# 打卡 新建脚本就有了默认程序集

[202207102242](GIF/202207102242.gif)

# 打卡 根目录新建程序集就抢了默认程序集的地位

[202207102244](GIF/202207102244.gif)

# 打卡 子目录（不在根目录）新建程序集，默认程序集又有地位了

[202207102246](GIF/202207102246.gif)

# 打卡 程序集在VS中的显示

[202207102256](GIF/202207102256.gif)

# 打卡 程序集面板

[202207111039](GIF/202207111039.PNG)

# 加载方式------------------

# 01 拖

# 02 Resources

上限制2G

AssetBundle加载更快

# 打卡 03 AssetBundle 04AssetDataBase

[202207111217](GIF/202207111217.gif)

# 打卡 Xml2Class

[202207111341](GIF/202207111341.gif)

# 打卡 Class2Xml

[202207111352](GIF/202207111352.gif)

# 打卡 Class2Bin

[202207111432](GIF/202207111432.gif)

# 打卡 读取Asset(里面是ScriptableObject)

[202207111505](GIF/202207111505.gif)

# AB-----------------------------

# 打卡 配置一个ABConfig

[202207111902](GIF/202207111902.gif)

# 打卡 打包Prefab

太快，闪过去了

[202207111945](GIF/202207111945.gif)

# 打卡 标记 命名需要AB的文件

[202207121028](GIF/2022071121028.gif)

# 打卡 清理标记 清理命名的AB文件

[2022071221045](GIF/202207121045.gif)

# 打卡 打包标记的文件

[2022071121140](GIF/202207121140.gif)

# bug The name `AssetDatabase' does not exist in the current context

之前也没报错

暂时注释掉

# 打卡 删包

[202207121259](GIF/202207121259.gif)

# 打卡 生成xml（可视化），Bin（读取）

[202207121609](GIF/202207121609.gif)

# 打卡 只写出有效路径

## 写之前

[202207121613](GIF/AssetBundleConfig.xml)

## 写之后

[202207121631](GIF/AssetBundleConfig 之后.xml)

[202207121633](GIF/202207121633.gif)

# bug Bin文件放在StreamingAssets报Null

先放根目录

做成AB，再加载

# bug SerializationException: Could not find type 'System.Collections.Generic.List

我把ABConfig放Editor

# 注意 ABConfig

Demo01-04有一个

Editor和Demo05有一个

不一样的

我用程序集

# bug 加载的预制体丢失贴图 shader hiden internalError

代码控制正常

[202207130128](GIF/202207130128.txt)

# bug 生成xml为空

生成代码顺序

# 打卡 游戏中加载AB中的prefab

[202207130130](GIF/202207130130.gif)

# 注意 Mono中有AssetDataBase，打包时会报错，打包前注释掉

# 打卡 将生成的配置Config，做成ab包来加载

[202207130154](GIF/202207130154.gif)

# Mgr----------------------------

# 打卡 通过Mgr加载AB

[202207131605](GIF/202207131605.gif)

# 打卡 字典的get set

[202207141035](GIF/202207141035.gif)

# 打卡 演示同步资源的加载和卸载

[202207141652](GIF/202207141652.gif)

# 打卡 演示异步资源的加载和卸载

[202007150130](GIF/202207150130.gif)

# 打卡 是否预加载的速度差距

因为刚停止，马上再运行。与隔了很久才运行是有差别的。

所以预加载与否、以上两种情况，一共演示四种情况

[202207151354](GIF/202207151354.gif)

# 了解 写代码习惯

经常开头renturn掉不对的条件

[202207152115](GIF/202207152115.PNG)

# 了解 数据结构

crc，与path互转，索引AB包

GUID，唯一标识符，GetInstaniateID()；AB也有，所有在Unity中的资源都有，但ABItem不存储该字段

ref，引用计数，ABItem,ResItem有，因为ABItem被ResItem引用，ResItem被ResObj引用，被引用就计数

JmpClr，跳转场景是否清除缓存，ResItem和ResObject在内存中，而ABItem是数据来源

ABItem,ResItem活跃在AssetBundleMgr,ResourceMgr，ResObj活跃在ObjectMgr

[202207171424](GIF/202207171424.PNG)

# bug 修改脚本名导致bytes不可用，解析不了

在打包2后，修改了涉及bytes的脚本名字。导致bytes废弃。

而要再次拿到可用bytes

01 恢复打包1的代码，包括隔离生成原生bytes的代码

02 修改bytes中被涉及到的部分

03 改回原来的代码名



# 了解

运行后退出，隔一会，再运行，不然上次的内存来不及清理

# bug 清存后不能深度清存

[202207181501](GIF/202207181501.gif)

RefCnt函数正负放了，导致RefCnt>0，请不了

# 打卡 同步加载的清存（缓存）、深度清存（不缓存）

实例的任务Texture2D变化大，所以重点看Texture2D

[202207181632](GIF/202207181632.gif)

# bug 从AB包加载时候材质丢失

# 打卡 从AB包同步加载的清存（缓存）、深度清存（不缓存）

有材质丢失（其实是Shader（benghuai））丢失情况，暂时用函数强制修改

[202207181803](GIF/202207181803.gif)

# 异步实例-------------------------

# 了解 代码命名

从不同地方得到数据的动作命名

[202207182059](GIF/202207182059.PNG)

# bug 异步实例，深度清存，内存泄漏

AB依赖没错，对比以前

异步的resObj.m\_ResItem.m\_ABName 为空，m\_AB为null

[202207191300](GIF/202207191300.gif)

到源头异步加载完成的回调函数中发现resObj.m\_ResItem中的m\_ABName，m\_AB为null

[202207191303](GIF/202207191303.gif)

找上游cbLst没错，在找上游，ResourceMgr.InitCortinus，看到控制资源来源的m\_loadFromAB（true，从AB包加载，False从Editor也就是DataBase加载）

改变#UNITY\_EDITOR的位置（放if里面），m\_loadFromAB改成false，生效

[202207191333](GIF/202207191303.gif)

m\_loadFromAB改成true，不生效，那就是再后面的代码问题了

发现缓存函数 CacheResItem，存在代码错误（new 了）。

[202207191431](GIF/202207191431.PNG)

此时【异步的resObj.m\_ResItem.m\_ABName 为空，m\_AB为null】解决了

[202207191435](GIF/202207191335.gif)

# bug 异步加载时RefCnt=2导致后面计数清存失败

[202207191515](GIF/202207191515.gif)

原因

[202207191658](GIF/202207191658.gif)

进一步 LoadCfdg后查看字典，方向初始时就为1

所以修改初始数量，测试无效

发现异步加载，调用了同步加载方法LoadResObj(里面有Cache方法导致计数加一)，

# 打卡 异步实例资源

[202207191805](GIF/202207191805.gif)

# 离线数据------------------

改得太多，还原时，Pool使用难受

# 离线数据编辑器，生成离线数据（发生git Err，remote弄丢了一个异步commit，但内容没丢，看前面的打卡）

[202207201324](GIF/202207201324.gif)

# UIOfflineData的AllCreate和Reset

[202207201712](GIF/202207201712.gif)

# ParticleOfflineData的泛型AllCreate（UI不行，要检测layer）

[202207201712](GIF/202207201712.gif)

# UI管理---------------------

# UI框架的基本演示

[202207212345](GIF/202207212345.gif)

# 了解 AB包的结构（在了解加载路径时）



# bug unity运行游戏后卡死，只能重启

while改成if

初学者不懂协程，调试很惨

[202207241644](GIF/202207241644.PNG)

# 打卡 Mgr加载场景

[202207241648](GIF/202207241648.gif)

# 打卡 场景切换后资源（可实例）清存[AB包加载]

[202207251100](GIF/202207251100.gif)

# 打卡 场景切换后预加载

Ab编辑器两种加载

[202207251131](GIF/202207251131.gif)

# bug AB挂脚本不能打包Script assets cannot be included in AssetBundles:

目前的办法，强行打包，因为使用不影响（记得增减AB资源后，重新打包）

有的可以后面Add

但是有些不能Add，因为数据需要Editor驱动

<https://docs.unity3d.com/550/Documentation/Manual/scriptsinassetbundles.html>

// Load the TextAsset object

TextAsset txt = bundle.Load("myBinaryAsText", typeof(TextAsset)) as TextAsset;

// Load the assembly and get a type (class) from it

var assembly = System.Reflection.Assembly.Load(txt.bytes);

var type = assembly.GetType("MyClassDerivedFromMonoBehaviour");

// Instantiate a GameObject and add a component with the loaded class

GameObject go = new GameObject();

go.AddComponent(type);

# bug The name `AssetDatabase' does not exist in the current context

因为不熟，我之前的做法是注释掉，现在都加上

#if UNITY\_EDITOR

#endif

# 打卡 场景切换后预加载资源（不可实例）清存

[202207251352](GIF/202207251352.gif)

# 打卡 场景切换后资源（不可实例）的清存（LoadFromAB）

[202207251532](GIF/202207251532.gif)

# 打卡 场景切换后资源（不可实例）的清存（LoadFromAB=false）

[202207251535](GIF/202207251535.gif)

# 打卡 场景切换后使用预加载的可实例资源，然后回收

[202207251545](GIF/202207251545.gif)

# 打卡 场景切换后使用预加载的可实例资源，然后销毁

[202207251613](GIF/202207251613.gif)

# 了解 调试的情况

变量：场景切换

变量：数据来源是AB还是编辑器

变量：资源能不能实例

变量：是不是预加载

结果：看内存表，看回收节点

# 了解 cfg的变化

之前卸载SO里面，现在直接卸载代码里， SetABName(m\_outputABName, m\_inputBytes);//cfg，实现分离



# 打卡 fix 能新增AB包并且可以实例

[202207262123](GIF/202207262123.gif)

# 了解 A是AB包，里面是B的路径



# 打卡 两种方式来异步加载资源（不可实例）

[202207271412](GIF/202207271412.gif)

# 打卡 打包exe和导出AB

(没安装安卓SDK,所以不演示apk)

/../上一级的写法导不出，所以用自己写的TrimName来解决

同样是忽视OfflineData脚本的不能添加进AB包的Err

打包太大了，所以删掉了

可以看到StreamingAsset少了个/，后面补上了

还有meta不应该打包，后面也会改过来

[202207280039](GIF/202207280039.gif)

# 配置表---------------------

# 数据配置



# 关系图

[202207281430](GIF/202207281430.PNG)

# 打卡 类转Xml

[202207291152](GIF/202207291152.gif)

# 打卡 Xml转Bin

[202207291925](GIF/202207291925.gif)

# 打卡 读取外部xml

[202207292337](GIF/202207292337.gif)

# 打卡 epplus写入excel

Plugins/Editor引入 epplus.dll

[202207302112](GIF/202207302112.gif)

# 打卡 反射读取属性值

[202207302159](GIF/202207302159.gif)

# 打卡 反射读取列表

[202207302237](GIF/202207302237.gif)

# 打卡 反射读取列表（存的是类）

[202207312054](GIF/202207312054.gif)

# 打卡 根据已有数据反射一个类

[202207312131](GIF/202207312131.gif)

# 打卡 根据已有数据反射一个类（浮点，枚举）

[202207312156](GIF/202207312156.gif)

# 打卡 根据已有数据反射一个类（列表）

[202208011109](GIF/202208011109.gif)

# 打卡 根据已有数据反射一个类（列表存类的实例）

[202208011349](GIF/202208011349.gif)

# 打卡 xml转Excel

[202208041452](GIF/202208041452.gif)

# 打卡 CfgMgr读取两个Bin

[202208042238](GIF/202208042238.gif)

# 打卡 框选xml转Excel

[202208042327](GIF/202208042327.gif)

# 打卡 xml转Excel（补充类表不拆分）

[202208051440](GIF/202208051440.gif)

# 打卡 Excel转Xml（仅MonsetrData）

[202208060029](GIF/202208060029.gif)

# 打卡 Excel转Xml（默认值）

[202208061159](GIF/202208061159.gif)

# 打卡 Excel转Xml（处理非类列表）

[202208061544](GIF/202208061544.gif)

# 打卡 Excel转Xml（不拆分的类列表）

[202208062111](GIF/202208062111.gif)

# 打卡 Excel转Xml（拆分的类列表）

[202208070342](GIF/202208070342.gif)

# 了解 Reg

Reg（数据结构，Xml）,是两种数据格式， Xml与Excel交流的中介，必须存在

# 打卡 Excel转Xml （All）

[202208070540](GIF/202208070540.gif)

# 打卡 架构图

[202208070552](GIF/202208070552.png)

# 了解 const 与 static

const更严格，不能包含变量

static则可以

# 打卡 Excel转Bin （All）

[202208070623](GIF/202208070623.gif)

# 了解 打包位置

位置从内部AstreamingAssets到外部AssetBundle，所以AssetBundleMgr加载就没用了

所以其它所有使用资源的地方都要进行处理

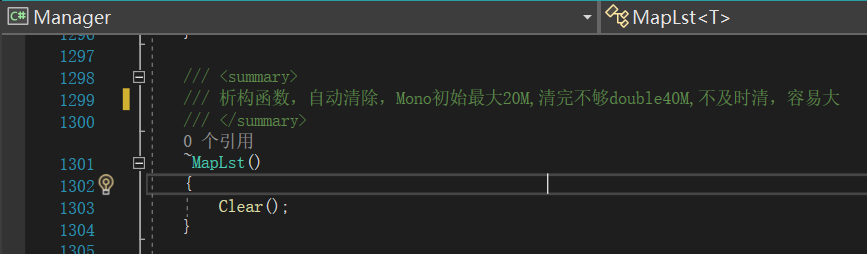
# 打卡 加载生成的Bin

[202208070736](GIF/202208070736.gif)

# 了解 AB结构

AssetBundleEditor中生成assetbundleconfig.bytes需要被打包，所以步骤3的生成xml,bin，杂在标记时进行

# 了解析构函数



# 了解 编辑旗下卸载有不如意的

比如清掉引用都还在

所以对编辑器下的要特意清存，AB不会

但是以下的写法会更省性能

[202208080238](GIF/202208080238.PNG)

# 了解 File的使用范围

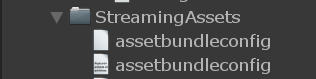
[202208080343](GIF/202208080343.PNG)

# 打卡 用法

**01 编辑器下读取AB**

打一次，生成Assets/RealFrame/GameData/Data/ABData/AssetBundleConfig.bytes，后删除xml和ab包，再打一次

比原来多生成一个如下图，编辑旗下通过它可以读取Assets/RealFrame/GameData/Data/ABData/AssetBundleConfig.bytes



**02 真正使用**

直接打一次，就行了