Excel2Json2Object

Excel 表格 转 Json 文档 转 Object

By xudawang

目录

1. 插件介绍

此插件用于把 Excel 表格转化为对象。

这个过程非常简单，你可以根据此文档，非常简单的明白此插件应该如何使用。

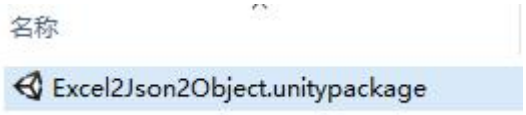
[注意：此插件不支持 float 类型，浮点数请使用 double 类型!!!!]

2. 第一步：导入.unitypackage 资源包

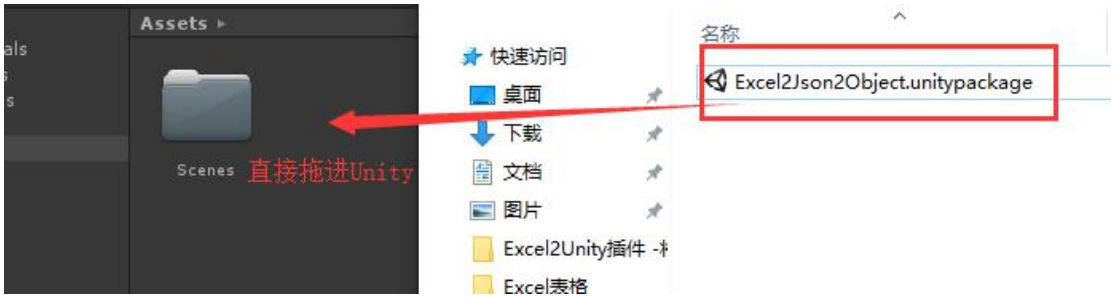
1. 首先，下载插件。

   插件下载地址：暂无

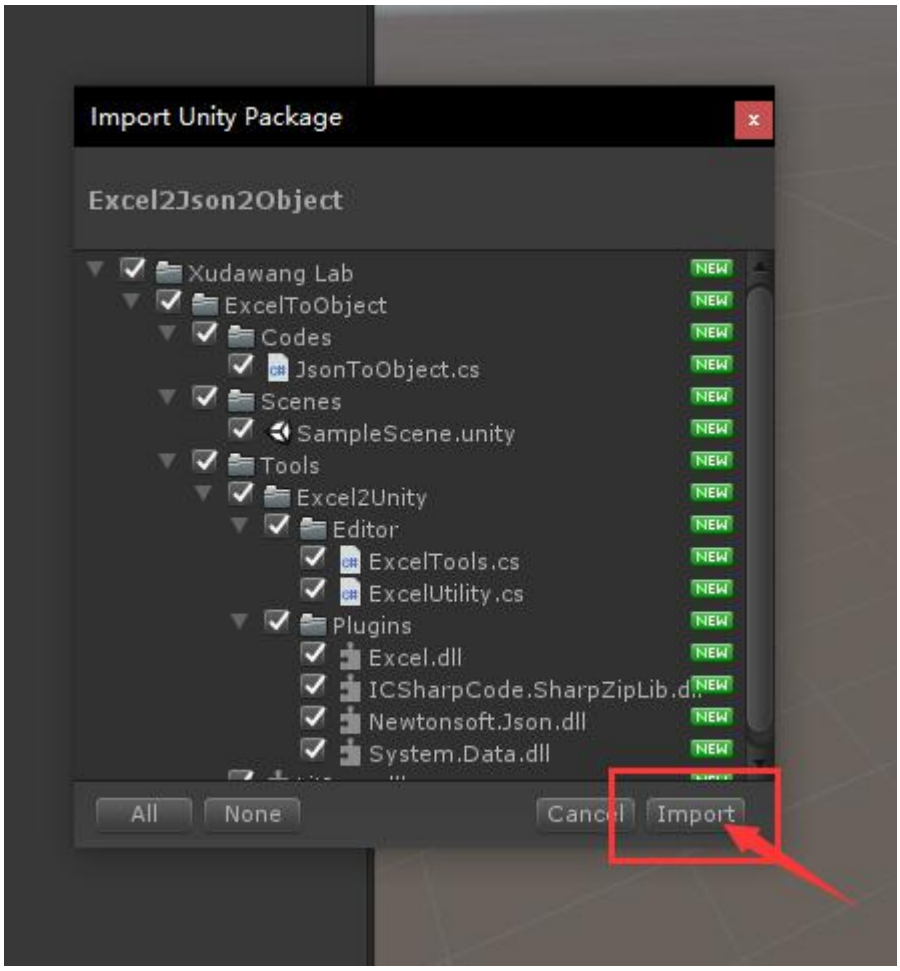
   你下载到的是一个.unitypackage 格式的文件。



2. 直接把文件拖进 Unity 中



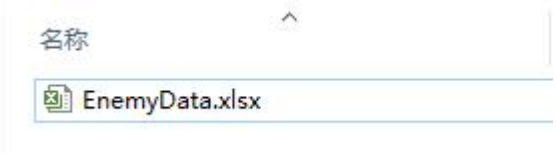
3. 然后单击[Import]按钮，就可以将此插件导入到你的 Unity 工程中啦！



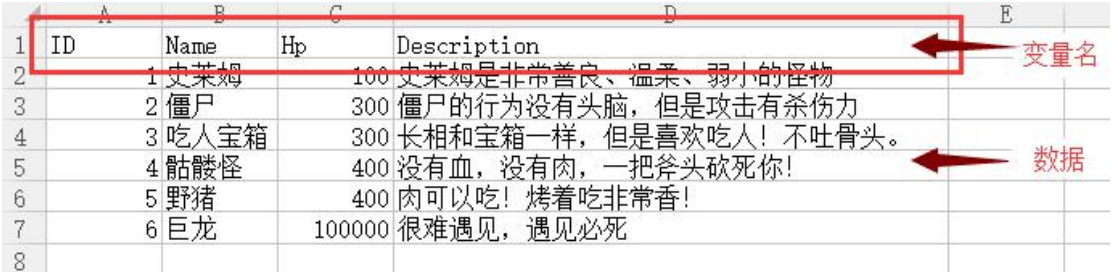
3. 第二步：创建一个 Excel 表格

创建 Excel 表格非常简单，但是你需要注意以下几点：

1. Excel 表格文件的后缀名必须是“.xlsx”



2. 表格中的第一行是变量名



	A	B	C	D
1	ID	Name	Hp	Description
2	1	史莱姆	100	史莱姆是非常善良、温柔、弱小的怪物
3	2	僵尸	300	僵尸的行为没有头脑，但是攻击有杀伤力
4	3	吃人宝箱	300	长相和宝箱一样，但是喜欢吃人！不吐骨头。
5	4	骷髅怪	400	没有血，没有肉，一把斧头砍死你！
6	5	野猪	400	肉可以吃！烤着吃非常香！
7	6	巨龙	100000	很难遇见，遇见必死
8				

3. 表格中的每一行都是一条数据（除了第一行以外）

比如我们现在有一个敌人的数据表。

我们知道一个敌人有编号(id)，名字(name)、血量(hp)、以及一些说明(description)。

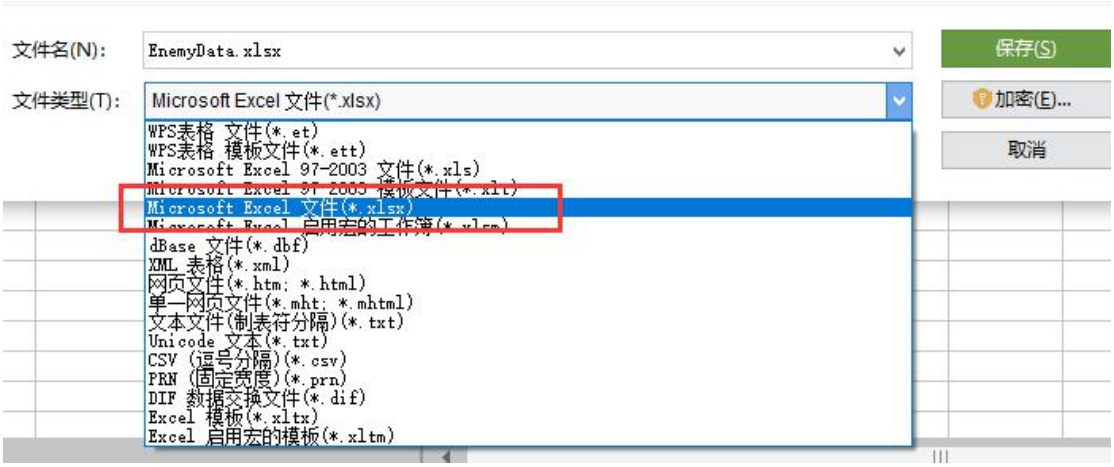
所以我们就把每种敌人的信息填写到表格里，就像是下面这样。

注意：表格的第一行是变量名！

	A	B	C	D
1	ID	Name	Hp	Description
2	1	史莱姆	100	史莱姆是非常善良、温柔、弱小的怪物
3	2	僵尸	300	僵尸的行为没有头脑，但是攻击有杀伤力
4	3	吃人宝箱	300	长相和宝箱一样，但是喜欢吃人！不吐骨头。
5	4	骷髅怪	400	没有血，没有肉，一把斧头砍死你！
6	5	野猪	400	肉可以吃！烤着吃非常香！
7	6	巨龙	100000	很难遇见，遇见必死

4. 现在，我们的表格就算是完成啦！

保存的时候注意，一定要保存成“.xlsx”格式！



4. 第三步：创建数据类

我们需要创建一个数据类。

注意，此数据类中的每一个变量的名字，都必须和你的 Excel 表格中的第一行 里的变量名，

一模一样！！！！(包括大小写，也必须一模一样！)

我们还是拿之前的那个表格举例子：

这是表格里的内容：

	A	B	C	D	E
1	ID	Name	Hp	Description	
2	1	史莱姆	100	史莱姆是非常善良、温柔、弱小的怪物	
3	2	僵尸	300	僵尸的行为没有头脑，但是攻击有杀伤力	
4	3	吃人宝箱	300	长相和宝箱一样，但是喜欢吃人！不吐骨头。	
5	4	骷髅怪	400	没有血，没有肉，一把斧头砍死你！	
6	5	野猪	400	肉可以吃！烤着吃非常香！	
7	6	巨龙	100000	很难遇见，遇见必死	
8					

变量名

数据

这是我们创建的数据类：

```
0 个引用
public class EnemyData
{
    public int ID;
    public string Name;
    public int Hp;
    public string Description;
}
```

类名可以随便填写

但是变量名一定要和表格里的一模一样！

这样，我们的数据类就创建好啦！

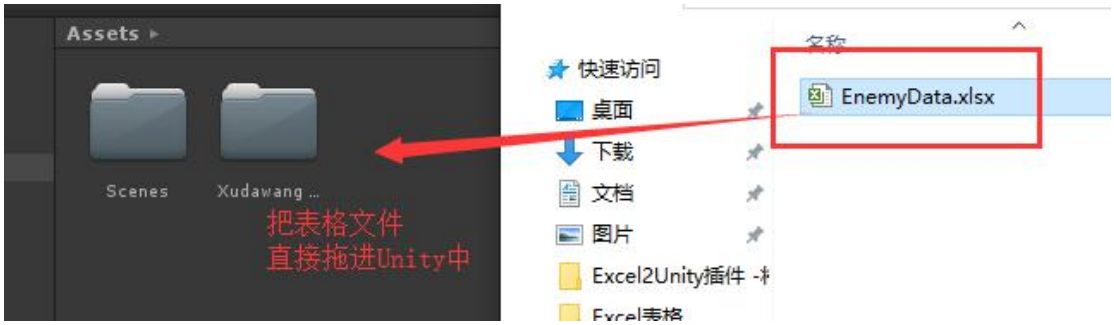
5. 第四步：把 Excel 表格转换为 Json 文件

这一步，要非常感谢 Excel2Unity 插件的制作者！

这个插件可以让我们很简单的把 Excel 表格，转换为 Json 文件。

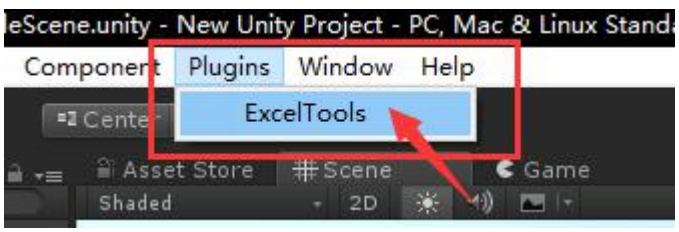
怎么做呢？

1. 首先，把我们之前制作好的 Excel 表格，放进 Unity 中。

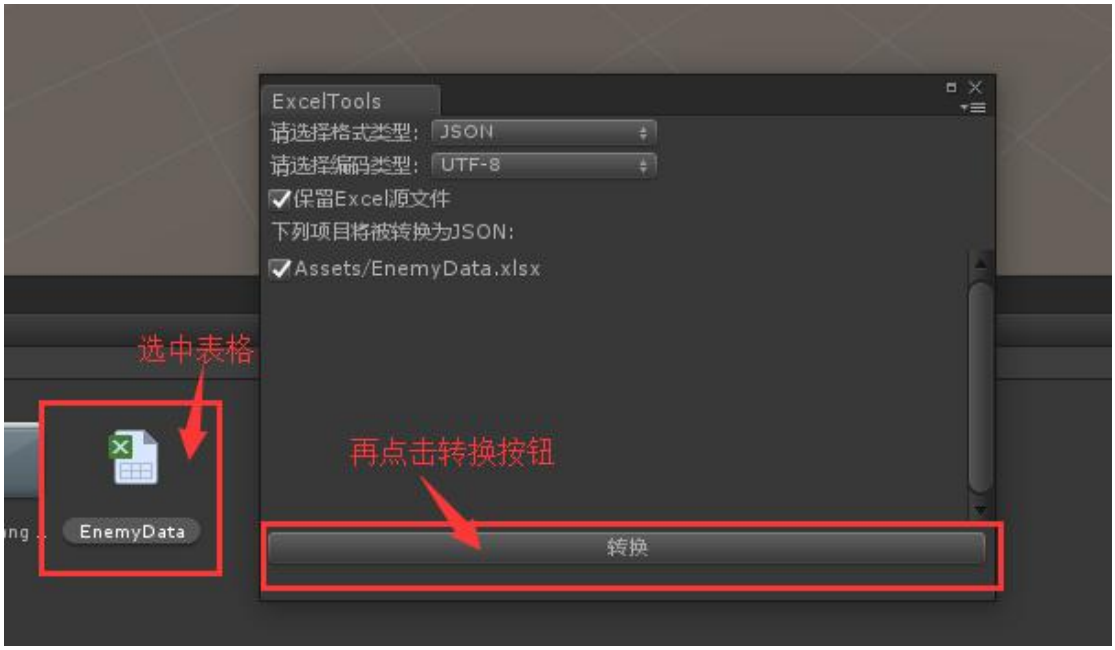


2. 然后选中这个 Excel 表格

3. 点击上方菜单栏中的 Plugins → ExcelTools



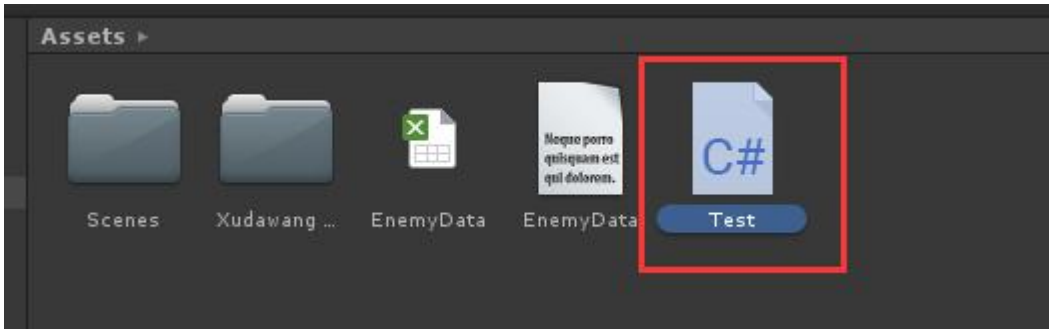
4. 然后点击[转换] 按钮



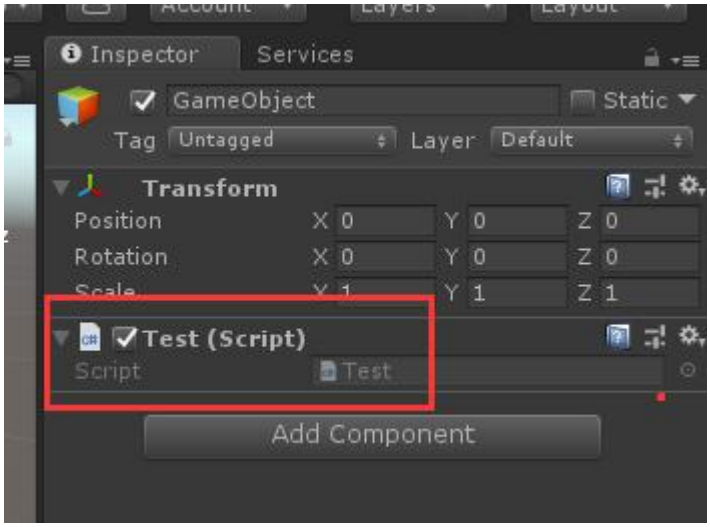
5. 这样你就会发现，旁边多出来一个.txt 文件，这个就是我们的 Json 文件啦！

6. 第五步：把 Json 文件转换为 Object 对象

1. 首先，我们新建个脚本，就叫做“Test.cs”好了！



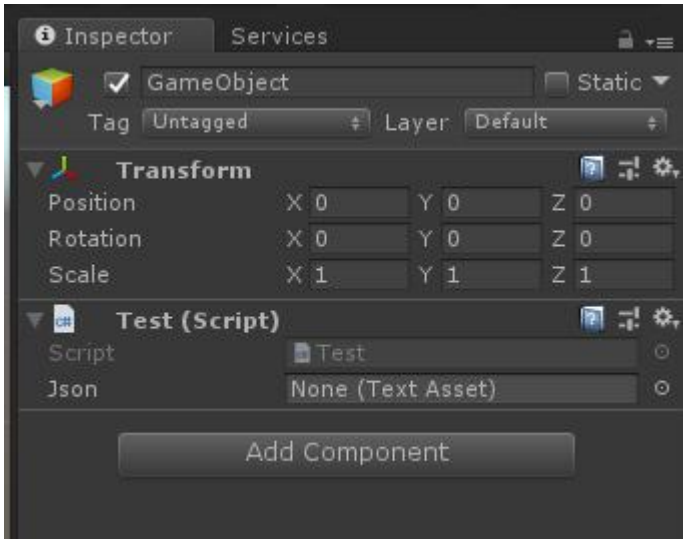
2. 然后我们把脚本随便挂到某个游戏物体上



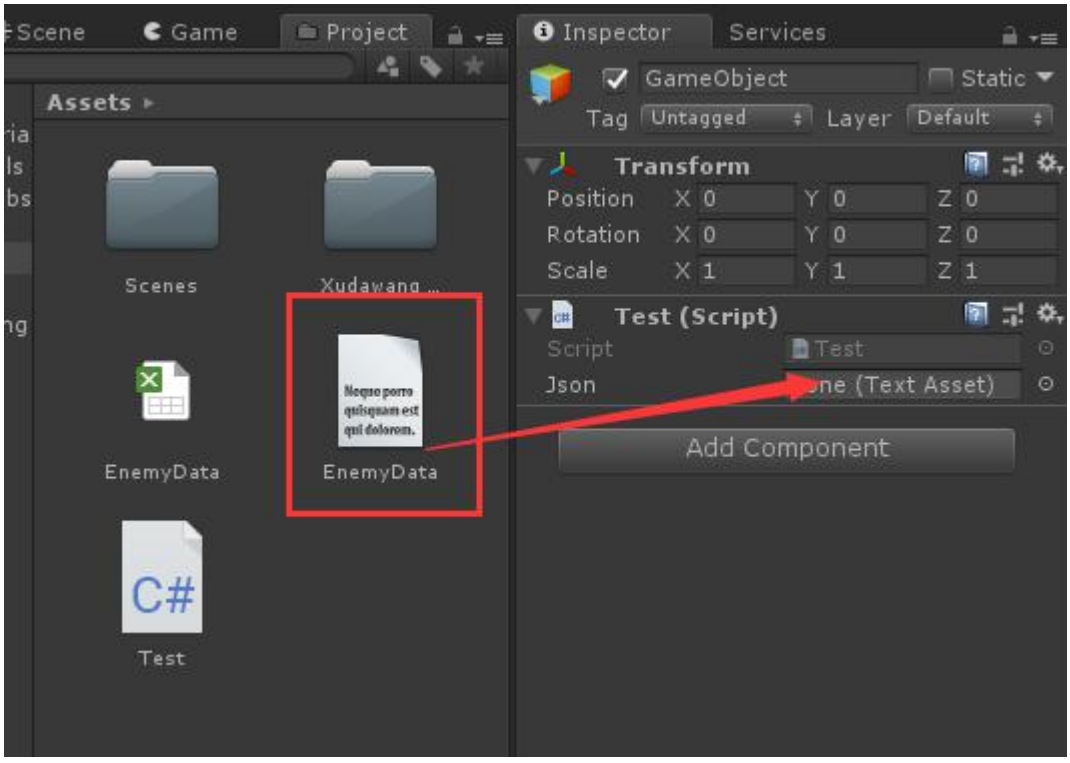
3. 然后，我们在 Test.cs 脚本中，声明一个变量。

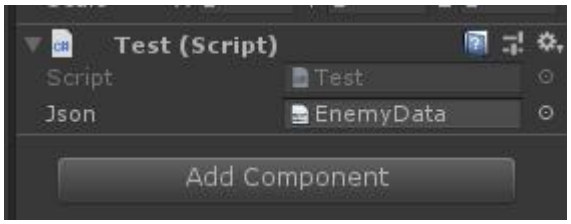
```
public class Test : MonoBehaviour
{

    public TextAsset json;//声明一个存放.txt文件的变量
}
```

4. 此时，我们到 Unity 的 Inspector 面板上查看，就可以看到我们刚刚声明的变量啦！



5. 把我们的 Json 文件，直接拖到这个变量中来！

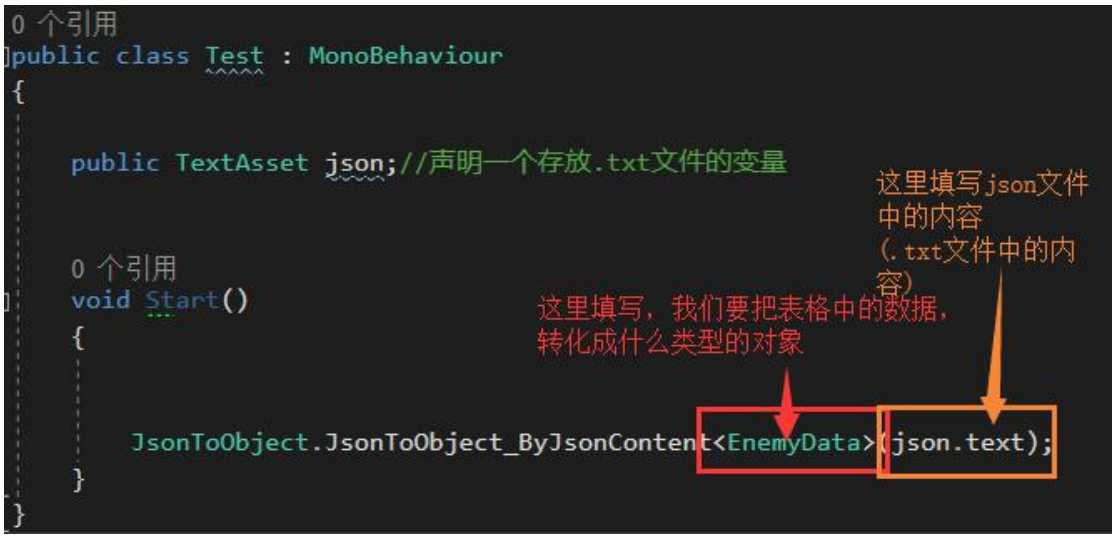




6. 我们继续编写我们的代码

```
0 个引用
public class Test : MonoBehaviour
{

    public TextAsset json;//声明一个存放.txt文件的变量


    0 个引用
    void Start()
    {

        //此方法用于将一个Json文件的内容，转换成一个类型的对象
        JsonToObject.JsonToObject_ByJsonContent<EnemyData>(json.text);
    }
}
```

下图，详细解释了此方法：



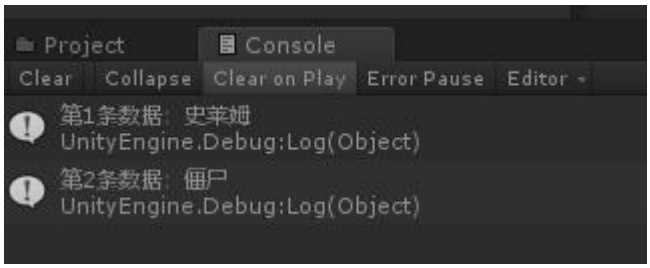
7. 我们注意，此方法是有返回值的。返回值，就是我们表格中的数据

```
//此方法用于将一个Json文件的内容，转换成一个类型的对象
//此方法的返回值，就是我们存放Excel表格中所有数据
//(一个EnemyData类的对象，就是一条数据)
List<EnemyData> datas
    = JsonToObject.JsonToObject_ByJsonContent<EnemyData>(json.text);
```

8. 测试一下吧!

```
public class Test : MonoBehaviour
{

    public TextAsset json;//声明一个存放.txt文件的变量


    0 个引用
    void Start()
    {

        //此方法用于将一个Json文件的内容，转换成一个类型的对象
        //此方法的返回值，就是我们存放Excel表格中所有数据
        //(一个EnemyData类的对象，就是一条数据)
        List<EnemyData> datas
            = JsonToObject.JsonToObject_ByJsonContent<EnemyData>(json.text);

        //输出
        Debug.Log("第1条数据: " + datas[0].Name);
        Debug.Log("第2条数据: " + datas[1].Name);
    }
}
```

测试结果:



打完收工!!!

7. API

JsonToObject.JsonToObject_ByJsonFile() 静态方法：

用于把一个 Json 文本文件，转成一个对象(Object)

参数：Json 文本文件的地址(需要加上文件名的后缀名)

返回值：泛型类型的列表

```
/// <summary>
/// 把一个Json文本文件，转成一个对象(Object)
/// </summary>
/// <typeparam name="T">对象的类型</typeparam>
/// <param name="filePath">Json文本文件的地址(需要加上文件名和后缀名)</param>
/// <returns></returns>
0 个引用
public static List<T> JsonToObject_ByJsonFile<T>(string filePath)...
```

JsonToObject.JsonToObject_ByJsonContent() 静态方法：

用于把一个 Json 格式的文本，转成一个对象(Object)

参数：Json 文本文件中的内容

返回值：泛型类型的列表

```
/// <summary>
/// 把一个Json格式的文本，转成一个对象(Object)
/// </summary>
/// <typeparam name="T">对象的类型</typeparam>
/// <param name="filePath">Json文本中的内容</param>
/// <returns></returns>
2 个引用
public static List<T> JsonToObject_ByJsonContent<T>(string conntent)...
```

8. 常见问题

1. 请不要使用 float 类型的变量!

此插件不支持 float 类型，浮点数请使用 double 类型!!!!

(原因：因为 LitJson 只支持 double 类型。)