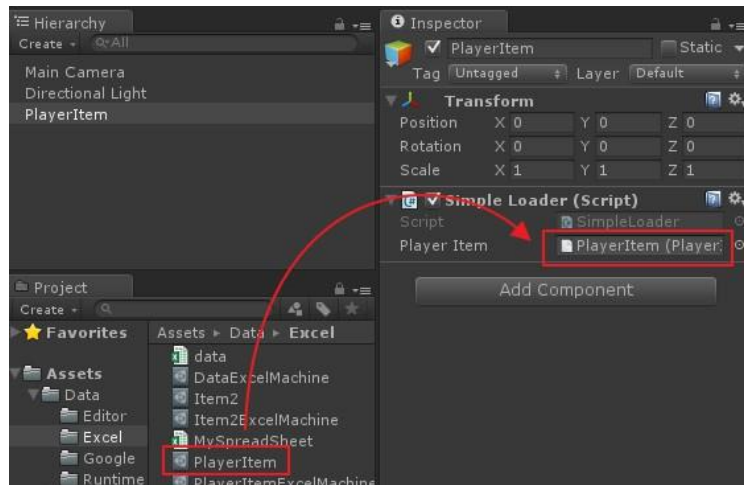
加载.asset scriptableObject的操作与预制文件相同。

首先，创建一个新的MonoBehaviour脚本文件，然后将PlayerItem类实例声明为public。

```
公共类SimpleLoader: MonoBehaviour
{
    //你已经有了PlayerItem类public PlayerItem
    PlayerItem;

    无效启动 ()
    {
        // PlayearItem.dataArray数组包含所有导出的数据。
        //它会将Value0值0输出到控制台上。LogFormat ( "Value0:{ 0 }",
        PraveIt.j.DATAARDATAL (0) .Value0) ;
    }
}
```

现在，只需拖动.asset文件并将其放入Inspector视图即可。



这就是在运行时加载 `.asset` 文件的全部内容。现在开始播放，`'Value0'` 的值将被放入控制台。

使用LINQ

本文档的这一部分描述了如何使用LINQ（语言集成查询）轻松完成查询从复杂电子表格导出的特定数据集。

强烈建议看[101-LINQ样品](#)如果您还不熟悉LINQ。LINQ还有很多文档，所以只需谷歌搜索。

好的，我们开始吧。

正如您已经知道的那样，像Excel这样的电子表格是游戏设计师日常用来制作游戏中使用的各种表格和数据集的最强大工具之一。

下图显示了游戏中所有武器物品的表格数据。请注意，显示的列不是表中的所有列，对于通常的游戏可能会有更多，特别是如果游戏是RPG类型。

	A	B	C	D	E	F	G	H	I	Q	R	S	T	U	V
1	ID	Name	STR	DEX	INT	FTH	Uni	Cr	Ph	S	Dx			WeaponType	DamageTyp
2	1000	Battle Axe	12	8	0	0	4	100	250	C	D			Axe	Standard
3	1001	Hand Axe	9	8	0	0	2.5	100	220	C	D			Axe	Standard
4	1002	Thrall Axe	8	8	0	0	1.5	100	208	C	C			Axe	Standard
5	1003	Dragonslayer's Axe	18	14	0	0	4	100	180	D	D			Axe	Standard
6	1004	Butcher Knife	24		0	0	7	100	190	S				Axe	Slash
7	1005	Brigand Axe	14	8	0	0	3	100	248	C	D			Axe	Standard
8	1006	Winged Knight Twinaxes	20	12	0	0	8.5	100	244	C	D			Axe	Standard
9	1007	Elonora	20	8	0	0	6.5	100	284	D	D			Axe	Standard
10	1008	Man Serpent Hatchet	16	13	0	0	4	100	250	C	D			Axe	Standard
11	1009	Short Bow	7	12	0	0	2	100	154	E	D			Bow	Projectile

无论给定电子表格的表格有多复杂，都可以通过Unity-Quicksheet将表格轻松输入Unity编辑器。因此，假设已成功完成将表导入Unity。

现在我们已经自动生成了WeaponTable.cs脚本文件，该文件将该类显示如下：

```
公共类WeaponData
{
    公共int ID {get; 组;public string
    Name {get;组;}公共int STR{get; 组;}
    ...
}
```

还有一个ScriptableObject派生类，用于.asset文件。

```
公共类WeaponTable: ScriptableObject
{
    ...
    公共武器数据[]数据阵列;
    ...
}
```

为了便于管理项目，电子表格中的所有项目都有其唯一ID（请参阅第一列），该ID用于标识项目类型与其范围的关系。

类型	范围
武器	1000 ~ 1999
盔甲	2000 ~ 2999
耗材	3000 ~ 3999

范围可以是多种，取决于其项目类型的数量或项目本身的每个数量。

因此，只要具有项目ID的整数数组就足以描述清单中的项目。

```
公共课库存
{
    ...
    //当前属于清单中的项目int [] items = {
        1000, 1001, 1002, 1003, 2002, 2003,
        2004, 2005, 2006, 2007, 2008, 2009,
        2010, 2011, 2012, 2013, 2014, 2015,
        2016, 2017, 2018, 4014, 4015, 4016,
        4017, 4018, 4019, 4020, 4021, 4022,
        4023, 4024, 4025, 4026, 4027, 4028, 4029,
        4030, 4031, 4032, 4033, 4034, 4035, 4036
    };
    ...
}
```

然后我们需要检索清单中的实际项目数据，这些数据可用于查询项目表。

ItemManager是一个类，它提供处理和访问清单中各种项目的方法。将武器成员字段注释为从Unity-Quicksheet自动创建的intance。

```

公共类ItemManager: Singleton <ItemManager>
{
    公共武器表武器表;
    ...
    //武器IDN: 1000~1999
    公共列表<武器数据> GETARONSONLIST (INT[IDS])
    {
        清单<武器数据>项目=新列表<武器数据> ();

        VAR武器ID= IDS。 (x=>x>1000和& x<2000) 。
        var weaponResult = from w武器表带.DataArray //来自电子表格的所有武器类型物品
        .
            来自武器ID中的n //库存中的武器。其中w. ID == n
            选择w;返回武器
        结果
    }
}

```

从上面的代码可以看出，使用LINQ作为GetWeaponList方法可以轻松地检索武器类型的项目。

将电子表格中的表格视为一种数据库。然后在编辑时使用Unity-Quicksheet提取和导入数据到Unity，并且更喜欢使用LINQ在运行时查询表中的数据。无论桌子多么复杂，它都是小菜一碟。

在Quicksheet中使用ENUM类型

指定数据类的枚举类型很简单。假设您要为'RearType'设置枚举类型
在电子表格中，如下所示：

	A	B	D	E	F
1	Id	Id	SkillType	RearType	Card
2	1	Smack2	Active	Normal	2
3	2	Strike2	Passive	Rare	2
4	3	Smack3	Active	Legend	3
5	4	Strike3	Passive	Normal	3

'RearType'只能有三种类型的值中的一种，即Normal，Rare和Legend。

由于QuickSheet本身无法生成枚举，因此应在生成脚本文件之前首先声明枚举类型。

创建一个空的.cs文件，该文件通常包含各种枚举，然后声明'RearType'枚举类型，您可以在电子表格中设置。

公共枚举类型

```
{  
    普通,  
    稀有,  
    传奇,  
}
```

现在您可以生成必要的脚本文件而不会出错！

将数组类型与QuickSheet一起使用

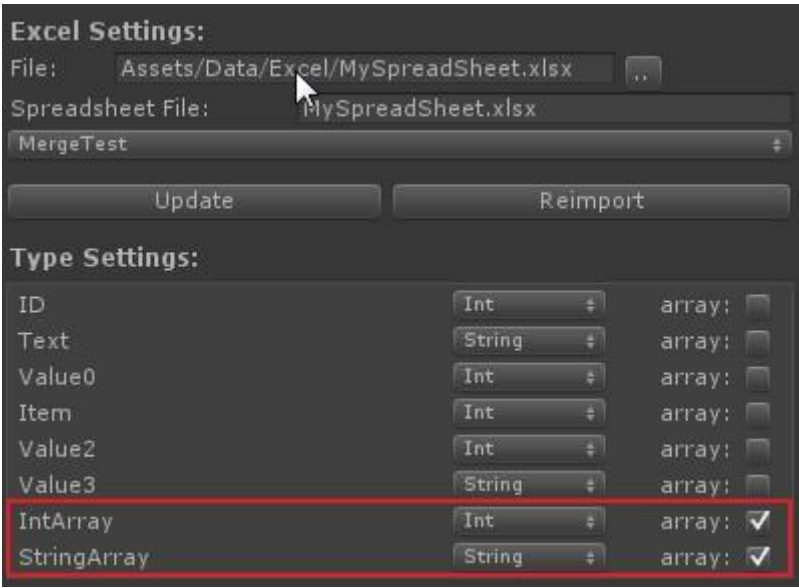
您可以在单元格中使用带逗号值的数组类型，如下所示：

	A	B	C	D	E	F	G	H
1	ID	Text	Value0	Item	Value2	Value3	IntArray	StringArray
2	1	Armor03	0	23	10	a	1,2,3,4,5,	a,b,c,d,e,
3	2	Sword01	1	55	20	b	1,2,3,4,5,	a,b,c,d,e,
4	3	Sword02	2	43	30	c	1,2,3,4,5,	a,b,c,d,e,
5	4	Shield	3	53	40	d	1,2,3,4,5,	a,b,c,d,e,
6	5	Hammer	4	67	50	e	1,2,3,4,5,	a,b,c,d,e,
7								

请注意，不要错过应该在单元格中的最后一个值之后的最后一个逗号。（在v. 0. 0. 0. 0上更改）

1, 2, 3, 4, 5, ->在v. 1. 0. 0. 0之后不再需要' 5' 之后的逗号。

使用给定的excel文件导入后，指定每个列标题的类型，并检查数组类型的数组选项。



它将生成具有指定类型的数据类的数组类型的memeber字段，如下所示：

```
[序列化字段]
INT[] INTARTRAI=新INT [0] ;

公共INT[] INTARTRAI{{GET {返回INTARTRAY; } {ntLay=值; } [ SerializeField ]
String [] String数组=新字符串[0 ];

公共字符串[] String数组{get {String String数组; } {StRealStord=值; }}
```

注意：枚举类型数组仅支持谷歌电子表格，不支持excel。

配方细胞类型

您甚至可以在电子表格中公式化单元格类型。

C2

✕

✓

=AbitiliesBase[@cooldown]/Metadata[Pacing]

	A	B	C	D
1	AbilityType	castTime	cooldown	power
2	Heal	0.666666667	6.666666667	5
3	Poke	0	2.666666667	15
4	Barr	1.333333333	4	0
5	Nuke	6.666666667	40	6
6	Stun	0	6.666666667	15
7	Swap	0	6.666666667	0
8	Copy	0	0	0
9	Taunt	0	13.33333333	0
10	Buff	0	6.666666667	2

使用任何公式计算单元格上的值，就像在电子表格上一样。Unity-Quicksheet还可以将其与任何其他单元格类型（如数字或字符串）无差异地序列化。

注意：目前只能在Excel上使用。（v. 1.0.0.1）

公式计算的自动化

RPG中的角色通常具有各种统计数据，例如HP，MP，等级等，并且这些数据通常通过各种给定的公式计算。例如，角色的力量随着它们的升高而增加，这可以用公式的形式表示。

但是如何处理公式存在问题。您可以在代码中编写公式。容易但是每当公式改变时都应该重写它。游戏设计师在整个开发时间一次又一次地改变这一点并不奇怪。

所以我们需要更灵活的方式来处理公式，作为一种数据形式而不是代码片。

本文档的这一部分描述了如何使用电子表格和Unity-Quicksheet插件以完全数据驱动的方式轻松完成公式计算。

在进一步阅读之前，如果您不熟悉RPG的典型统计数据和公式，强烈建议您阅读该文章，[游戏系统的工艺：实例第一](#)。

电子表格上的公式

首先要做的是在电子表格中定义各种统计数据和公式，这样它就可以在游戏中使用。

您可以使用Excel或Google电子表格中的任何一种，但本文档中的说明基于Excel电子表格。

让我们考虑一下你试图制作的游戏中的角色随着他的升级而增加。他有三个基本的统计数据来描述他的力量。每个都是力量（STR），灵巧（DEX）和智力（ITL）。随着他的数据增加，他获得更多的健康（HP）和法力（MP）。

下图显示了电子表格中的每个统计信息及其相应的公式。

	A	B
1	Stat	Formula
2	STR	Round((5 * (SkillLevel * 0.6)), 0) + 10
3	DEX	Round((5 * (SkillLevel * 0.2)), 0) + 10
4	ITL	Round((5 * (SkillLevel * 0.2)), 0) + 10
5	HP	(5 * ((STR * 0.5) + (DEX * 0.39) + (ITL * 0.23)))
6	MP	(5 * ((STR * 0.01) + (DEX * 0.12) + (ITL * 0.87)))

电子表格中的数据将用于本文档的示例。

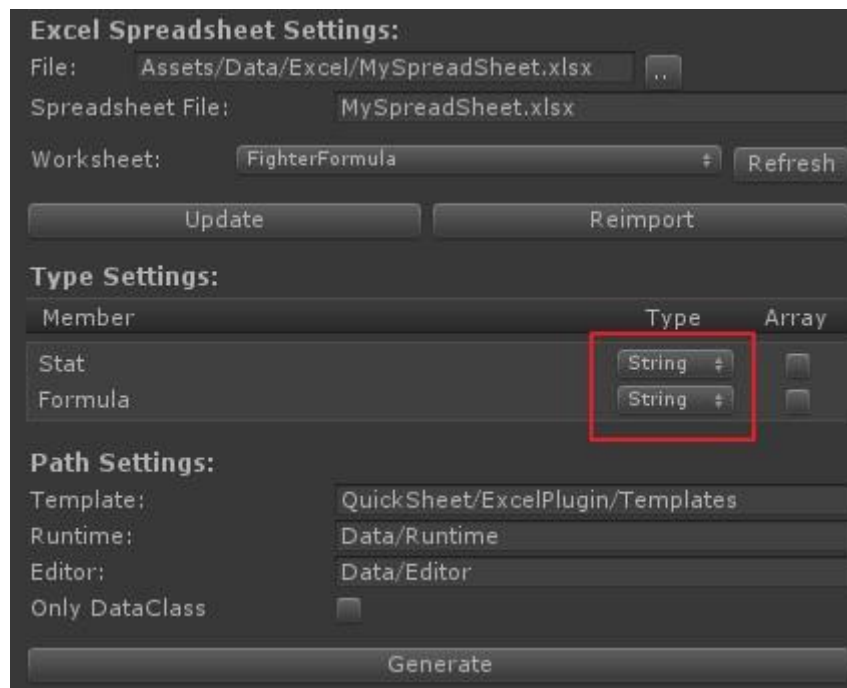
请注意，上述图像的统计和公式来自文章，[游戏系统的工艺：实例](#)。

数据导入

首先，让我们将工作表中的所有数据导入Unity编辑器。

创建导入设置资产文件并通过指定电子表格导入文件。（有关详细步骤，请参阅如何使用Excel页面。）

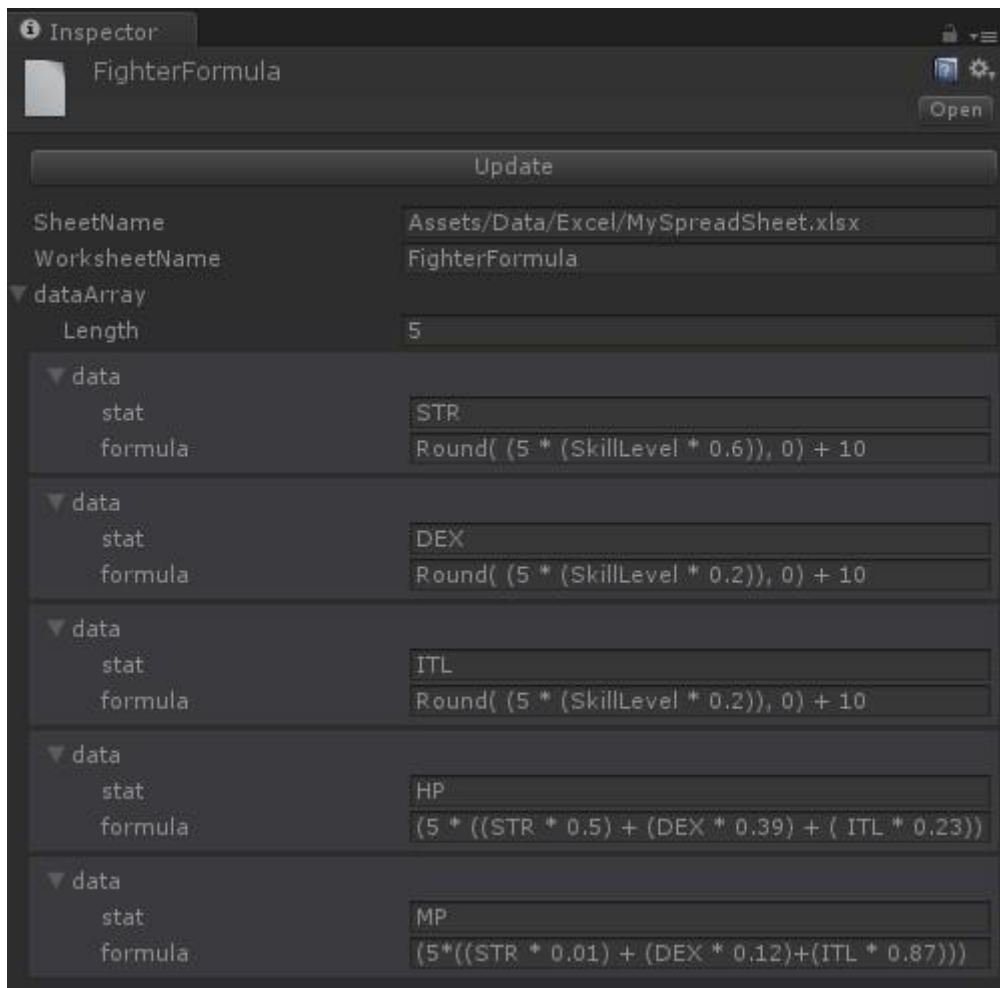
将“类型设置”中的统计类型和公式设置为字符串类型。Stat只是属于一个玩家的每个stat的名称，所以将其类型设置为字符串似乎没问题，但为什么Formula也被设置为字符串？答案将在稍后找到。此刻，只要再想一想，你就会知道除了字符串之外，其他公式没有正确的类型。



单击Generate按钮，它将在'Editor'和'Runtime'文本字段中指定的目录下生成所有必需的script文件。

然后在Project窗口中选择.xlsx Excel文件，并选择右键单击所选excel文件所示的Reimport菜单。

在重新导入excel文件时，它会将“ScriptableObject”创建为具有工作表名称的资产文件，并将所有数据导入其中。选择创建的资产文件并检查improted数据是否正确。您可以在Unity编辑器的Inspector视图中看到以下图像：



您可以看到生成的代码，尤其是可以在指定的运行时设置目录下找到的FighterFormuladata类。

```
[系统可串行化]
公共类FighterFormulaData
{
    [序列化字段]字符串统
    计;
    公共字符串STAT {GET {返回STAT; }设置{STAT=值; }}

    [SerializeField]字符串
    公式;
    公共字符串公式 {get {返回公式; }设置{{公式=值}}
}
}
```

如果您不熟悉从电子表格导入数据，请参阅[如何使用Excel](#)或者[如何使用Google电子表格](#)页面取决于电子表格的类型。

编写计算代码

现在，让我们写下代码来计算从电子表格导入的给定公式的每个统计数据。

首先，我们需要一个类来表示游戏中战斗机类型的玩家的所有统计数据。可以使用以下简单的POCO类，PlayerStat。除了SkillLevel属性之外，其他所有属性都显示在电子表格中作为属性。

```
公共类PlayerStat
{
    公共int SkillLevel{get; 组;public float
    STR {get;组;public float DEX {get;组;
    公共浮点ITL{get; 组;公共浮动HP {get;
    组;public float MP {get;组;}
}
```

接下来，我们需要做的是使用相应的公式计算每个stat，并将结果设置回每个stat。

通常，RPG的公式只是一个多项式，我们的导入公式是字符串数据。所以我们需要一个计算器可以将多项式计算为字符串数据的形式。

那么，我们如何计算公式作为字符串数据的形式？

虽然可以通过编写一个简单的计算器来完成，但幸运的是已经存在一堆由C#编写的计算器。理所当然，我们不需要重新发明轮子。

其中一个值得注意的事情是.NET的计算引擎它易于使用，并且已经具有我们的公式计算的大多数功能。

还有钙钙发动机，港口.NET的计算引擎到可以找到的可移植.NET库（PCL）[这个github项目页面](#)。

让我们看看CalcEngine如何使用给定的公式计算为字符串类型。

```
CalcEngult.CalcEng=计算器=新的CalcEngult.CalcEngor（）；计算器：变
量[ a ]=1；

//结果输出2如您所料
VaR结果=计算器。评估（“A+ 1”）；
```

如上所示，只需在评估然后调用之前指定公式的任何变量

评估是获得结果的所有事情。够简单吗？

创建一个脚本文件，因为'Player.cs'为Unity的游戏对象的组件派生MonoBehaviour类。

```

公共类玩家: MonoBehaviour
{
    // ScriptableObject包含电子表格中的导入数据。公共战士公式战士公式;

    //持久性数据的类实例
    PraseStasePraseStase=新PraseStaseUs ();
    ...
}

```

现在是时候用相应的公式来计算每个stat。

CalcEngine方便而强大的特性之一是'DataContext'，这是一种将.NET对象连接到CalcEngine评估环境的绑定机制。

不是将每个变量都指定给CalcEngine，只需将“PlayerStatus”类实例设置为CalcEngine的DataContext，就可以进行计算了。

如前所述，调用CalcEngine的“Evaluate”函数计算给定的公式并返回相应的stat值。无论它是什么公式，计算它并获得结果都没有区别。

```

无效启动 ()
{
    //指定技能等级
    playerStatus.SkillLevel = 4;

    CalcEngult.CalcEng=计算器=新的CalcEngult.CalcEngor () ;

    // CalcEngine使用Reflection来访问PlayderData对象的属性
    //因此可以在表达式中使用它们。计算
    器. DATACONTRONET=播放器状态;

    //计算玩家的每个属性
    PraseStasuS.STR=Real. ToSingle (计算器.Apple (GetFormula (STR) ) ) ; LogFormat ( “STR: { 0 }”,
    PraveStest.STR) ;

    ToSingle (计算器) (GetFormula ( “DEX”) ) ; LogFormat ( “DEX: { 0 }”, PraveStest.Dx) ;

    PrimeStasuS. ITL=Trime. ToSingle (计算器. 评估 (GetFormula ( “ITL”) ) ) ; LogFormat ( “ITL:{ 0 }”,
    PraveStest. ITL) ;

    PrimeStasuS. Hp=Trime. ToSingle (计算器.Apple (GetFormula (HP) ) ) ; LogFormat
    ( “HP: { 0 }”, PraveStist.HP) ;

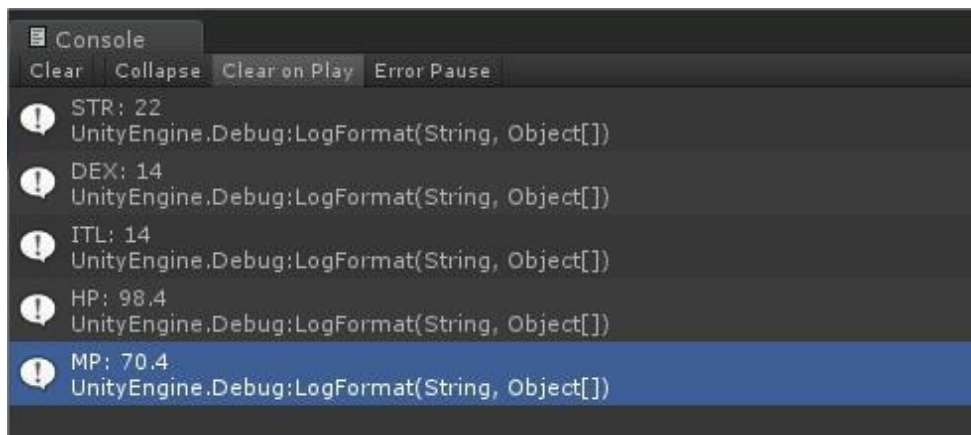
    PrimeStasuS. MP=Real. ToSingle (计算器.Apple (GetFormula (MP) ) ) ; LogFormat
    ( “MP:{ 0 }”, PraveStest. MP) ;
}

//一个辅助函数，用于检索具有给定公式名称的公式数据。字符串GetFormula (字符串公
式名)
{
    返回FraveTrime. DATA数组. 其中 (E= > E. STAT= =公式名称)
        。.FirstOrDefault () 式;
}
}

```

就这样。即使您更改了电子表格中的任何公式，也无需更改代码。只需更新导入的 `scriptableObject.asset` 文件，最后的事情就在运行时完成而不需要任何代码端更改。

现在，运行Unity编辑器，您将在控制台上获得以下结果：



将上述计算统计数据的结果与电子表格中的原始数据进行比较Dungeon Siege 2系统电子表格页。如您所见，它具有相同的结果。

	A	B	C	D	E	F
1	FIGHTER STATISTICS					
2	Melee Skill	0	1	2	3	4
3	STR	10	13	16	19	22
4	DEX	10	11	12	13	14
5	INT	10	11	12	13	14
6	Total Damage	8.0	10.0	12.0	14.0	16.0
7	Total Armor	16	20	24	28	32
8	Health	56.0	66.6	77.2	87.8	98.4
9	Mana	50.0	55.1	60.2	65.3	70.4

在开发周期中，随着调整游戏平衡，设计师尝试多次更改公式。虽然这是自然过程，但您不希望每次更改公式时都更改代码。

如文档所示，这种方法可以避免在设计人员更改公式时更改代码，这样不仅可以节省大量时间，还可以防止错误。

公式计算的自动化：第二部分

在前一个文档中，每个stat都是单独计算的。这是一项繁琐的工作。更糟糕的是，每当将新的stat添加到PlayerStatus类时，我们都应该计算并重新设置。

```
PraseStuts.SkiLeal= 4;

计算器=新的CalcEngult.Calc工程（）；计算
器.DATACONTRONET=播放器状态；

PraseStasuS.STR=Real.ToSingle（计算器.Apple（GetFormula（STR）））；ToSingle（计算
器）（GetFormula（“DEX”））；PrimeStasuS.ITL=Trime.ToSingle（计算器.评估
（GetFormula（“ITL”）））；PrimeStasuS.Hp=Trime.ToSingle（计算器.Apple（GetFormula
（HP）））；PrimeStasuS.MP=Real.ToSingle（计算器.Apple（GetFormula（MP）））；
```

举个例子，考虑如果我们添加一个新的stat，'LUK'用于字符的幸运数据，我们应该像下面的代码一样再次编写它的计算代码。

```
PrimeStasuS.MP=Real.ToSingle（计算器.Apple（GetFormula（Luk）））；
```

这可能不是什么大问题，但是，是的，单调乏味。

如果是这样，如果我们像下面的代码一样完成所有计算，它会是什么样子？公式化

```
PraseStase=新PraveStaseUs（）；
PraseStuts.SkiLeal= 4;

//热潮！
公式计算器=新公式计算器（）；公式计算器.计算<PraseSturt>（PraveSturt,FraveTrime.DATaLART）；
```

它看起来不魅力吗？它不仅简单而且灵活。好的，让我们看看它是如何可能的。

首先要做的是提取公式并解析其对应的统计数据。它可以通过调用FormulaCalculator.GetFormulaTable方法来处理。考虑它创建一个带有给定'FighterFormulaData'数组的字典，该数组从电子表格导入并位于'FighterFormula'类中，它是第一个参数的ScriptableObject派生类。第二个参数是'stat'名称，最后一个是'formula'本身作为字符串类型。

公式计算器


```

public void Calculate <T> (System.Object数据, System.Object [] obj, 字符串键, 字符串值)
{
    // obj: FighterFormulaData数组
    // key: FighterFormulaData类的Stat属性
    // value: FighterFormulaData类的公式属性
    字典<字符串, 字符串>可公式化=可公式化 (OBJ, KEY, VALUE) ; 计算<T> (data,
    formulaTable) ;
}

```

顺便说一句，我们应该只计算公式所需的属性。请参阅’PlayerStatus’类。并非所有的proeprit都被用作公式计算的变量。为此，不需要SkillLevel属性。所以我们需要一种方法来区分用于变量的属性和用于变量的属性。但是怎么样？

可以通过在使用公式变量的任何属性上设置自定义属性来解决此问题。该以下代码中显示的FormulaVariable属性用于此目的。PraseStuls.CS

```

公共类PlayerStatus
{
    公共int SkiLeale{get; 组;}

    //自动识别具有’FormulaVariable’的stat成员字段
    //作为公式变量。[公式]
    公共浮动STR {get;组;}[公式化]
    公共浮动DEX {get; 组;}[公式化]
    公共浮动ITL{get; 组;}[公式化]
    公共浮动HP {get;组;}[公式化]
    公共浮动MP {get;组;}
}

```

’FormulaVariableAttribute’是C#的一个简单属性。公式化属性

使用系统；

```

[属性用法 (AdvestTealGr.Field)] 公共类公式AvavaRable
属性：属性
{
}

```

在计算给定公式之前，GetFormulaProperties方法从 “PlayerStatus”类中收集具有 “FormulaVariable”属性的所有属性。通过反射机制，我们可以轻松地做到这一点。

公式计算器

```
私有属性信息[]
{
    VarType = Type (t) ;

    清单<属性信息>公式Apple: =新列表< PrimeType信息> ( ) ;

    //反思获取给定类T的所有属性。
    PrimeTyFiel[]属性= yType。
;

foreach (属性中的PropertyInfo p)
{
    if (! (p.CanRead &&
        p.CanWrite) ) 继续;

    对象[]属性= P.GeCuto属性 (TRUE) ; foreach (属性中的对
    象o)
    {
        //我们得到一个具有'FormulaVariable' 自定义属性的属性。如果 (O.GETType
        () == Type (公式avabable属性))
            公式P, 加法 (P) ;
    }
}
返回公式公式, toRayay. ( ) ;
,
```

现在，是时候进行实际计算了。我们通过调用'Calculate' 一个泛型方法来实现它，因此它可以对任何类型的类进行计算，而不仅仅是'PlayerStatus' 类。

公式计算器

```

公共空隙计算 (Stase.Objo数据位置, 字典<字符串, String >可公式化)
{
    VaR型= Type (t) ;

    //告诉CalcEngine变量的值。
    CalcDeNo. DATACONTRONT=数据表;

    //评估具有 “FormulaVariable”属性的每个属性。PrimeTyFiel[]属性=
    GETFAULTAICS属性 ( ) ;
    foreach (属性中的PropertyInfo p)
    {
        字符串公式= NULL;
        TryGetValue (P.NoD, Out公式)
        {
            如果 (! ) IsNullOrEmpty (公式)
            {
                var值=CalcEng. Apple (公式) ;
                P.StValue (DeaSt立场, Real. ChangeType (value, P.PrimyType) , NULL) ;
            }
        }
    }
}

```

请注意，应计算的属性的声明顺序很重要。

再次参见’PlayerStatus’类。要评估HP属性的变量保持，首先应评估并设置STR，DEX和ITL的正确值。

```
hp = (5 ( (str 0.5) = (dex 0.39) = (它 0.23) ) )
```

PraseStuls. CS

```

公共类PlayerStatus
{
    ...
    [公式]
    公共浮动STR {get;组;}[公式化]
    公共浮动DEX {get; 组;}[公式化]
    公共浮动ITL{get; 组;}[公式化]
    公共浮动HP {get;组;}
    ...
}

```

就这些。

要记住的一件事是我们大量使用反射来简化我们的公式计算。如您所知，即使在运行时，反射也不是一种快速的方法。所以，当你使用这种方法时，应该认真对待游戏循环速度很重要的地方。例如，避免在MonoBehaviour的更新中进行计算。

以下显示了FormulaCalculator类代码的整行。公式计算器

公共类FormulaCalculator

```
{
    CalcDeal.CalcEngCalcEng=新的CalcEng.

    公共空隙计算< T> (Stase.Objor DATA, Soal.Obj[]Obj.)
    {
        PropertyInfo [] infos = obj [0] .GetType () 。 GetProperties (BindingFlags.Public | BindingFlags.Ins
, 孟清
湘);
        String KEY=iFoSs [0] 。 字符串
        值=iFoS[1 ]。

        计算<T> (数据, obj, 键, 值);
    }

    public void Calculate <T> (System.Object数据, System.Object [] obj, 字符串键, 字符串值)
    {
        字典<字符串, 字符串>可公式化=可公式化 (OBJ, KEY, VALUE); 计算<T> (data,
formulaTable);
    }

    公共空隙计算 (Stase.Objo数据位置, 字典<字符串, String >可公式化)
    {
        VaR型= Type (t) ;

        CalcDeNo. DATACONTRONT=数据表;

        PrimeTyFiel[]属性= GETFAULTAICS属性 () ; foreach (属性中的
PropertyInfo p)
        {
            字符串公式= NULL;
            TryGetValue (P.NoD, Out公式)
            {
                如果 (! ) IsNullOrEmpty (公式)
                {
                    var值=CalcEng.Apple (公式);
                    P.StValue (DeaSt立场, Real. ChangeType (value, P.PrimyType) , NULL) ;
                }
            }
        }
    }

    公共字典<字符串, 字符串> GETFAULTABLE (System .Obj[]) OBR, String键, Stry-Valu
e)
    {
        字典<字符串, 字符串>结果=新字典<字符串, 字符串> ();

        foreach (obj中的System.Object o)
        {
            PropertyInfo [] infos = o.GetType () 。 GetProperties (BindingFlags.Public | BindingFlags.Inst
;

            //字典的关键字
```

```

String STATNAME=空;
VaRe=iFo..West.Debug (E= > Enname=yKEY) .FrStReDebug () ; if
(找到! = null)
{
    对象值=找到.GETValk (0, NULL) ; StasNord=
    值.toStRew () ;
}

//字典字符串的值formula =
null;
找到= iFo..West.Debug (E= > Enname==值) .FrStestDebug () ; if
(找到! = null)
{
    对象值=找到.GETValk (0, NULL) ; 公式=值。
}

If (String, IsNullOrEmpty (StAtNeX) =false and & String, IsNullOrEmpty (公式) =false) 结果。
其他
{
    //错误
}
}

返回结果;
}

私有属性信息[]
{
    VaR型= Type (t) ;

    清单<属性信息>公式Apple:=新列表< PrimeType信息> () ;

    PrimeTyFiel[]属性= yType。
;

foreach (属性中的PropertyInfo p)
{
    if (! (p.CanRead && p.CanWrite) )
        继续;

    对象[]属性= P.GeCuto属性 (TRUE) ; foreach (属性中的对
    象o)
    {
        如果 (O.GETType () == Type (公式Avabiable属性) ) 公式
        Apple, AdP (p) ;
    }
}

返回公式公式, toRayay. () ;
}
}

```

公式计算的自动化：第三部分

(工程进行中...)

提示和已知问题

高强

在一个excel文件中保留少量工作表。让我们考虑一个excel文件的情况，该文件在一个excel文件中包含超过二十个工作表，并且您为每个工作表生成了所有ScriptableObject资产文件。并且您已在同一个excel文件中创建了一个新工作表，然后尝试生成必要的脚本文件。怎么了？即使您只创建了一个工作表，并且只想将数据导入到新工作表中，但Quicksheet会尝试重新导入所有工作表中的所有数据，因为从excel查询数据到Unity重新导入完成的新创建的ScriptableObject。所以请记住，使用包含太多工作表的excel文件可能会很慢。

设置OAuth2以访问Google云端硬盘

自2015年5月5日起，Google已更改了身份验证方案。现在需要OAuth2。设置此访问[谷歌开发者控制台](#)，创建一个新项目，启用Drive API，创建“服务帐户”类型的新客户端ID并下载json文件。请参阅上面的OAuth2 Google服务帐户部分[Google电子表格Howto](#)页面了解更多详情。

请参阅页面以获取设置凭据并获取OAuth2'client_ID'和'client_secret'。

通过子树添加QuickSheet

您可以通过子树将QuickSheet添加到您的github项目，如下所示：

```
Git子树添加-前缀=资产/QueQueStHTPSE:/Github.COM/YouthGiuButhActudio/YouSurj.Git QuiCube
```

它在Assets unity project文件夹下创建QuickSheet文件夹，然后将所有必要文件放在QuickSheet文件夹下。

对于远程存储库的任何更改都可以轻松地使用git子树拉动，如下所示：

```
Git子树拉-前缀=资产/QueQueStHTPSE:/Github.COM/KimSAMA/UNITY-QuijSet
```

常问问题

问：我可以在运行时从Google表格中提取数据吗？

答：不，Unity-Quicksheet是一个在编辑时轻松处理数据的工具。从技术的角度来看，从Excel或Google电子表格中检索的所有数据都是以资产文件的形式存储的↑ScriptableObject派生类对象，以便于序列化和反序列化。

Unity的↑ScriptableObject对于持久性数据而言，这不是一个，这意味着无法序列化任何运行时更改的数据。Unity-Quicksheet不是为此目的而发明的。因此，如果您拥有的数据可以在运行时更改并且应该是持久性的，那么您应该考虑其他的東西，例如json用于此目的。

有许多双向序列化格式，例如JSON，BSON或XML等。决定完全取决于您的平台和/或要求。

另一方面，在大多数情况下，没有理由在运行时从Excel获取数据。不太可能是Excel，在运行时从Google云端硬盘连接和获取数据可能很有用，但为了与Excel保持相同的界面，Unity-Quicksheet也没有为Google电子表格提供。（虽然将来可能会改变）

问：它是否支持所有平台？

答：是的，它可以在指定为Unity的构建目标平台的任何平台上运行，除了一个限制，即Web Player构建目标设置上的Google电子表格。

由于Unity方面的安全限制，Google Spreadsheet插件无法在Unity网络播放器的安全沙箱中使用。因此，您应该在“构建设置”中将平台设置更改为“独立”或其他内容，例如“iOS”或“Android”平台，以使其正常工作。

问：我可以将Unity编辑器中的任何更改数据保存回Excel或Google电子表格吗？ 答：不，你可以。保持对Excel或Google驱动器的更改，并为数据流提供一种方法，以便更好地进行错误验证。

问：我可以在Mac机的OSX上从Excel文件中获取数据吗？

答：是的，它在Mac机器上运行没有任何问题。请记住，将您的Excel文件保存为“.xls”而不是Mac机器上的“.xlsx”文件，否则您可能会收到错误消息“无法读取内容类型部分！”

问：我可以使用“Open Office”而不是Excel吗？

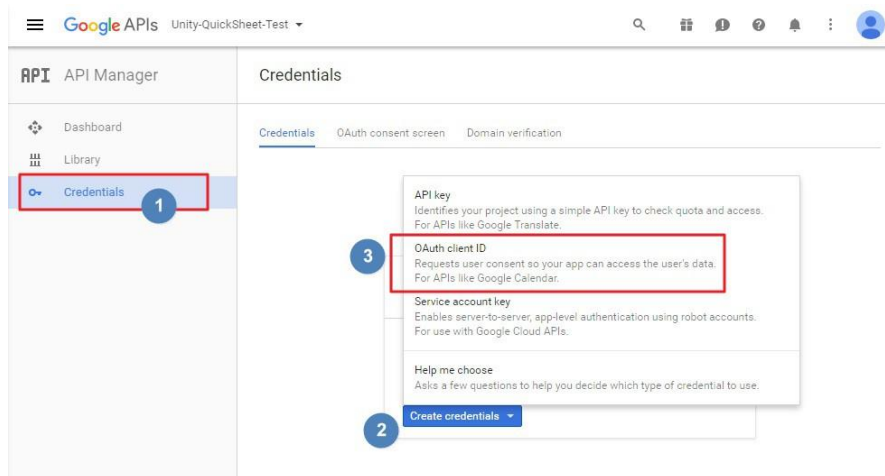
答：是的，两者都得到’.xls’文件的相同结果。

设置OAuth2以访问Google云端硬盘

Google API上的OAuth2设置

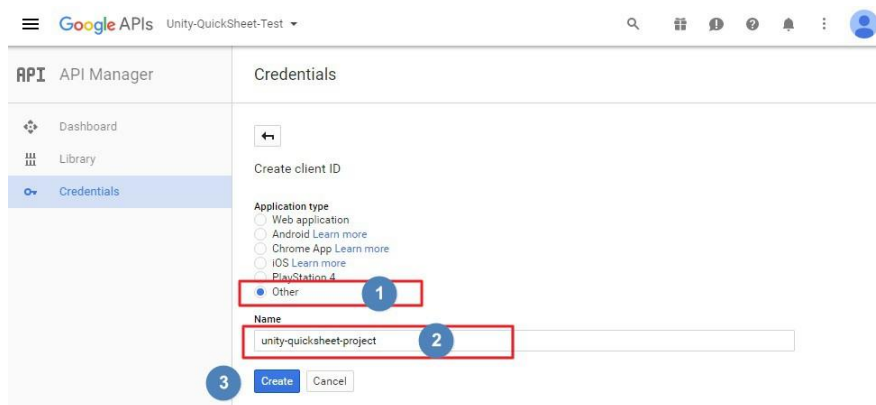
要访问Google驱动器，尤其是Unity-Quicksheet的Google电子表格，它需要OAuth2身份验证。设置此访问Google API页面并创建一个新项目。

接下来描述了获取oauth2凭证所必需的'client_ID'和'client_secret'令牌的方法。

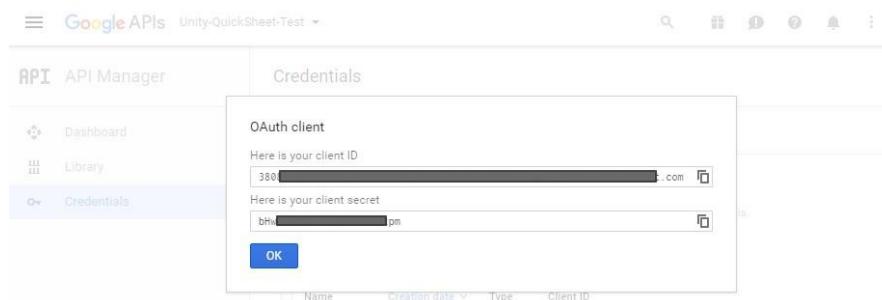


选择1) 左侧面板上的凭证和顶部菜单上的Credential选项卡，然后单击2) 创建凭证按钮，打开一个对话框，显示您可以创建的凭证类型。

请注意，您应该选择3) OAuth客户端ID，否则您可能具有错误的json文件格式。

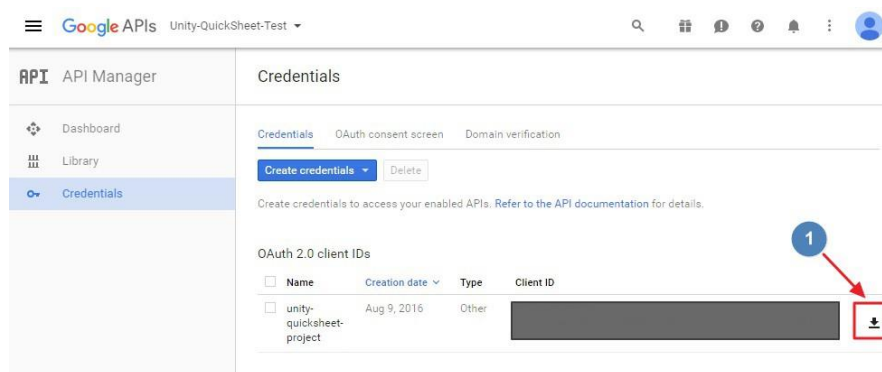


选择1) other for Application type和2) 在Name字段中插入一个正确的名称，然后按3) 'Create' 按钮将创建一个凭证。



现在，您可以看到“client_ID”和 “client_secret”用于新创建的凭据。

此时，您可以直接将’ client_ID’ 和’ client_secret’ 复制并粘贴到UnityQuicksheet的google设置中，或者下载包含’ client_ID’ 和’ client_secret’ 的oauth2 json文件，这在Unity-Quicksheet设置中是必需的。



最后，单击1) 下载json文件的下载图标，其中包含’ client_ID’ 和’ client_secret’。

Unity-Quicksheet上的OAuth2设置

谷歌认证的事情已经完成，现在是时候设置在Unity端访问谷歌电子表格的遗骸。

请参阅上面的OAuth2 Google服务帐户部分[Google Spreadsheet Howto](#)页面更多细节。

代码模板

Unity-Quicksheet使用各种模板文件自动生成从电子表格导入数据所需的“.cs”脚本文件。

所有模板文件都是简单的ascii'.txt'文件。因此，如果您希望它生成其他形式的脚本，您可以轻松更改。

生成的脚本根据其目的分为两类。一个用于运行时，另一个用于编辑器脚本。

运行	编辑
数据控件	后处理程序
TXT	脚本对象
TraceTable对象类	N/A

编辑器脚本将放在名为“Editor”的文件夹下，这是一个Unity特定文件夹。

使用自定义模板

Unity-QuickSheet的一个灵活功能是您可以将模板文件更改为生成的脚本文件，无论您想要什么。

但有一点需要牢记。

当Unity-QuickSheet生成脚本文件时，将替换其前缀为'\$'的单词。所以你应该小心，如果你改变任何单词'\$'，如果没有，生成的脚本文件将不会被编译与获取语法错误。

如果您熟悉Unity编辑器脚本，请尝试从简单的电子表格导入数据并查看生成的脚本文件。最后一件事是自我解释。

参考

数据控件

关键词	说明
\$类名	DataClass类的类名。对应于电子表格的名称。（或谷歌电子表格中的“工作表”名称）
\$成员字段	对应于电子表格中“列标题”的名称。

TXT

关键词	说明
\$字段名	“DataClass”类的成员字段名称。
\$CABOLD字段名	与\$ FieldName相同但是大写。

TraceTable对象类

关键词	说明
\$类名	派生ScriptableObject类的“类名”
数据名	它应该与‘DataClass.txt’模板文件的\$ ClassName相同。

脚本对象

关键词	说明
WorkSeCub类名称	对应于电子表格的名称。（或谷歌电子表格中的“工作表”名称）
\$类名	与‘ScriptableObjectClass’类的\$ ClassName相同。
数据名	与‘DataClass’类的\$ ClassName相同。

后处理程序

关键词	说明
AsExtPB处理器类	
\$类名	与‘ScriptableObjectClass’类的\$ ClassName相同。
数据名	与‘DataClass’类的\$ ClassName相同。

参考

- [Unity序列化](#)在Unity的论坛上了解序列化机制的详细信息。
- [格达达布](#)用于从Google电子表格中检索数据。注意[格达达布](#)略微修改以支持枚举类型。
- ~~[曝光特性](#)用于在Unity3D的检查器视图上轻松公开电子表格的变量并让它[格达达布](#)通过get / set访问器访问。（在v. 1.0.0上删除。）~~
- [恩波伊](#)用于读取xls和xlsx文件。
- Google Data SDK的所有“*.dll”文件均可在以下位置获得[谷歌数据API SDK](#) 有关net
- 2.0的Newtonsoft.Json源代码，请访问：[这里](#)
- [团结](#)，[Google资料](#)，我之前将电子表格数据导入Unity的努力。