**Unity3D脚本化物件ScriptableObject设置资料成为AssetBundle**

Posted on 2013年04月07日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 189 次

**Unity3D**脚本化物件ScriptableObject设置资料成为AssetBundle。不管是制作游戏或应用软体，多少都会有一些系统设定值，有些人可能会使用一个文件档，依照定义好的格式，要求制作团队全部的人都依照这个格式去设定这些值，可是因为文件档的编写太过方便和自由，如果解读这个文件档的程式漏掉某些检查的地方，有可能因为人员的疏忽，弄错了空白、换行、引号、逗号等而造成解读结果错误，而且要针对各种设定资料的文档编写解读程式并转换成可用的物件成员，比较费劲，而且也不方便管理；对于自定义的资料资源，又该如何运用在AssetBundle 呢？此时就要使用到ScriptableObject。

如果把需要设定的资料值都写成script的public栏位，然后再拉到场景中的物件做设定，那么我们可能要每个场景建立时都要做一个这样的设定物件，当然，此时可以为这个物件建立个Prefab，那么每次建场景时直接拉进来就能设定了，但是，如果有一天发现其中一个共用的设定值需要变更，就必须把场景一个打开来修改；此时可能会想不如做个游戏启始设定场景，只要转换场景时，这个场景的物件不要销毁就好了，但常常一个场景中并不用使用到整个游戏全部的设定值，而且把这些设定值放在场景中的物件来使用，可携性不佳；此时，如果能在Project view里的资源档案设定好，游戏进行中再读入可能会方便些，但可能又会有上一段所提到的文件格式问题；有鉴于以上顾虑到的部份，我们可以使用ScriptableObject在Project view建立专用的Asset来做设定，并将它存放在Resources资料夹中，于是，游戏中有需要用到时，才使用 Resources.Load() 取到设定值，编辑场景内容时也不用再加入设定值专用的物件，有任何设定值需要修改的话，只要打开Project view内的Resources资料夹统一设定；另外，由于ScriptableObject的栏位名称及内容值的型态都已规范好，所以也大大降低人员在设置上输入错误的问题。

首先，先写一个继承自ScriptableObject的class ...

public class DataHolder : ScriptableObject {

public string[] strings;  
public int[] integers;  
public float[] floats;  
public bool[] booleans;  
public byte[] bytes;  
}

然后制作选单命令，使这个命令执行时在Resources资料夹中建立DataHolder Asset ...

[MenuItem("Custom Editor/Create Data Asset")]

static void CreateDataAsset(){

//资料 Asset 路径

string holderAssetPath = "Assets/Resources/";

if(!Directory.Exists(holderAssetPath)) Directory.CreateDirectory(holderAssetPath);

//建立实体

DataHolder holder = ScriptableObject.CreateInstance<DataHolder> ();

//使用 holder 建立名为 dataHolder.asset 的资源

AssetDatabase.CreateAsset(holder, holderAssetPath + "dataHolder.asset");

}

当点击选单Custom Editor/Create Data Asset之后，会发现在Project view会多出Resources资料夹，且里面会多一个名为dataHolder的资源，点击它的话可以在Inspector view查看到它的内容栏位就如同我们在前面撰写的DataHolder class中的栏位一样，此时我们就可以开始使用它来设定栏位值，如果我们需要多个同类型的设定，只要在点选Resources资料夹内的dataHolder时按键盘的Ctrl+D，就能复制多个来使用。

当游戏执行中，只需要用一句即可取得这个资源。

DataHolder holder = (DataHolder)Resources.Load("dataHolder" , typeof(DataHolder));

到这里应该就能明白，holder.strings代表的就是Inspector view中strings栏位的内容，以此类推，这些设定值是不是就变得很容易取得。

接下来，如果要将设定值做成外部资源来读入即可。

[MenuItem("Custom Editor/Create Data AssetBundle")]

static void CreateDataAssetBundle(){

// AssetBundle 的资料夹路径及副档名

string targetDir = "\_DataAssetBundles" + Path.DirectorySeparatorChar;

string extensionName = ".assetData";

//取得在 Project 视窗中选择的资源(包含资料夹的子目录中的资源)

Object[] SelectedAsset = Selection.GetFiltered(typeof (Object), SelectionMode.DeepAssets);

//建立存放 AssetBundle 的资料夹

if(!Directory.Exists(targetDir)) Directory.CreateDirectory(targetDir);

foreach(Object obj in SelectedAsset){

//只处理型别为 DataHolder 的资源

if(!(obj is DataHolder)) continue;

string targetFile = targetDir + obj.name + extensionName;

//建立 AssetBundle

if(BuildPipeline.BuildAssetBundle(obj, null, targetFile, BuildAssetBundleOptions.CollectDependencies)) Debug.Log(obj.name + " 建立完成");

else Debug.Log(obj.name + " 建立失败");

}

}

当选择我们在前面所建立的Resources资料夹内的dataHolder之後再点击选单Custom Editor/Create Data AssetBundle，即会在Unity专案目录下建立\_DataAssetBundles资料夹并将此资源储存为档名是 dataHolder.assetData的档案。

之后游戏中要载入，则调用以下方法。

private IEnumerator LoadDataHolder(){

// AssetBundle 档案路径

string path = string.Format("file://{0}/../\_DataAssetBundles/{1}.assetData" , Application.dataPath , "dataHolder");

// 载入 AssetBundle

WWW bundle = new WWW(path);

//等待载入完成

yield return bundle;

//取出 dataHolder 资源的内容

DataHolder holder = (DataHolder)bundle.assetBundle.mainAsset;

//卸载 AssetBundle

bundle.assetBundle.Unload(false);

}

此处的holder区域变数内容则与前面使用Resources.Load()取得的内容相同，当然，实作时可能不是使用区域变数，这就看个人如何运用了。

以上，如果是使用制作为 AssetBundle 的方式来使用的话，那么在 CreateDataAsset() 那边的就不需要是在Resources资料夹，以免都打包成AssetBundle要外部使用了，结果游戏在输出时还全都编进去了，反而变成多余的浪费。