**简介**

开发人员一般通过BuildPipeline函数去打包文件，然后通过WWW去下载

但是BuildPipeline打包的对象类型是有限制的，像GameObject，TextAsset这些文件是可以直接打包的，但是，如果要读取一些Unity不支持的类型，我们就需要用到ScriptableObject了。这里用打包Csv表作为例子示例一下

实现

**1. 编辑ScriptObject对象**

首先，csv表的内容就是一段字符串，又或者说是一段bytes，看程序需求，我这里把csv表的内容当成一段bytes读取，吧

所以，csv表的ScriptableObject可以这样写

**[csharp]** [view plaincopy](http://blog.csdn.net/xv_ly15/article/details/9330617)

1. **using** UnityEngine;
2. **using** System.Collections;
4. **public** **class** soCsv : ScriptableObject
5. {
6. **public** **string** fileName;
7. **public** **byte**[] content;
8. }

**2. 打包文件**

    有了Csv的ScriptableObject后，就可以开始写打包函数了

简单说一下流程：

1.获取选中的csv表对象

2.赋值到刚声明的csv表对应的ScriptableObject

3.用该ScriptableObject生成Asset

4.读取生成的Asset后打包（生成与读取asset的过程有点别扭，但只有这样获得的Object对象，才能打包成功...）

**[csharp]** [view plaincopy](http://blog.csdn.net/xv_ly15/article/details/9330617)

1. **using** UnityEngine;
2. **using** System.Collections;
3. **using** UnityEditor;
4. **using** System.IO;
5. **using** System.Collections.Generic;

8. **public** **class** BuildTest : MonoBehaviour
9. {
10. [MenuItem("Assets/Build Csv")]
11. **static** **void** BuildCsv()
12. {
13. **string** applicationPath = Application.dataPath;
14. **string** saveDir = applicationPath + "/Resources\_out/";
15. **string** savaPath = saveDir + "csv.unity3d";
17. //获取选中的对象
18. /\*关于SelectionMode参数，目前我只用过Assets,DeepAssets，其他的暂且不说
19. Assets：获取当前选中的单一对象
20. DeepAssets：获取当前选中的对象，如果选中的文件夹，则会获取到文件夹的子对象
21. \*
22. \*
23. Unfiltered  ： Return the whole selection.
24. TopLevel ：   Only return the topmost selected transform. A selected child of another selected transform will be filtered out.
25. Deep：    Return the selection and all child transforms of the selection.
26. ExcludePrefab   ： Excludes any prefabs from the selection.
27. Editable     ：Excludes any objects which shall not be modified.
28. Assets   ：Only return objects that are assets in the Asset directory.
29. DeepAssets  ： If the selection contains folders, also include all assets and subfolders within that folder in the file hierarchy.
30. \*
31. \*/
33. Object[] selections = Selection.GetFiltered(**typeof**(Object), SelectionMode.DeepAssets);
34. List<Object> outs = **new** List<Object>();
35. **for** (**int** i=0,max=selections.Length; i<max; i++)
36. {
37. Object obj = selections[i];
38. //asset path  : 相对Asset目录
39. **string** fileAssetPath = AssetDatabase.GetAssetPath(obj);
40. //判断后缀
41. **if** (fileAssetPath.Substring(fileAssetPath.LastIndexOf('.') + 1) != "csv")
42. **continue**;
44. //
45. **string** fileWholePath = applicationPath + "/" + fileAssetPath.Substring(fileAssetPath.IndexOf("/"));
47. //
48. soCsv csv = ScriptableObject.CreateInstance<soCsv>();
49. csv.fileName = obj.name;
50. csv.content = File.ReadAllBytes(fileWholePath);
52. //这一个存与读的步骤是否必要的?
53. **string** assetPathTemp = "Assets/Resources\_Local/Temp/" + obj.name + ".asset";
54. AssetDatabase.CreateAsset(csv, assetPathTemp);
56. Object outObj = AssetDatabase.LoadAssetAtPath(assetPathTemp, **typeof**(soCsv));
58. //
59. Debug.Log("package : " + outObj.name);
60. outs.Add(outObj);
61. }
63. //
64. Object[] outObjs = outs.ToArray();
65. **if** (BuildPipeline.BuildAssetBundle(**null**, outs.ToArray(), savaPath))
66. Debug.Log("build " + savaPath + " success,length = " + outObjs.Length);
67. **else**
68. Debug.LogWarning("build " + savaPath + " failed");
69. }
71. }

**3. 下载并解析内容**

**[csharp]** [view plaincopy](http://blog.csdn.net/xv_ly15/article/details/9330617)

1. **using** UnityEngine;
2. **using** System.Collections;
3. **using** System.IO;
5. **using** LumenWorks.Framework.IO.Csv;
7. **public** **class** Test : MonoBehaviour {
9. WWW w;
10. **string** path ;
11. **void** Start () {
12. //File协议主要用于访问本地计算机中的文件，所以这里还需要加file:///
13. path = "file:///" + Application.dataPath + "/Resources\_out/csv.unity3d";
14. w = **new** WWW(path);
15. }
17. // Update is called once per frame
18. **void** Update () {
19. **if** (w != **null**)
20. {
21. **if** (w.isDone)
22. {
23. Debug.Log("download " + path);
25. Debug.Log(w.assetBundle.mainAsset);
27. /\*loadall的参数如果不指定类型，很可能会读取到额外的内容
28. \* 像我测试的时候，就发现一直多出了一个名字叫soCsv的对象
29. \* 最好还是指定一下类型
30. \* \*/
31. Object[] objs = w.assetBundle.LoadAll(**typeof**(soCsv));
32. Debug.Log("count : " + objs.Length);
33. **for** (**int** i = 0, max = objs.Length; i < max; i++)
34. {
35. Object obj = objs[i];
36. soCsv csv= obj **as** soCsv;
38. //用string表做一个例子吧
39. **if** (csv.fileName != "string")
40. {
41. **continue**;
42. }
44. //
45. MemoryStream ms = **new** MemoryStream(csv.content);
46. **if** (ms == **null**)
47. {
48. Debug.LogWarning("转换csv失败");
49. **continue**;
50. }
52. StreamReader sr = **new** StreamReader(ms);
53. TextReader tr = sr **as** TextReader;
54. **if** (tr == **null**)
55. {
56. Debug.LogWarning("text reader is null");
57. **continue**;
58. }
60. CsvReader cr = **new** CsvReader(tr, **true**);
61. **if** (cr == **null**)
62. {
63. Debug.LogWarning("CsvReader is null");
64. **continue**;
65. }
67. //将字段头输出
68. **string**[] headers = cr.GetFieldHeaders();
69. **foreach** (**string** szHeader **in** headers)
70. Debug.Log(szHeader);

73. }
75. w = **null**;
76. }
77. }
78. }
79. }

http://blog.csdn.net/xv\_ly15/article/details/9330617