# **Unity中如何使用AssetBundle打包**

作者：瞬间

链接：<http://gad.qq.com/article/detail/287854>

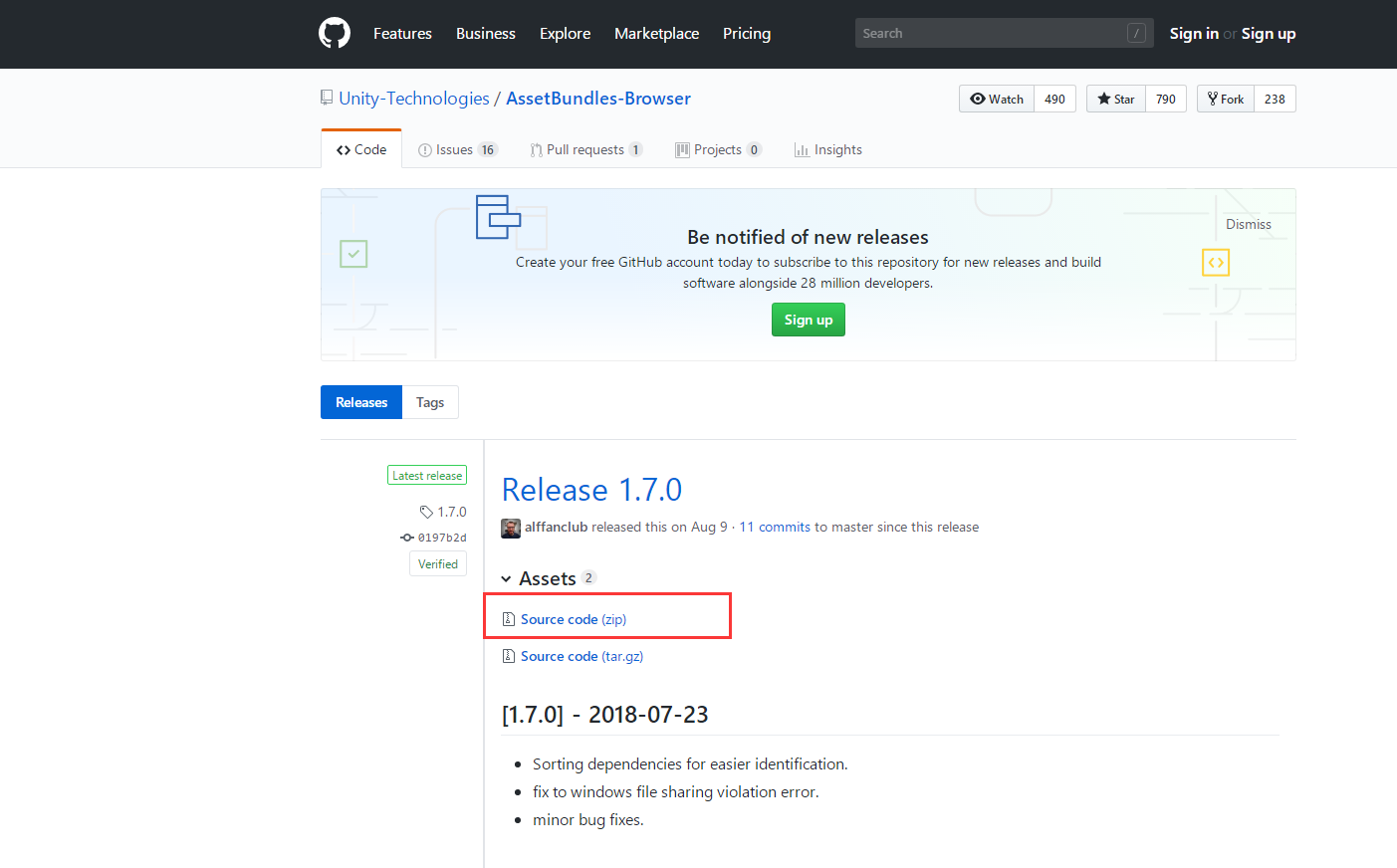
前面我们说过AssetBundle的作用<http://gad.qq.com/article/detail/287741>

这次来我告诉大家怎么使用

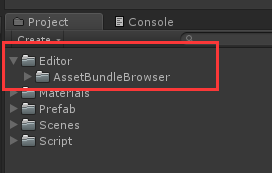
首先我们去github网站下载官方推荐插件，在此附上链接：

<https://github.com/Unity-Technologies/AssetBundles-Browser/releases>

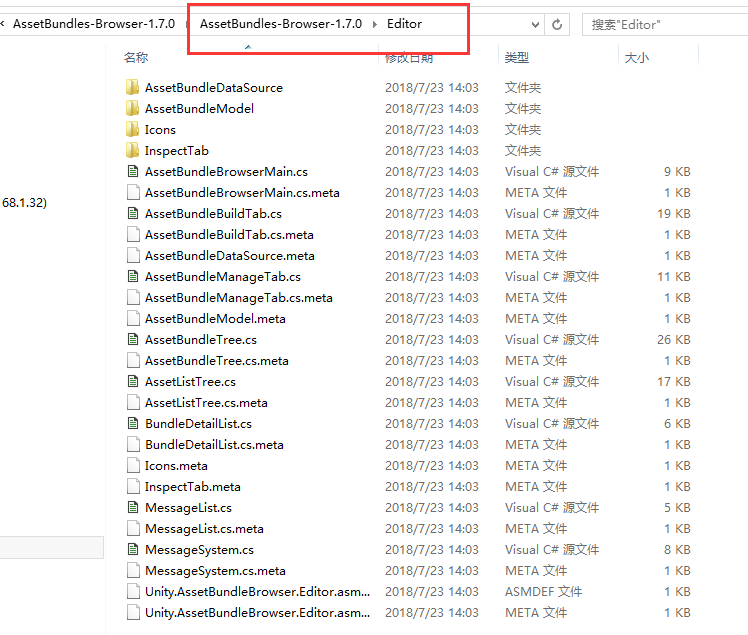
进入后点击这里



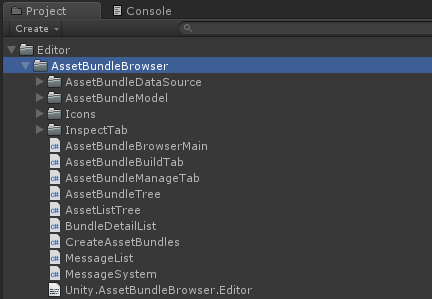
下载成功之后解压 然后在Asset文件夹下新建Editor文件夹然后在Editor文件夹下面新建AssetBundleBrowser文件夹 目录结构如图



然后将下载的压缩包解压后的Editor目录下的所有文件全选拖到AssetBundleBrowser文件夹里面

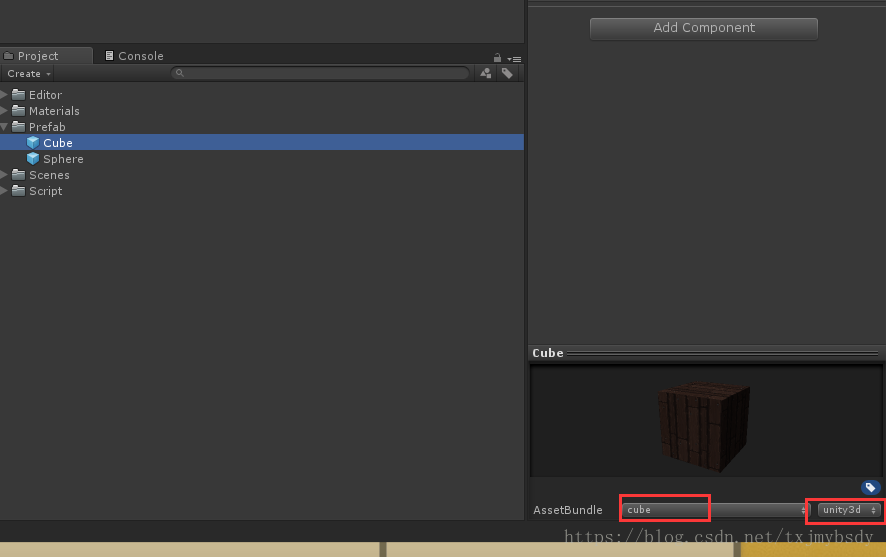


操作完成后你的unity项目里应该是这样的



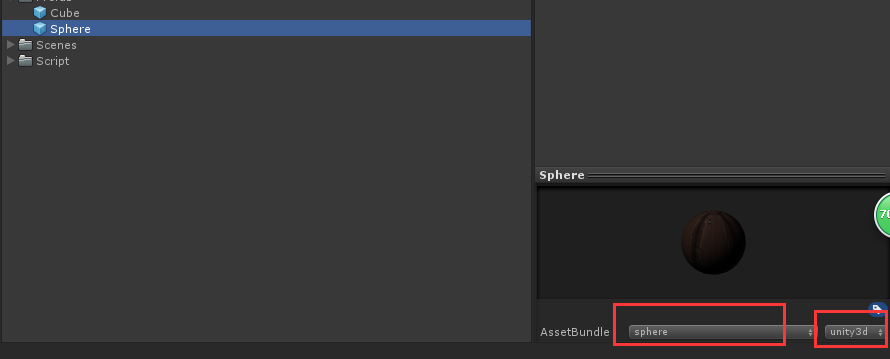
到这里第一步我们就完成了 导入了这个AssetBundle打包插件下面我们来设置一下打包的配置

我在unity里新建了两个Prefab一个是Cube一个是Sphere 在这里我们就把这两个当作项目中的资源，点击Cube会出现

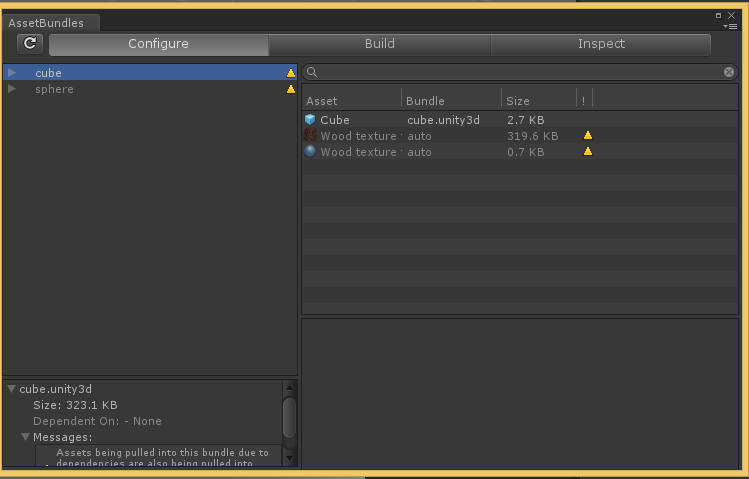


Project下的每一个文件下面都有一个AssetBundle,点击红色区域弹出New点击New会提示你输入AssetBundle的name，左边的红色区域就是这个资源被AssetBundle打包后的文件名字，右边的红色区域就是这个AssetBundle打包后的文件后缀，如上图我设置的，打包后文件为cube.unity3d，文件名和文件后缀可以随便取

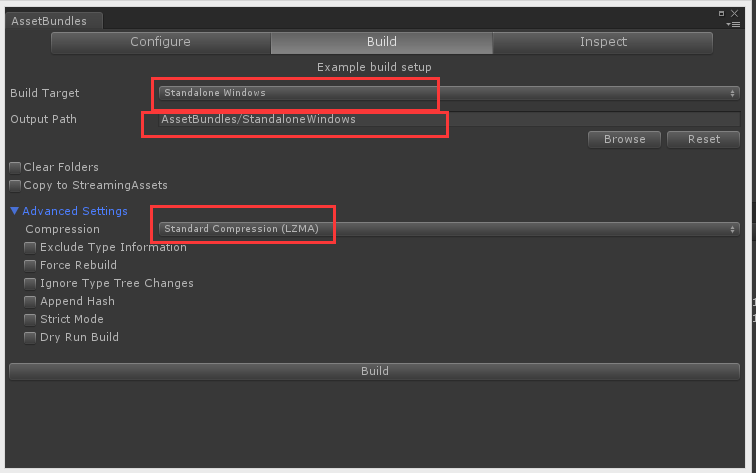
然后我们把Sphere也按照同样的方法设置一下如图



设置完成后 点击unity上面菜单的Window下拉菜单的AssetBundleBrowser



在这里我们可以看到我们刚才设置的要打包的prefab都在这里面了然后我们点击Build选项卡



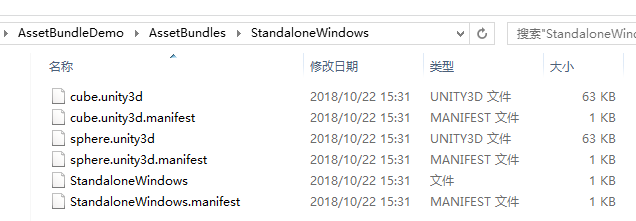
这里面有三个配置比较重要在这里要说明一下

第一个是目标平台如果你要打pc的游戏包需要选择Standalone Windows如果你要打安卓的游戏包需要选择Android如果你要打ios的游戏包需要选择IOS

第二个是打包输出的路径 就是打包后包所在的位置

第三个是打包压缩模式，第一个LZMA格式压缩这个格式压缩程度很高，可以使打出的包非常小，但使用起来需要完全解压，速度相对较慢，还有一种是LZ4的压缩格式，这种格式压缩程度比较低，打出来的包比LZMA格式打出的包大了一些(但是也比不压缩的包小了一些)，使用起来速度比较快，还有就是No Compression不压缩 这个就是保留原资源大小不进行压缩，使用起来是最快的，一般商业上使用LZMA格式和LZ4格式的比较多。

这三个选择完后点击Build就可以打包了 打包成功后应该是这样的



在这里我们打包结束了该使用代码在需要的时候加载我们的资源了

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.Networking;

public class LoadFromFileExamle : MonoBehaviour ｛

// Use this for initialization

IEnumerator Start () ｛

//UnityWebRequest取代www下载

string uri =@"file:///F:\shengcun\AssetBundleDemo\AssetBundles\cube.unity3d";//AssetBundle资源路径这里可以填写服务器地址和该资源的完整路径 每个人的电脑上的路径不一样，这里需要改成你自己的打出的包的完整路径

UnityWebRequest request = UnityWebRequestAssetBundle.GetAssetBundle(uri);

yield return request.SendWebRequest();

AssetBundle ab = DownloadHandlerAssetBundle.GetContent(request);//得到AssetBundle对象

Object[] objs = ab.LoadAllAssets();//读取AssetBundle里面的所有资源

foreach (Object o in objs)

｛

Instantiate(o);//实例化对象 这里可以根据对象的类型进行转换在游戏中使用

｝

AssetBundle manifestAB = AssetBundle.LoadFromFile("AssetBundles/AssetBundles");

AssetBundleManifest manifest = manifestAB.LoadAsset<AssetBundleManifest>("AssetBundleManifest");//读取manifest文件

string[] strs = manifest.GetAllDependencies("cube.unity3d");

foreach (string name in strs)

｛

Debug.Log(name);//打印cube.unity3d的依赖AssetBundle包 后续应该根据这个依赖包继续加载AssetBundle包 加载方法和上面相同

｝

｝

// Update is called once per frame

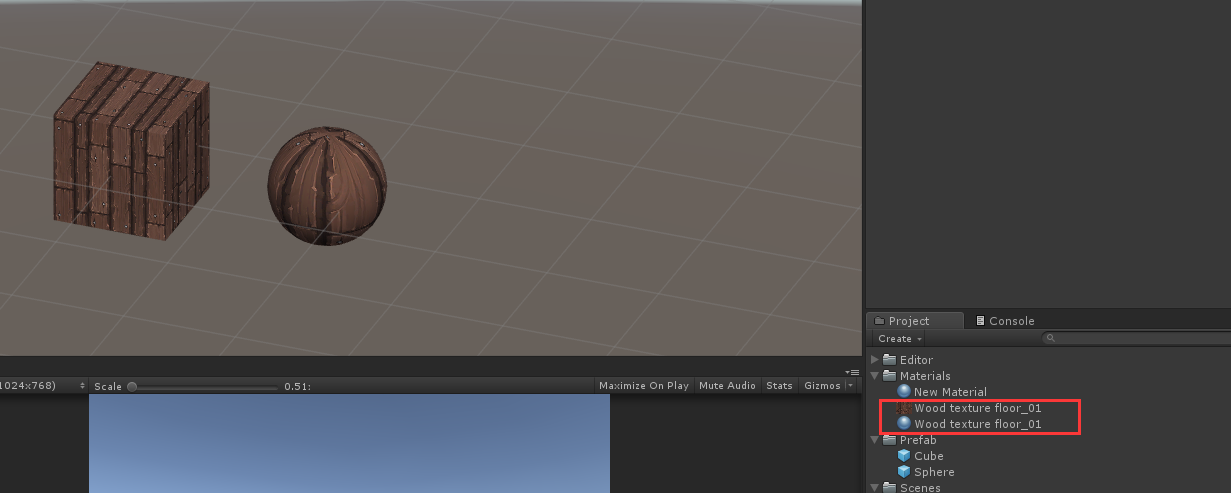
void Update () ｛

｝

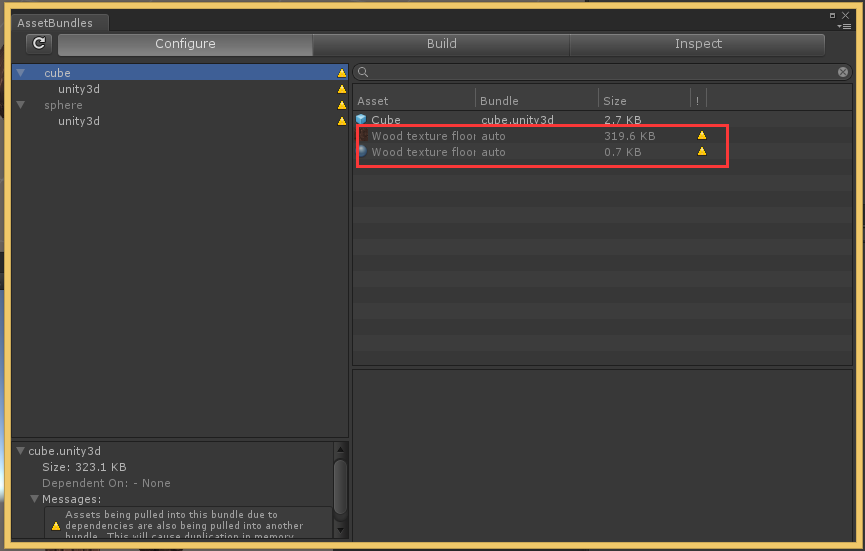
｝

上面的代码拷贝过去后不能直接使用，这时你需要在string uri=""设置你打出的AssetBundle包的完整路径，然后还有一点在代码中的Debug.Log(name);这里只是打印了AssetBundle包的依赖包名字，实际上需要继续使用上面的UnityWebRequest加载AssetBundle包的依赖包，那么什么是AssetBundle包的依赖包呢？下面我们来进行解释

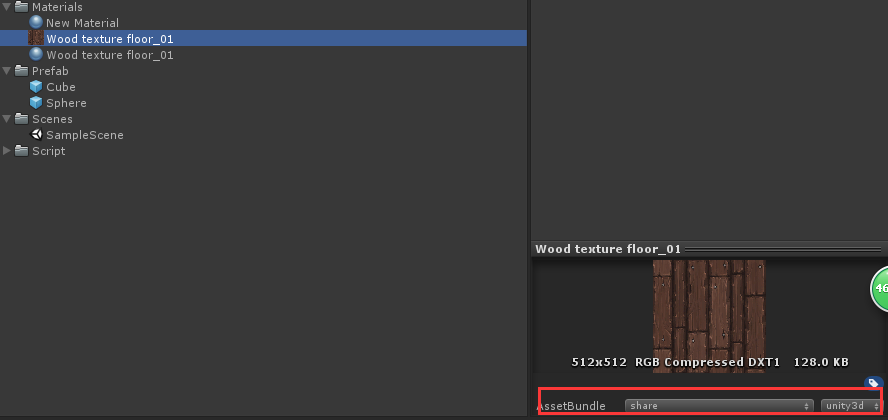
我们都知道模型上面有材质和贴图，如果我们直接将一个模型打包成AssetBundle包，unity会自动把这个模型引用的材质和贴图也打包到一起，在这里我设置一下我刚刚的一个Cube和Sphere分别给这两个物体添加一个材质和一个贴图 如图



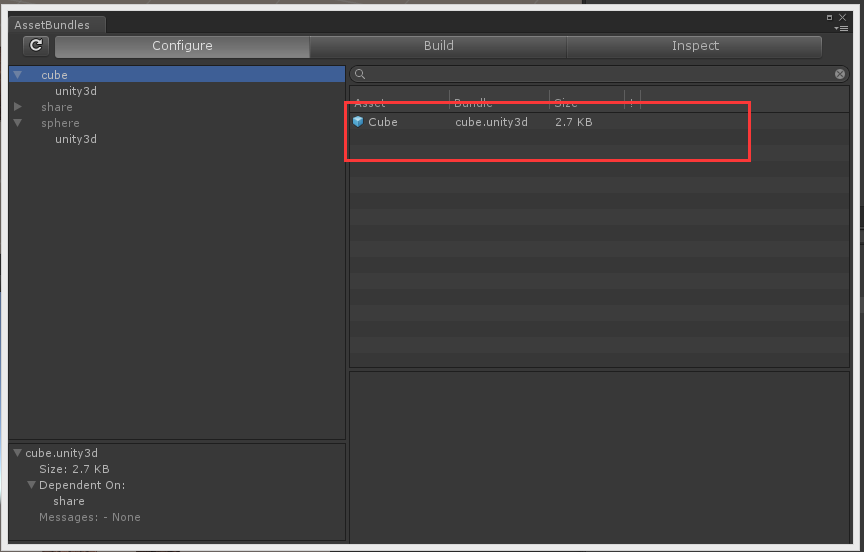
这个Cube和Sphere我给他们同样的材质和贴图，按照上面的原理，打包插件已经给出警告了



大概意思是两个AssetBundle包里面包含了相同的资源，这样会增加你的包的大小，这里是因为Cube和Sphere两个prefab都引用了同样的材质和贴图，这样资源重复反而会让我们打包后包变得更大，这绝对是我们不想要的结果，那么如何解决呢，这里需要用到AssetBundle包分组策略，在这里我推荐一种常用的分组策略，就是把Cube和Sphere两个prefab引用的模型和贴图分别打包，



我把这两个Wood texture floor\_01设置一下AssetBundle，这样的话，unity会自动检测到，Cube和Sphere两个prefab都引用了Wood texture floor\_01，那么Wood texture floor\_01打出的包就是Cube和Sphere这两个包的依赖包，然后我们重新打开插件，发现Cube里的size仅为2.7k



Cube和Sphere的Dependent On依赖包都是share包，share包里面是那两个资源文件，这样打包就达到了资源共用的效果，打包里面没有重复资源，这才是我们想要的效果。

最后，本文只是起到一些抛砖引玉的作用，更多使用细节需要在使用时慢慢研究