<http://blog.csdn.net/qq_28221881/article/details/53537658>

**Unity中的资源路径**

Application.dataPath

此属性用于返回程序的数据文件所在的文件夹的路径。例如在Editor中就是Asset了

Application.streamingAssetsPath

此属性用于返回流数据的缓存目录，返回路径为相对路径，适合设置一些外部数据文件的路径

Application.persistentDataPath

此属性用于返回一个持久化数据存储路径的路径，可以在此路径下存储一些持久化的数据文件

Application.temporaryCachePath

此属性用于返回一个临时数据的缓存目录

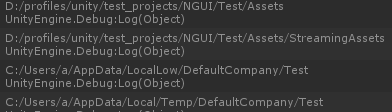
在PC上：

Debug.Log(Application.dataPath);

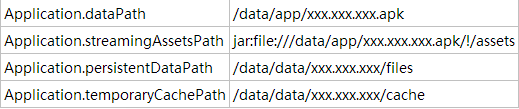
Debug.Log(Application.streamingAssetsPath);

Debug.Log(Application.persistentDataPath);

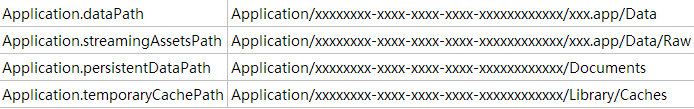
Debug.Log(Application.temporaryCachePath);



**Android平台**



**IOS平台**



从上面的表中可以看出，dataPath和streamingAssetsPath的路径位置一般是相对程序的安装目录位置，而persistentDataPath和temporaryCachePath的路径位置一般是相对所在系统的固定位置

下面介绍几个常用的资源处理类：

Resources、StreamingAssets、AssetBundle、persistentDataPath

**（1）Resources**

是作为一个unity的保留文件出现的，也就是如果你新建的文件夹名字交Resources，那么里面的内容在打包都会被无条件打到发布包中，它的特点就是：

1. 只读，即不能动态修改，所以想要动态更新的资源不要放在这里
2. 会将文件夹内的资源打包继承到.asset文件中，因此建议可以放一些prefab，因为prefab在打包时会自动过滤掉不需要的资源，有利于减小资源包的大小
3. 主线程加载
4. 资源读取方式Resources.Load()

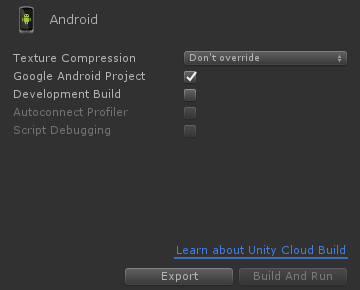
**（2）StreamingAssets**

和Resources比较像。同样作为一个只读的Unity3D的保留文件夹出现。不过两者也有很大的区别，那就是Resources文件夹中的内容在打包时会被压缩和加密，而StreamingAssets文件夹中的内容则会原封不动的打入包中，因此，StreamingAssets主要用来保存一些二进制文件，它的特点就是：

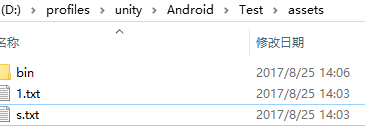
1. 只读，不可写
2. 主要用来存放二进制文件
3. 只能通过WWW类来读取

**实际上unity已经把StreamingAsset目录下的资源放在android的assets下了（已经证实）**

**在unity中的StreamingAssets中存放文件1.txt与s.txt，然后导出android文件：**



得到结果：









**（3）AssetBundle**

简而言之就是把prefab或者二进制文件封装成AssetBundle文件（也是一种二进制）。但是也有硬伤，就是在移动端无法更新脚本，它的特点是:

1. 是unity3D定义的一种二进制类型
2. 最好将prefab封装成AssetBundle，不过上面不是说了在移动端无法更新脚本吗？那从AssetBundle中拿到的prefab上挂的脚本是不是就无法运行了？也不一定，只要这个prefac上挂的是本地脚本就可以
3. 使用WWW类来下载

**（4）PersistentDataPath**

看上去只是一个路径呀，可为什么要把它从路径中单独拿出来介绍呢？因为它蛮特殊的，这个路径下是可读写的，而且在ios上就是应用程序的沙盒，但是在android上可以是程序的沙盒，也可以是sdcard。并且在android打包时，ProjectSetting页面有个选项Write Access，可以设置它的路径是沙盒还是sdcard。它的特点是：

1. 内容可读写，不过只能运行时才能写入或者读取，提前将数据存入这个路径是不可行的
2. 无内容限制，你可以从StreamingAsset中读取二进制文件或者从AssetBundle读取文件来写入PersistentDataPath中
3. 写下的文件，可以在电脑中查看，同样也可以清除

有些游戏为了让所有的资源全部使用assetbundle，会把一些初始的assetbundle放在StreamingAssets目录下，运行程序的时候在把这些assetbundle拷贝在Application.persistentDataPath目录下，如果这些assetbundle有更新的话，那么下载到新的assetbundle在把Application.persistentDataPath目录下原有的覆盖掉。

因为Application.persistentDataPath目录是应用程序的沙盒目录，所以打包之前是没有这个目录的，直到应用程序在手机上安装完毕才有这个目录。

**（1）采用Resources读取文件**

将test.xml放入Resources文件夹中

public class Test : MonoBehaviour

{

string result;

void Start()

{

LoadXML("test"); **// 注意：这里不要添加后缀名**

}

void LoadXML(string path)

{

result = Resources.Load(path).ToString();

Debug.Log("1 reslut = " + result);

XmlDocument doc = new XmlDocument();

doc.LoadXml(result);

Debug.Log("2 result = " + result);

// 实际上1和2得到的result内容是一样的

}

void OnGUI()

{

GUIStyle titleStyle = new GUIStyle();

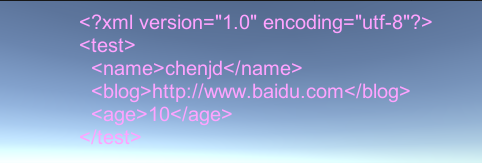
titleStyle.fontSize = 20;

titleStyle.normal.textColor = new Color(46f / 25f, 163f / 255f, 255f / 255f, 255f / 255f);

GUI.Label(new Rect(200, 10, 500, 200), result, titleStyle);

}

}



**（2）采用StreamingAssets读取文件**

创建StreamingAssets文件，并将文件放在该文件夹下

public class Test : MonoBehaviour

{

string result;

void Start()

{

StartCoroutine(LoadXML());

}

string filePath = "file://"; // **这里很重要**

IEnumerator LoadXML()

{

string path = filePath + Application.streamingAssetsPath + "/test.xml";

WWW www = new WWW(path);

Debug.Log("path = " + path);

yield return www;

result = www.text;

Debug.Log("result = " + result);

}

void OnGUI()

{

GUIStyle titleStyle = new GUIStyle();

titleStyle.fontSize = 20;

titleStyle.normal.textColor = new Color(46f / 25f, 163f / 255f, 255f / 255f, 255f / 255f);

GUI.Label(new Rect(200, 10, 500, 200), result, titleStyle);

}

}

