

ExcelExporter 使用说明

目录

1、简介	2
2、支持平台	2
2.1-Unity Editor (Windows + macOS)	3
2.2-macOS 应用程序	3
2.3-Windows 应用程序	4
3、导出数据格式	4
3.1-C#+二进制	4
3.2-Json	6
3.3-Lua	8
3.4-更多	8
4、数据类型	9
4.1-Excel 表格式	9
4.2-基本数据类型	10
4.3-数组	10
4.4-localizedstring	10
5、App 界面操作	11
5.1-路径选择	11
5.2-导出文档选择	12
5.3-导出格式选择	13
5.4-偏好/配置保存功能	13
5.5-导出	13
5.6-即时日记显示	14

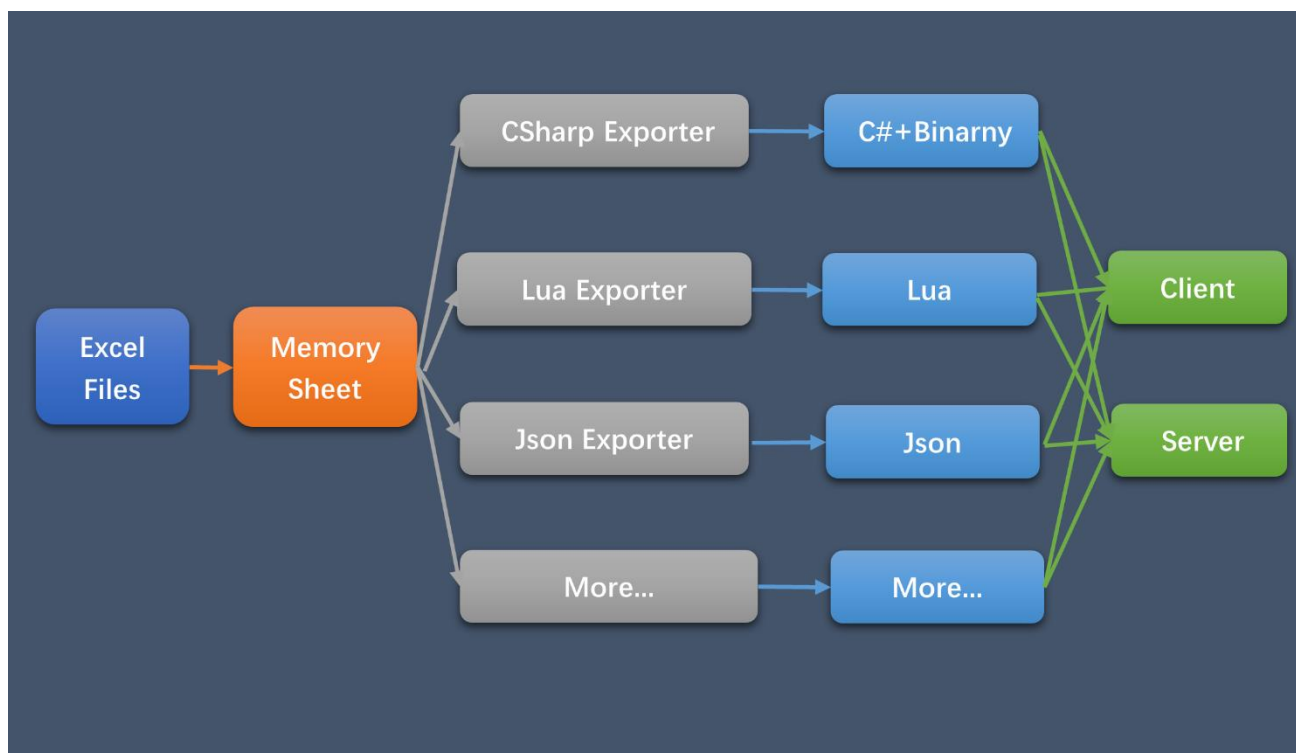
1、简介

ExcelExporter 是一个为了解决实际项目应用而开发的工具。在以往的项目开发经验中，使用的 Excel 文件导出数据工具一般都有代码混乱，难以维护，跨平台支持差（一般不支持 macOS），多种编程语言导出困难等问题。ExcelExporter 就是为了解决这些问题而开发的。

ExcelExporter 同时支持 macOS 和 Windows 的 Unity Editor 和 App 使用方式。它目前支持导出 C#+二进制文件，Json，Lua 格式的代码和数据，由于它良好的插件式设计，你可以方便的添加新的导出格式，而不用担心影响项目的其他地方。支持 byte、int、uint、long、ulong、float、double、string 等数据类型以及以他们为元素的数组，即 int[]、float[]、string[]等。ExcelExporter 还支持一种叫 localizedstring 的数据类型，用于支持文本的多国语言本地化功能。

你可以把 ExcelExporter 当成独立项目发布 App 使用，也可以抽取功能模块，集成到你自己的项目中，在 Unity Editor 中使用。

ExcelExporter 的模块流程设计如图所示：



2、支持平台

ExcelExporter 使用 Unity 开发，原生支持发布 App 跨平台使用。本项目使用的 Plugins 和代码符合 Unity 使用的 .NET Standard 2.0 跨平台标准。

2.1-Unity Editor (Windows + macOS)

ExcelExporter 代码支持在 Windows 和 macOS 的 Unity Editor 中使用，macOS 的用户也可以支持导出 Excel 数据了，不需要 VBA。

2.2-macOS 应用程序

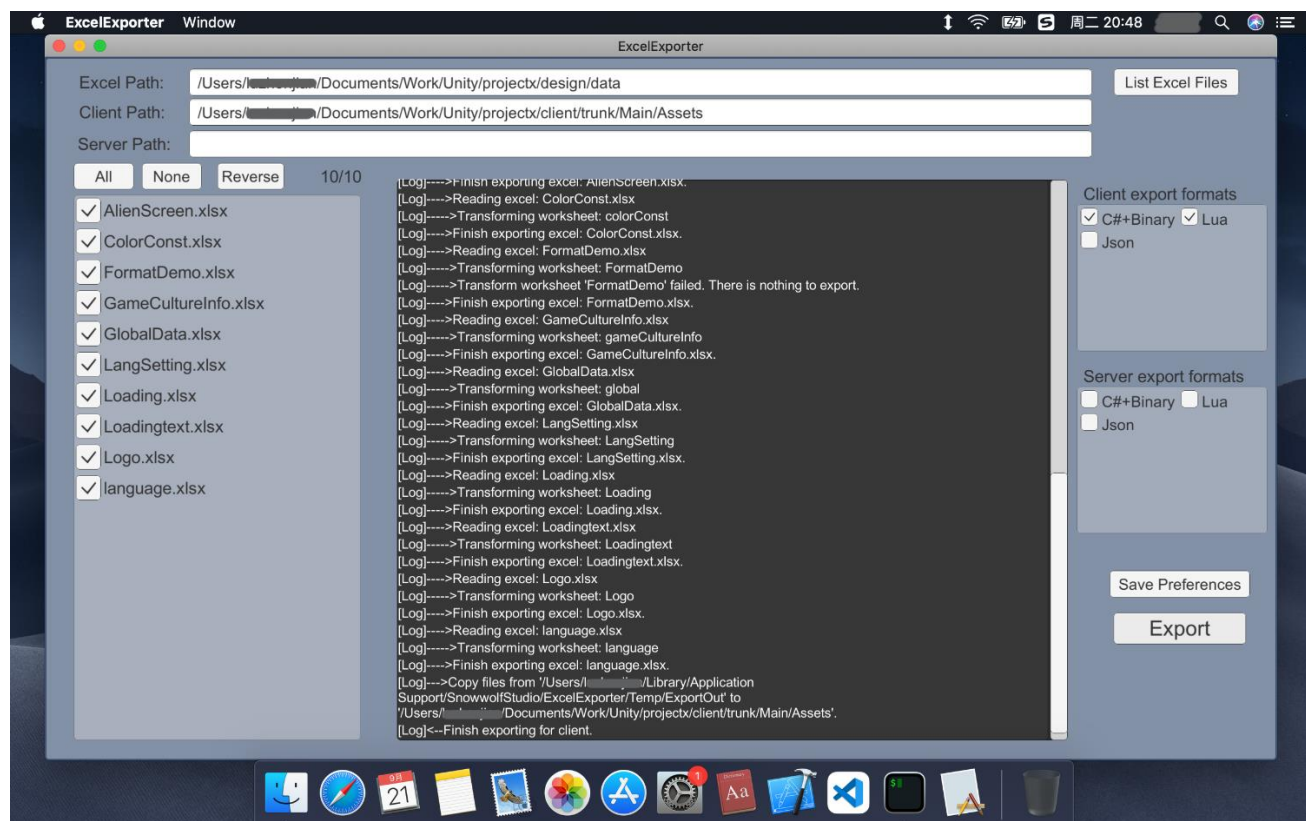


图 2-1 macOS 应用程序示例

ExcelExporter 支持发布为 macOS 应用程序使用。注意 Unity 的 PlayerSettings->Api Compatibility Level 不要设置为 .NET 3.5，它已经在 Unity2018 以上版本被弃用。

2.3-Windows 应用程序

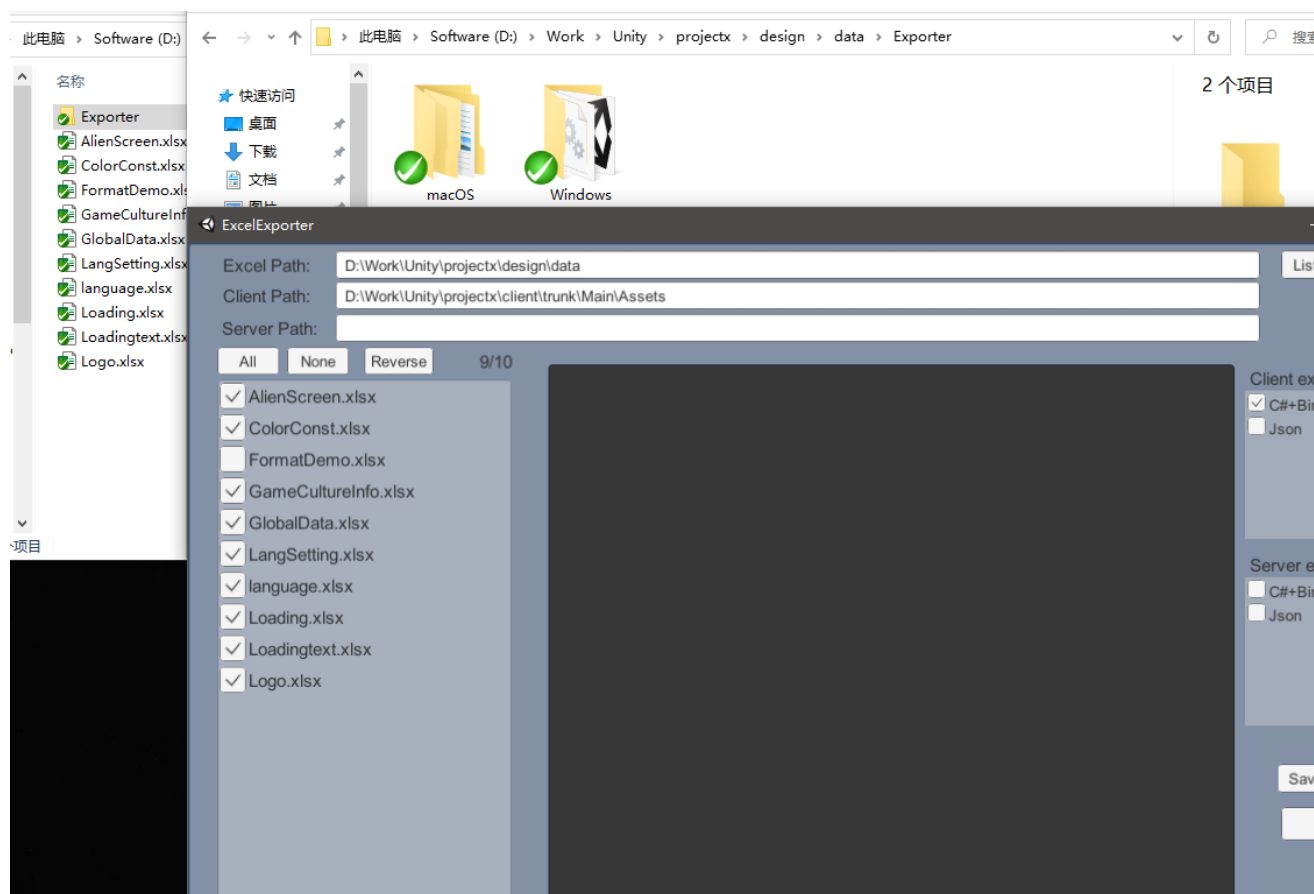


图 2-2 Windows 应用程序和目录结构示例

ExcelExporter 支持发布为 Windows 应用程序使用。注意 Unity 的 PlayerSettings->Api Compatibility Level 不要设置为 .NET 3.5，它已经在 Unity2018 以上版本被弃用。

3、导出数据格式

3.1-C#+二进制

此导出格式的形式为，将 Excel 表的数据结构解析为 C# class 类，而表的数据写入二进制文件中。这样，在不更改 Excel 表结构的情况下，C# 的代码不会变更，而只更新二进制文件，这样会减少代码量和编译时间，特别是在使用 IL2CPP 的时候。二进制文件为纯数据，无冗余信息，极大减小了数据大小，增加读取速度。

项目的 SWExcelExporterTest/ExporterOutput/CSharp 中，展示了使用例子。如图所示。

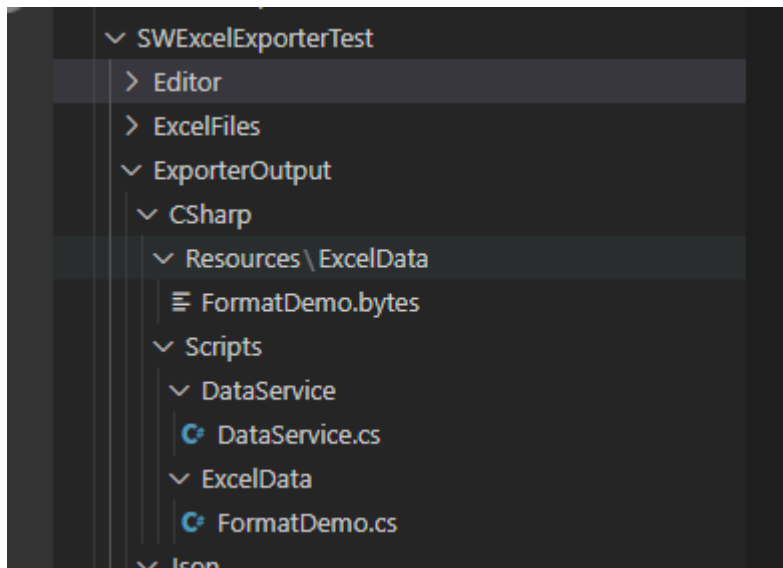


图 3-1 C#+二进制文件结构分布

```
Assets > SWExcelExporterTest > ExporterOutput > CSharp > Scripts > ExcelData > FormatDemo.cs > {} ExcelData > ExcelData.FormatDemo > RefreshLocalizationValues()
0 references
42 public static Item GetItem(string key)
43 {
44     Instance.m_Items.TryGetValue(key, out Item foundItem);
45     #if UNITY_EDITOR
46     if (foundItem == null)
47     {
48         UnityEngine.Debug.LogWarningFormat("{0} do not contains item of key '{1}'.", Instance.sheetName, key);
49     }
50     #endif
51     return foundItem;
52 }
53
0 references
54 public static IEnumerable<KeyValuePair<string, Item>> GetDict()
55 {
56     return Instance.m_Items;
57 }
58
4 references
59 private Dictionary<string, Item> m_Items = new Dictionary<string, Item>();
60
4 references
61 public string sheetName => "FormatDemo";
62
1 reference
63 private void Init()
64 {
65     byte[] bytes = DataService.GetSheetBytes(sheetName);
66     using (MemoryStream ms = new MemoryStream(bytes))
67     {
68         using (BinaryReader reader = new BinaryReader(ms))
69         {
70             reader.ReadString(); //sheetName
71
72             //Read header
73             SheetHeader sheetHeader = new SheetHeader();
74             sheetHeader.ReadFrom(reader);
75             List<SheetHeader.Item> headerItems = sheetHeader.items;
```

图 3-2 C#文件 FormatDemo.cs 示例

```
FormatDemo.bytes.hexdump X
1 | Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
2 | 00000000: 0A 46 6F 72 6D 61 74 44 65 6D 6F 0E 00 00 00 03 .FormatDemo.....
3 | 00000010: 6B 65 79 06 73 74 72 69 6E 67 06 69 6E 74 56 61 key.string.intVa
4 | 00000020: 6C 03 69 6E 74 07 75 69 6E 74 56 61 6C 04 75 69 l.int.uintVal.ui
5 | 00000030: 6E 74 07 6C 6F 6E 67 56 61 6C 04 6C 6F 6E 67 08 nt.longVal.long.
6 | 00000040: 75 6C 6F 6E 67 56 61 6C 05 75 6C 6F 6E 67 07 62 ulongVal.ulong.b
7 | 00000050: 79 74 65 56 61 6C 04 62 79 74 65 08 66 6C 6F 61 yteVal.byte.floa
8 | 00000060: 74 56 61 6C 05 66 6C 6F 61 74 09 64 6F 75 62 6C tVal.float.doubl
9 | 00000070: 65 56 61 6C 06 64 6F 75 62 6C 65 06 73 74 72 56 eVal.double.strV
10 | 00000080: 61 6C 06 73 74 72 69 6E 67 07 6C 73 74 72 56 61 al.string.lstrVa
11 | 00000090: 6C 0F 6C 6F 63 61 6C 69 7A 65 64 73 74 72 69 6E l.localizedstrin
12 | 000000a0: 67 08 69 6E 74 41 72 72 61 79 05 69 6E 74 5B 5D g.intArray.int[]
13 | 000000b0: 0A 66 6C 6F 61 74 41 72 72 61 79 07 66 6C 6F 61 .floatArray.floa
14 | 000000c0: 74 5B 5D 0B 73 74 72 69 6E 67 41 72 72 61 79 08 t[] stringArray.
15 | 000000d0: 73 74 72 69 6E 67 5B 5D 0D 63 6C 69 65 6E 74 4F string[] clientO
16 | 000000e0: 6E 6C 79 56 61 6C 05 66 6C 6F 61 74 06 00 00 00 nlyVal.float....
17 | 000000f0: 04 6B 65 79 31 0C 00 00 00 7B 00 00 00 C0 1D FE .key1....{...@.~
```

图 3-3 二进制文件 FormatDemo.bytes 示例

3.2-Json

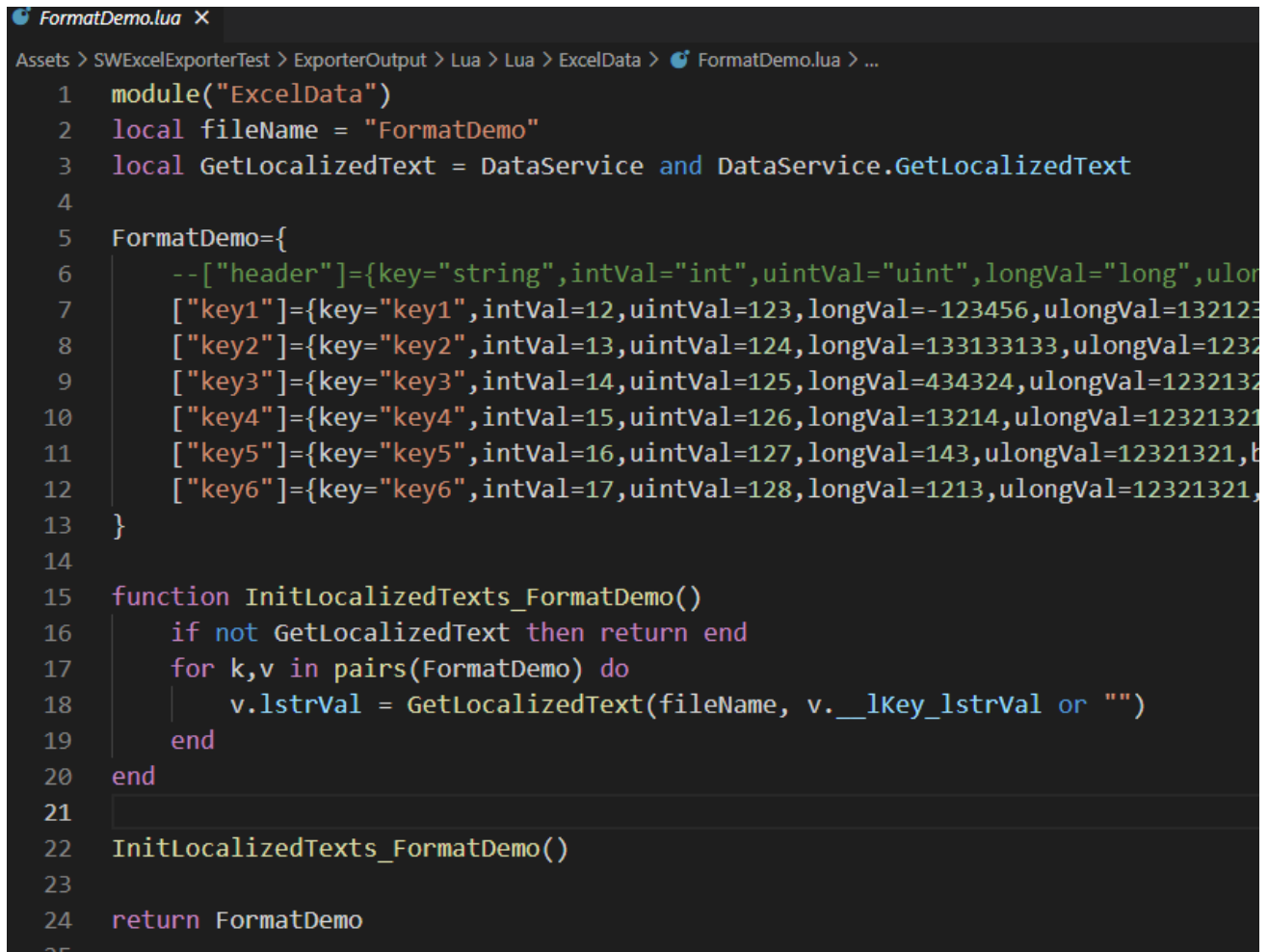
从项目的 SWExcelExporterTest/ExporterOutput/Json 目录中，可以找到 FormatDemo.json 的导出示例。如下图所示。

```
{ } FormatDemo.json X
Assets > SWExcelExporterTest > ExporterOutput > Json > { } FormatDemo.json > ...
18      },
19      "key1"      : {
20          "key" : "key1",
21          "intVal" : 12,
22          "uintVal" : 123,
23          "longVal" : -123456,
24          "ulongVal" : 13212321,
25          "byteVal" : 1,
26          "floatVal" : 123.456001281738,
27          "doubleVal" : 567.891,
28          "strVal" : "Normal string.",
29          "__lKey_lstrVal" : "Client_1000",
30          "intArray" : [
31              1,
32              2,
33              3
34          ],
35          "floatArray" : [
36              1.5,
37              2.29999995231628,
38              4.5
39          ],
40          "stringArray" : [
41              "Hello world",
42              "\" Hello world 2\""
43          ],
44          "clientOnlyVal" : 123.456001281738
45      },
46      "key2"      : {
47          "key" : "key2",
48          "intVal" : 13,
```

图 3-3 Json 文件 FormatDemo.json 示例

3.3-Lua

从项目的 SWExcelExporterTest/ExporterOutput/Lua 目录中，可以找到 FormatDemo.lua 的导出示例。如下图所示。



```
FormatDemo.lua X
Assets > SWExcelExporterTest > ExporterOutput > Lua > Lua > ExcelData > FormatDemo.lua > ...
1  module("ExcelData")
2  local fileName = "FormatDemo"
3  local GetLocalizedText = DataService and DataService.GetLocalizedText
4
5  FormatDemo={
6      --["header"]={key="string",intVal="int",uintVal="uint",longVal="long",ulongVal="ulong"}
7      ["key1"]={key="key1",intVal=12,uintVal=123,longVal=-123456,ulongVal=132123}
8      ["key2"]={key="key2",intVal=13,uintVal=124,longVal=133133133,ulongVal=12321}
9      ["key3"]={key="key3",intVal=14,uintVal=125,longVal=434324,ulongVal=12321321}
10     ["key4"]={key="key4",intVal=15,uintVal=126,longVal=13214,ulongVal=12321321}
11     ["key5"]={key="key5",intVal=16,uintVal=127,longVal=143,ulongVal=12321321,b}
12     ["key6"]={key="key6",intVal=17,uintVal=128,longVal=1213,ulongVal=12321321,}
13 }
14
15 function InitLocalizedTexts_FormatDemo()
16     if not GetLocalizedText then return end
17     for k,v in pairs(FormatDemo) do
18         v.lstrVal = GetLocalizedText(fileName, v.__lkey_lstrVal or "")
19     end
20 end
21
22 InitLocalizedTexts_FormatDemo()
23
24 return FormatDemo
25
```

图 3-4 Lua 文件 FormatDemo.lua 示例

3.4-更多

除了上述格式，你还可以通过扩展 *Exporter* 基类来实现自己需要的格式。

4、数据类型

4.1-Excel 表格式

ExcelExporter 表使用的格式可以从 SWExcelExporterTest/ExcelFiles/FormatDemo.xlsx 中找到, ExcelExporter 支持一个 Excel 文件中的多个 Excel 工作表的导出格式, 因此, 导出的文件名称以 Excel 工作表的名称来命名, 而不是以 Excel 文件名来命名。

P3	//[ClientOnly]" marks indicate this column will only be exported to client.										
	A	B	C	D	E	F	G	H	I	J	K
1											
2	//Use "#----->" to start and "#<-----" to stop payload.										
3	#-----> //use (//) to comment this column										
4	key	intVal	uintVal	longVal	ulongVal	byteV	floatVal	//unuse	doubleV	strVal	IstrVal
5	string	int	uint	long	ulong	byte	float		double	string	localizedstring
6	key1	12	123	-123456	13212321	1	123.456	nothing	567.89	Normal string.	Client_1000
7	key2	13	124	133,133,133	12321	0	124.456	nothing	568.89	""String with quotation mark""	Client_1001
8	key3	14	125	434324	123213213	1	125.456	nothing	569.89	String with space to trim.	Client_1002
9	key4	15	126	13214	123213213	0	126.456	nothing	570.89	" String with space to use."	Dialog_1003
10	//use (//) to comment this row										
11	key5	16	127	143	12321321	1	127.456	nothing	571.89	"Line with explicit linefeed\n and implicit linefeed"	Dialog_1004
12	key6	17	128	1213	12321321	0	128.456	nothing	572.89	你好&こんにちは&안녕하세요--CJK	Dialog_1005
13											
14											
15											
16											
17											
18											
19											
20											
21											
22	#<-----										
23											
24											
25											
26											

图 4-1 FormatDemo.xlsx 示例 1

≥ exported to client.

[illegible]

图 4-2 FormatDemo.xlsx 示例 2

4.2-基本数据类型

ExcelExporter 支持常用的基本数据类型，目前支持 byte、int、uint、long、ulong、float、double、string 等基本数据类型。你也可以在 CellValueType 中扩展自己的类型。

4.3-数组

ExcelExporter 支持 4.2 列举的基本数据类型为元素的数组，注意，在 Excel 表结构中，[数字型数组使用逗号\(,\)作为分隔符](#)，而[字符串数组使用\(\)作为分隔符](#)，这是因为在字符串中，逗号(,)非常常见，用(,)作为分隔符会带来许多不必要的麻烦。

4.4-localizedstring

ExcelExporter 定义了一种称为 localizedstring 的字符串类型，它实际上是一种字符串类型，表示了本地化字符串的 Key。

例如，如果你程序中使用了本地化的字符串，你可以按照以下流程来执行操作：

1. 在 Excel 表中用 string 类型表示本地化字符串的 Key，比如名称为 lstringKey。
2. 在运行时，读取配置该 Excel 表的 lstringKey，然后在每个调用的地方都写 GetLocalizedText(xx.lstringKey)。

LocalizedString 类型简化了这个过程，对于使用者来说，Excel 表中填写的是 Key，实际上在运行时读取到的是已经本地化后的字符串。下列各图展示了名为 lstrVal 的 localizedstring 的生成过程。

```
2 references
public string __lKey_lstrVal;
1 reference
public string lstrVal;
```

图 4-3 定义一个隐式的__lkey_lstrVal

```
else if (j == lstrValIndex)
{
    newItem.__lKey_lstrVal = reader.ReadString();
}
```

图 4-4 读取 Excel 表的字符串到__lkey_lstrVal

```
public void RefreshLocalizationValues()
{
    foreach(var kv in m_Items)
    {
        Item item = kv.Value;
        item.lstrVal = DataService.GetLocalizedText(sheetName, item.__lKey_lstrVal);
    }
}
```

图 4-5 使用__lkey_lstrVal 去获取本地化文本 lstrVal

5、App 界面操作

App 操作的流程按照 5.1-5.5 的流程顺序操作即可。

5.1-路径选择

Excel Path:	D:\Work\Unity\projectx\design\data
Client Path:	D:\Work\Unity\projectx\client\trunk\Main\Assets
Server Path:	

图 5-1 路径选择示例

Excel Path 表示存放 Excel 文档的目录。该目录下的所有 Excel 文件（仅支持.xlsx）都会被列举出来。

Client Path 表示客户端的输出路径，一般为客户端工程路径，为空表示不导出客户端文件。

Server Path 表示服务器端的输出路径，为空表示不导出服务器端文件。

注意：输入完 **Excel Path** 后程序一般会自动列举 Excel 文档，如果没有，你可以自己按 **List Excel Files** 按钮来列举 Excel 文档。

5.2-导出文档选择

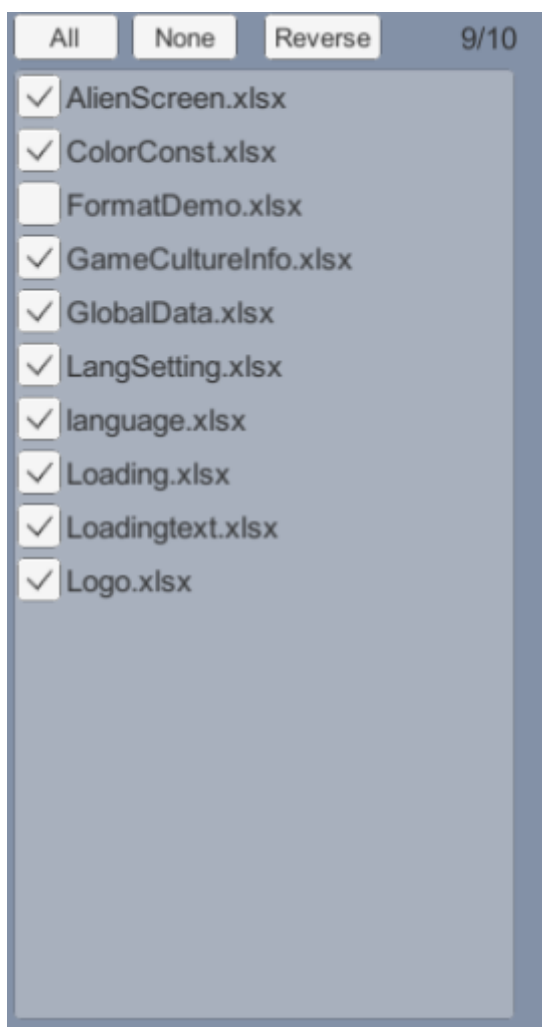


图 5-2 选择需要导出数据的 Excel 文档

5.3-导出格式选择

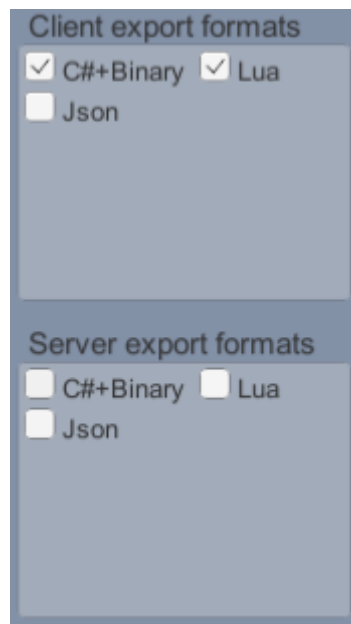


图 5-2 选择客户端和服务端各自需要的文件格式

5.4-偏好/配置保存功能

5.1~5.3 提到的功能参数都可以作为偏好保存，只需要按 **Save Preferences** 按钮即可。另外，如果按了 **Export** 按钮导出文件，上述偏好也会自动保存。

5.5-导出

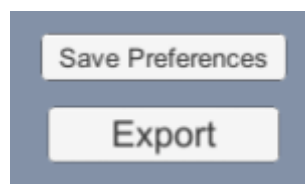
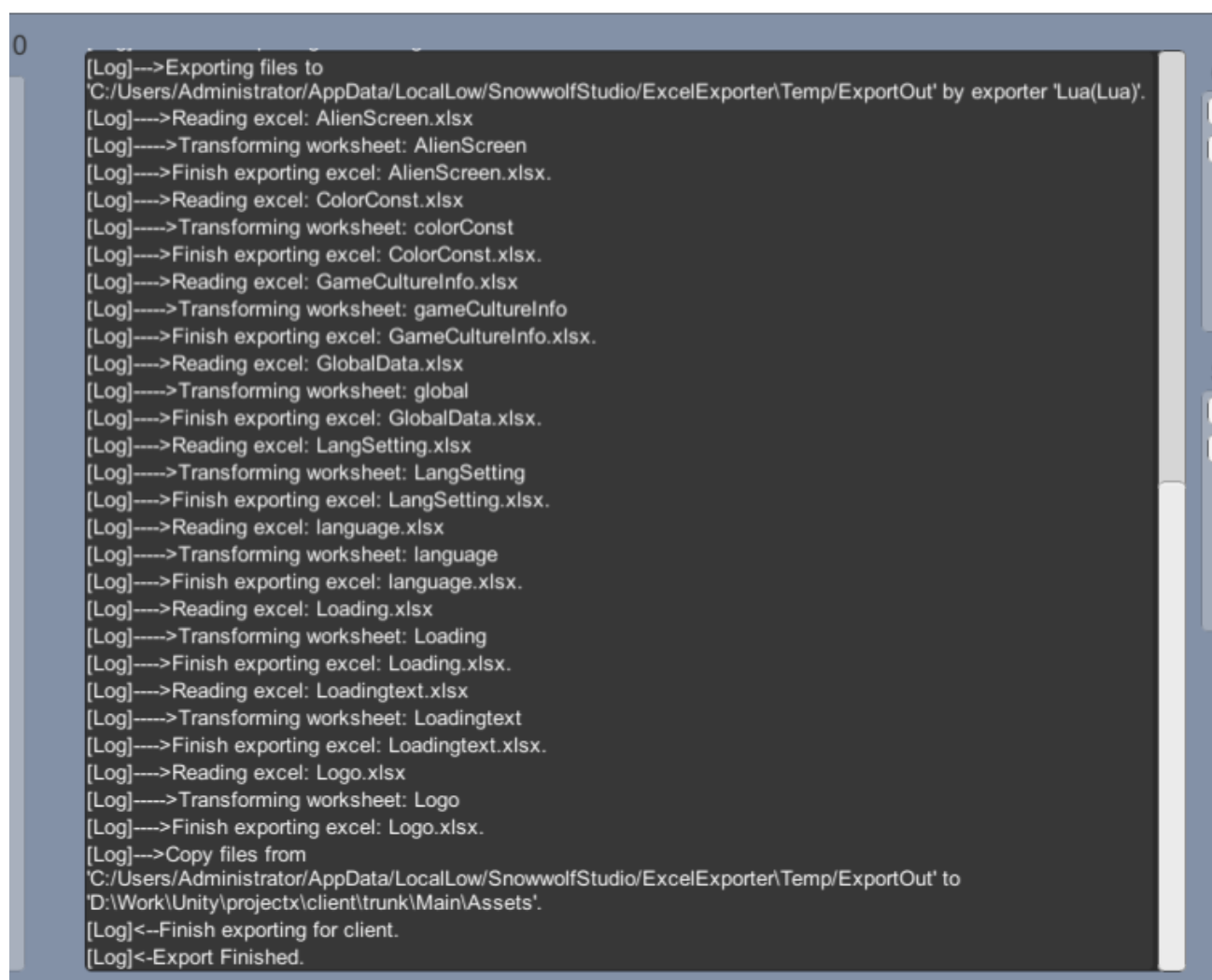


图 5-3 偏好保存和导出按钮

使用 **Export** 按钮去导出文件。输出的文件会在上述步骤指定的目录中找到。

5.6-即时日记显示



```
[Log]-->Exporting files to  
'C:/Users/Administrator/AppData/LocalLow/SnowwolfStudio/ExcelExporter/Temp/ExportOut' by exporter 'Lua(Lua)'.  
[Log]---->Reading excel: AlienScreen.xlsx  
[Log]---->Transforming worksheet: AlienScreen  
[Log]---->Finish exporting excel: AlienScreen.xlsx.  
[Log]---->Reading excel: ColorConst.xlsx  
[Log]---->Transforming worksheet: colorConst  
[Log]---->Finish exporting excel: ColorConst.xlsx.  
[Log]---->Reading excel: GameCultureInfo.xlsx  
[Log]---->Transforming worksheet: gameCultureInfo  
[Log]---->Finish exporting excel: GameCultureInfo.xlsx.  
[Log]---->Reading excel: GlobalData.xlsx  
[Log]---->Transforming worksheet: global  
[Log]---->Finish exporting excel: GlobalData.xlsx.  
[Log]---->Reading excel: LangSetting.xlsx  
[Log]---->Transforming worksheet: LangSetting  
[Log]---->Finish exporting excel: LangSetting.xlsx.  
[Log]---->Reading excel: language.xlsx  
[Log]---->Transforming worksheet: language  
[Log]---->Finish exporting excel: language.xlsx.  
[Log]---->Reading excel: Loading.xlsx  
[Log]---->Transforming worksheet: Loading  
[Log]---->Finish exporting excel: Loading.xlsx.  
[Log]---->Reading excel: Loadingtext.xlsx  
[Log]---->Transforming worksheet: Loadingtext  
[Log]---->Finish exporting excel: Loadingtext.xlsx.  
[Log]---->Reading excel: Logo.xlsx  
[Log]---->Transforming worksheet: Logo  
[Log]---->Finish exporting excel: Logo.xlsx.  
[Log]---->Copy files from  
'C:/Users/Administrator/AppData/LocalLow/SnowwolfStudio/ExcelExporter/Temp/ExportOut' to  
'D:\Work\Unity\projectx\client\trunk\Main\Assets'.  
[Log]<--Finish exporting for client.  
[Log]<-Export Finished.
```

图 5-4 即时日记窗口

可以查看导出流程打印提示，并可以查看异常、错误、警告等信息，方便跟踪问题。