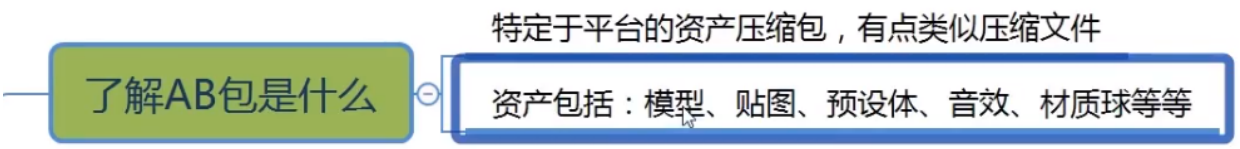
1

ＡＢ包是什么？



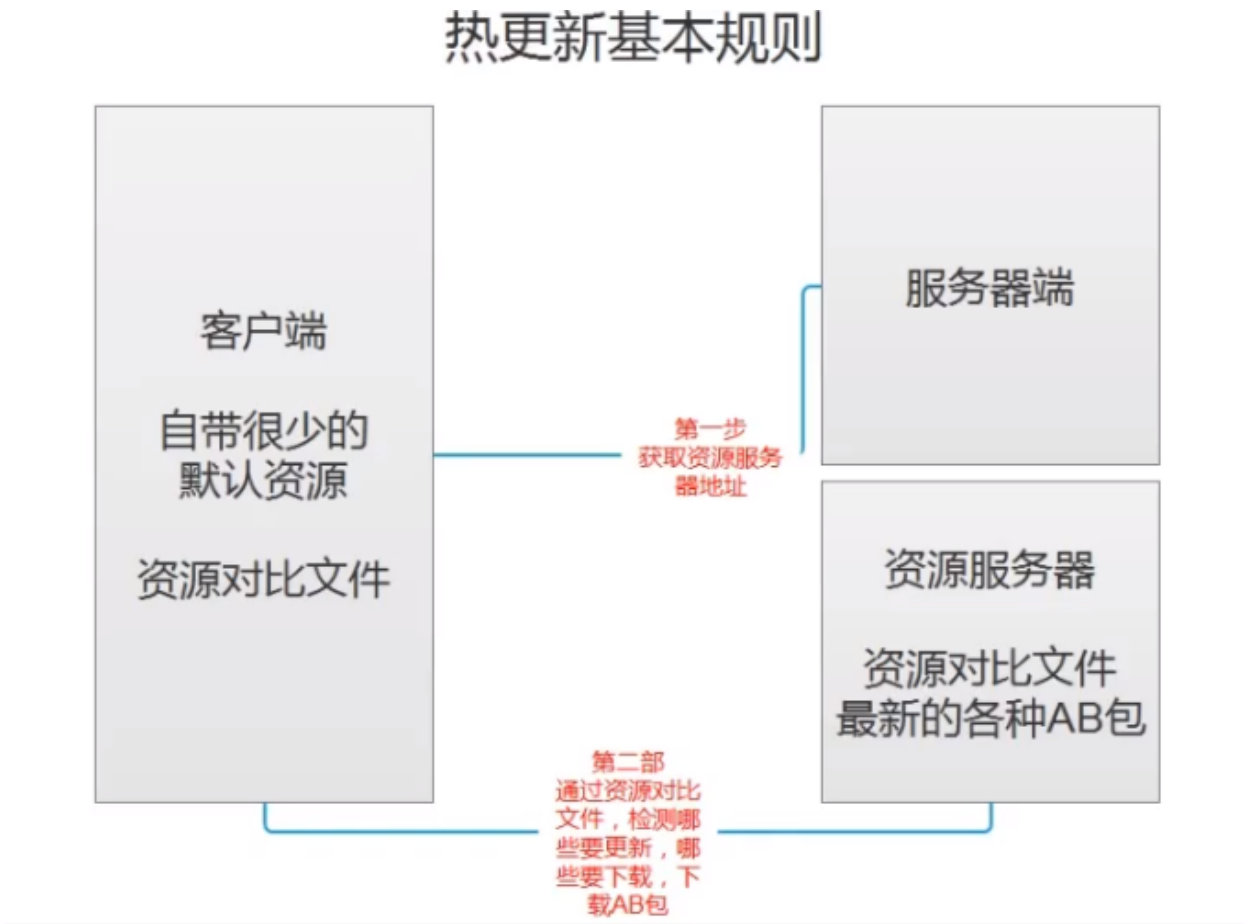
2

ＡＢ包有什么用？

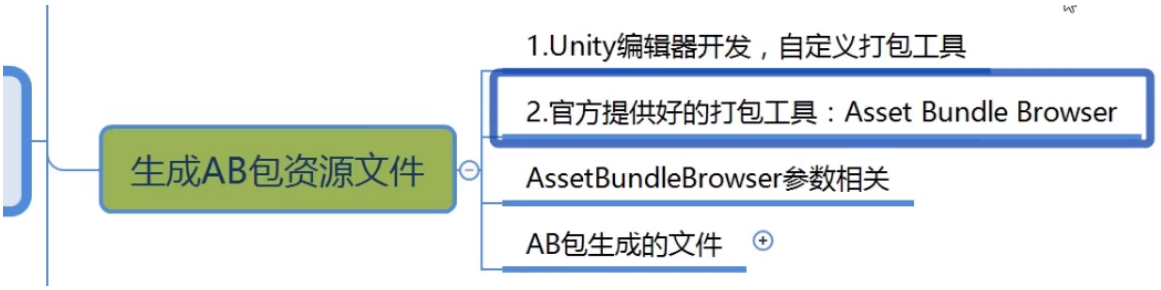
１.相对于Ｒｅｓｏｕｃｅ下的资源ＡＢ包更好的管理资源

２.压缩资源，减少初始包的大小（先下载较小的压缩包，之后通过服务器下载额外内容）

３.能够热更新

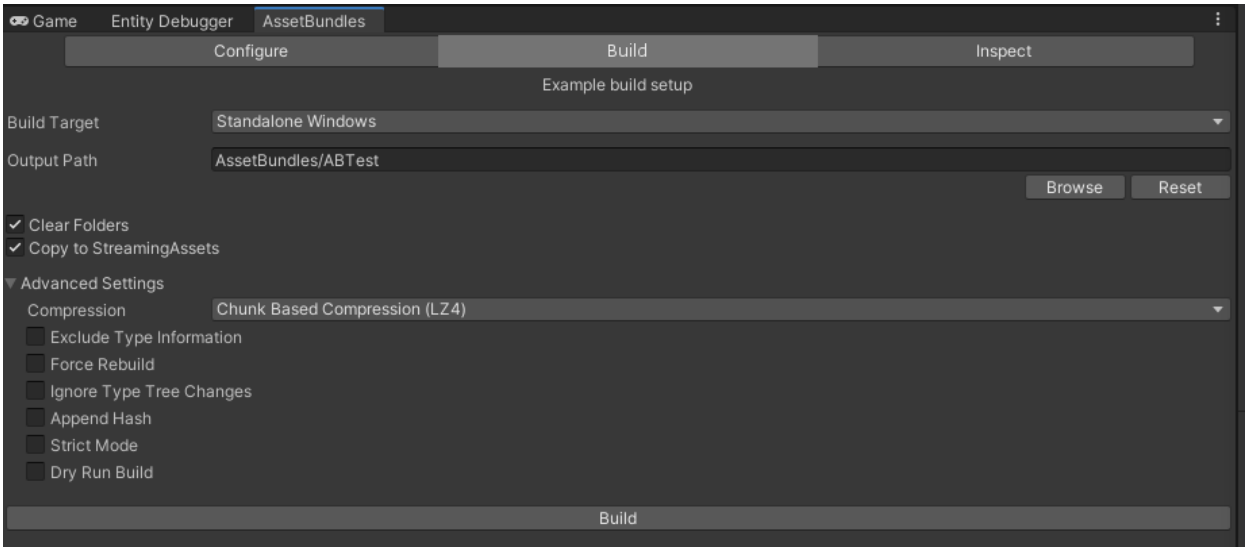


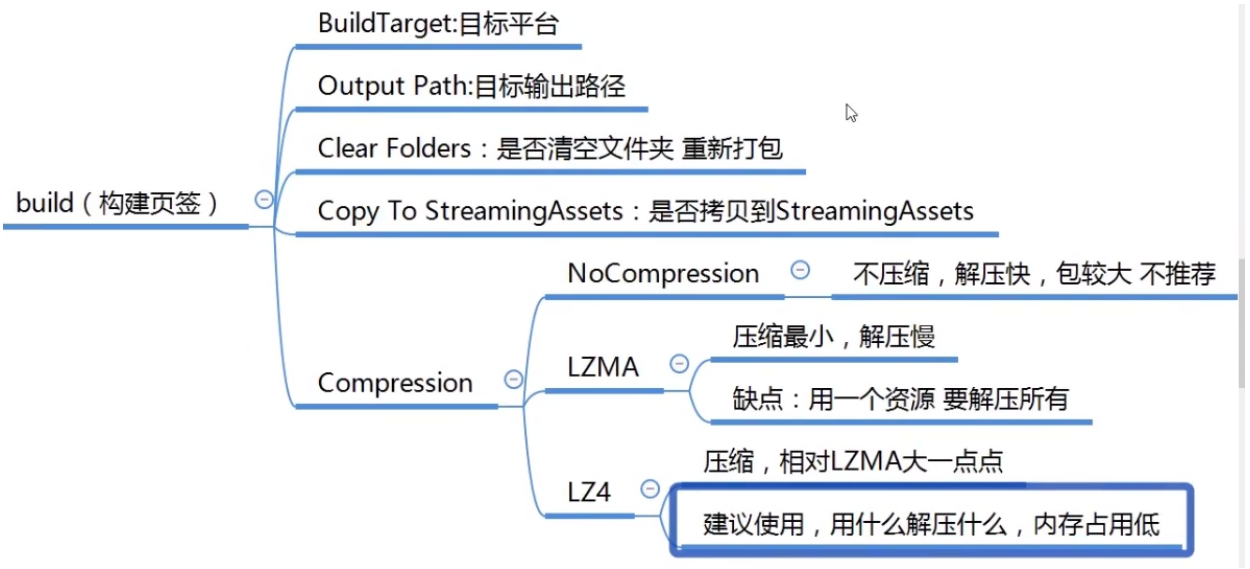
3



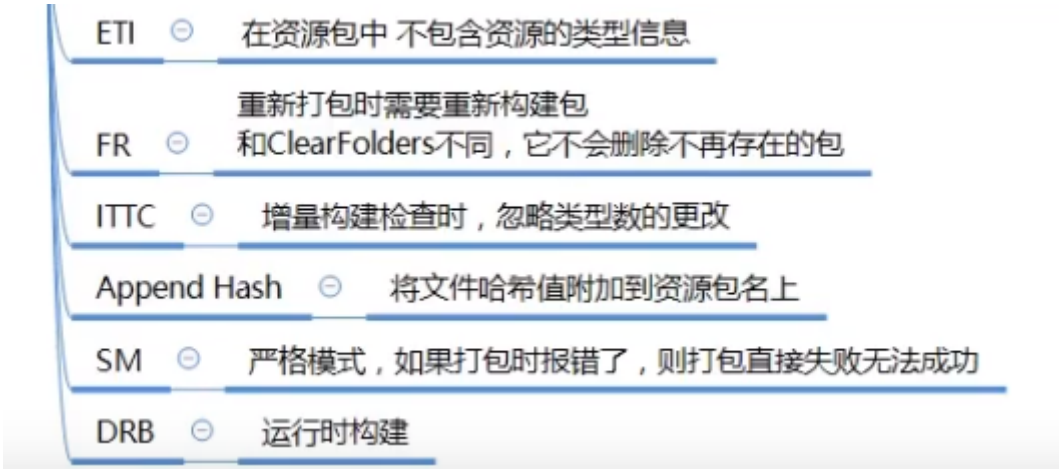
4

Asset Bundle Browser


Game
Entity Debugger
AssetBundles
Configure
Build
Inspect
Example build setup
Build Target
Standalone Windows
Output Path
AssetBundles/ABTest
Browse
Reset
Clear Folders
Copy to StreamingAssets
Advanced Settings
Compression
Chunk Based Compression (LZ4)
Exclude Type Information
Force Rebuild
Ignore Type Tree Changes
Append Hash
Strict Mode
Dry Run Build
Build


build（构建页签）
BuildTarget:目标平台
Output Path:目标输出路径
Clear Folders：是否清空文件夹 重新打包
Copy To StreamingAssets：是否拷贝到StreamingAssets
Compression
NoCompression
不压缩，解压快，包较大 不推荐
LZMA
压缩最小，解压慢
缺点：用一个资源 要解压所有
LZ4
压缩，相对LZMA大一点点
建议使用，用什么解压什么，内存占用低

不太重要的选项


ETI
在资源包中 不包含资源的类型信息
FR
重新打包时需要重新构建包
和ClearFolders不同，它不会删除不再存在的包
ITTC
增量构建检查时，忽略类型数的更改
Append Hash
将文件哈希值附加到资源包名上
SM
严格模式，如果打包时报错了，则打包直接失败无法成功
DRB
运行时构建

同步异步加载 a b 包资源

```
void Start()
{
    //第一步 加载AB包
   // AssetBundle ab=AssetBundle.LoadFromFile(Application.streamingAssetsPath +
"/cube1");

    //第二部 加载AB包中的资源
    //GameObject obj=ab.LoadAsset<GameObject>("cube1");
   // GameObject obj = ab.LoadAsset(" EcsCube",typeof(GameObject)) as GameObject;

    //Instantiate(obj);

    StartCoroutine(LoadABres("cube1", "EcsCube"));

    //卸载 所有加载的ab包  参数为true 会把通过ab包加载的资源也卸载掉
    AssetBundle.UnloadAllAssetBundles(false);
}

IEnumerator LoadABres(string abName, string reName)
{
    //第一步加载资源包
    AssetBundleCreateRequest abcr =
AssetBundle.LoadFromFileAsync(Application.streamingAssetsPath

                                                              + "/" + abName);
    yield return abcr;
    //第二部加载资源

    AssetBundleRequest abq = abcr.assetBundle.LoadAssetAsync(reName,
typeof(GameObject));
    Instantiate(abq.asset as GameObject);
    yield return abq;

}
```

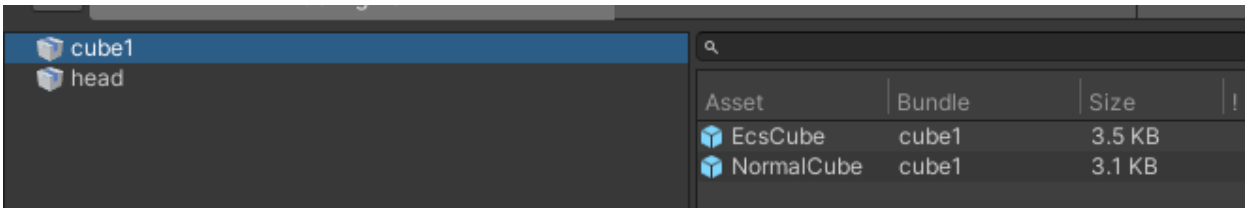
ＡＢ包依赖

关于ＡＢ包的依赖－－－－一个资源身上用到别的ＡＢ包中的资源时，如果只加载了自己的ＡＢ包

通过它创建对象　会出现资源丢失的情况

这个时候需要把依赖包一起加载，才能正常加载

一个资源可能依赖很多其他资源，通过获取主包的依赖信息进行加载。




```
//第一步　加载ab包
AssetBundle ab=AssetBundle.LoadFromFile(Application.streamingAssetsPath + "/"+"cube");
//依赖包的关键知识点　利用主包　获取依赖信息
AssetBundle abMain = AssetBundle.LoadFromFile(Application.streamingAssetsPath +
"/"+"ABTest");
//加载主包中的固定文件
AssetBundleManifest abMainfest = abMain.LoadAsset<AssetBundleManifest>
("AssetBundleManifest");
//从固定文件中获取依赖信息
string[] strs = abMainfest.GetAllDependencies("cube");
//Instantiate(ab.LoadAsset("EcsCube",typeof(GameObject)) as GameObject);
//得到了　依赖包的名字
for (int i = 0; i < strs.Length; i++)
{
    Debug.Log(strs[i]);
    AssetBundle.LoadFromFile(Application.streamingAssetsPath + "/"+strs[i]);
}
GameObject obj = ab.LoadAsset("EcsCube",typeof(GameObject)) as GameObject;
```

```
Instantiate(obj);
```

7

ＡＢ包资源加载管理器

（１）同步加载

```
public class ABManager : MonoBehaviour
{
    public static ABManager Instace;

    private void Awake()
    {
        Instace = this;
    }
    //ab包管理器目的是
    //让外部更方便进行资源加载

    //ab包不能重复加载  用字典存储已加载的ab包
    private Dictionary<string, AssetBundle> abDic = new Dictionary<string,
AssetBundle>();

    //主包
    private AssetBundle mainAB = null;

    //依赖包
    private AssetBundleManifest mainfest = null;

    //ab包存放路径方便修改
    private string PathUrl
    {
        get { return Application.streamingAssetsPath + "/"; }
    }

    private string MainABName
    {
        get
```

```
    {
        return "ABTest";
    }
}

/// <summary>
/// 加载ab包
/// </summary>
/// <param name="abName"></param>
private void LoadAB(string abName)
{
    //加载AB包
    if (mainAB == null)
    {
        mainAB = AssetBundle.LoadFromFile(PathUrl + MainABName);
        mainfest = mainAB.LoadAsset<AssetBundleManifest>("AssetBundleManifest");
    }

    AssetBundle ab = null;
    //获取依赖包的相关信息
    string[] strs = mainfest.GetAllDependencies(abName);
    for (int i = 0; i < strs.Length; i++)
    {
        if (!abDic.ContainsKey(strs[i]))
        {
            ab = AssetBundle.LoadFromFile(PathUrl + strs[i]);
            abDic.Add(strs[i], ab);
        }
    }

    //加载主包来获取资源
    if (!abDic.ContainsKey(abName))
    {
        ab = AssetBundle.LoadFromFile(PathUrl + abName);
        abDic.Add(abName, ab);
    }
}
```

```
/// <summary>
/// 同步加载
/// </summary>
/// <param name="abName"></param>
/// <param name="resName"></param>
/// <returns></returns>
public Object LoadRes(string abName, string resName)
{
   LoadAB(abName);

    //加载资源
    //为了方便 判断是不是gameobject 则直接返回gameobject
    Object obj= abDic[abName].LoadAsset(resName);
    if (obj is GameObject)
        return Instantiate(obj);
    else
        return obj;
}

/// <summary>
/// 同步加载  根据type指定类型  XLua会用到
/// </summary>
/// <param name="abName"></param>
/// <param name="resName"></param>
/// <param name="type"></param>
/// <returns></returns>
public Object LoadRes(string abName, string resName,System.Type type)
{

    LoadAB(abName);
    //加载资源
    //为了方便 判断是不是gameobject 则直接返回gameobject
    Object obj= abDic[abName].LoadAsset(resName,type);
    if (obj is GameObject)
        return Instantiate(obj);
```

```
    else
        return obj;
}

/// <summary>
/// 根据泛型指定类型 加载资源
/// </summary>
/// <param name="abName"></param>
/// <param name="resName"></param>
/// <typeparam name="T"></typeparam>
/// <returns></returns>
public T LoadRes<T>(string abName, string resName) where T:Object
{

    LoadAB(abName);
    //加载资源
    //为了方便 判断是不是gameobject 则直接返回gameobject
    T obj= abDic[abName].LoadAsset<T>(resName);
    if (obj is GameObject)
        return Instantiate(obj);
    else
        return obj;
}


//异步加载

//单个包卸载
public void UnLoad(string abName)
{
    if (abDic.ContainsKey(abName))
    {
        abDic[abName].Unload(false);
        abDic.Remove(abName);
    }
}
```

```
    //所有包的卸载
    public void ClearAB()
    {
        AssetBundle.UnloadAllAssetBundles(false);
        abDic.Clear();
        mainAB = null;
        mainfest = null;
    }

}
```

（2）异步加载

```
        public void LoadResAsync(string abName, string resName, System.Type
                                 type,UnityAction<object> callBack)
{
    StartCoroutine(ReallyLoadResAsyns(abName, resName,type, callBack));
}

private IEnumerator ReallyLoadResAsyns<T>(string abName, string resName,
UnityAction<object> callBack)
{
    LoadAB(abName);
    //加载资源
    //为了方便 判断是不是gameobject 则直接返回gameobject
    AssetBundleRequest abr= abDic[abName].LoadAssetAsync(resName);

    yield return abr;
    if (abr is GameObject)
        callBack(Instantiate(abr.asset));
    else
        callBack(abr.asset);
}
```