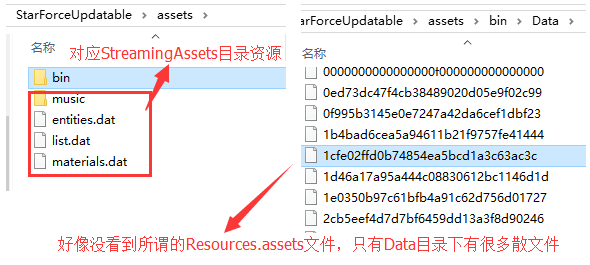
20180119技术部内部分享：如何做好一个Unity资源更新器和管理器—Efficient & Robust

**Unity目录及资源管理：**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unity特殊目录** | **介绍** | **资源加载** | **资源卸载** |
| Assets | 为Unity编辑器下的资源文件夹，Unity项目编辑时的所有资源都将置入此文件夹内。Assets下的资源除特殊文件夹内，或者在会打入包内的场景中引用的资源，其余资源不会被打入包中。 | AssetDatabase.LoadAssetAtPath("Assets/x.txt"); |  |
| Resources | All assets found in the Resources folders and their dependencies are stored in a file called *resources.assets*.  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。 | Resources.LoadAsync("fileName"); | Resources.UnloadAsset(Object);//卸载对应Object  //卸载所有没有被引用以及实例化的Object  Resources.UnloadUnusedAssets(); |
| StreamingAssets | StreamingAssets文件夹为流媒体文件夹，此文件夹内的资源将不会经过压缩与加密，原封不动的打包进游戏包内。因此StreamingAssets主要用来存放一些二进制文件。 | StreamingAssets文件夹下的文件在游戏中只能通过IO Stream或者WWW的方式读取（AssetBundle除外）。  AssetBundle.LoadFromFileAsync()  AssetBundle.LoadFromMemoryAsync()  assetBundle.LoadAssetAsync() | Resources.UnloadAsset(Object);  AssetBundle.Unload(); |



|  |  |  |
| --- | --- | --- |
| Application.dataPath | /data/app/package name-1/base.apk | Read Only |
| Application.streamingAssetsPath | jar:file:///data/app/package name-1/base.apk!/assets | Read Only |
| Application.temporaryCachePath | /storage/emulated/0/Android/data/package name/cache | Read Write |
| Application.persistentDataPath | /storage/emulated/0/Android/data/package name/files | Read Write |

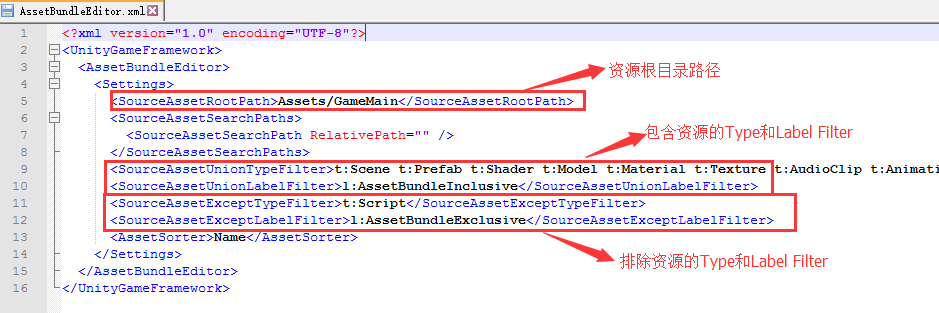
|  |  |  |
| --- | --- | --- |
| **资源管理方式** | **优缺点** | **热更新解决方案** |
| Resources目录 | 优点：方便管理和开发。不用处理复杂的引用问题。  缺点：无法热更。资源数量太多会导致程序启动变慢。 | 本来是不支持热更新。但可接入乐变SDK，机制是程序启动时Hook：先进行他们的资源更新；读取文件时Hook：先从他们的更新目录查找资源，找到自己读取返回给调用者。优点：简单，资源可单个更新。缺点：每次做包都得上传到他们的服务器做差异包分析，不可控。 |
| AssetBundle | 优点：可热更。  缺点：引用关系处理复杂。AssetBundle分组规划不好容易引起内存问题。 | 支持，自己写更新器。优点：灵活可控。缺点：管理比较复杂，AssetBundle中一个资源更新必须更新整个AssetBundle导致更新量大。 |

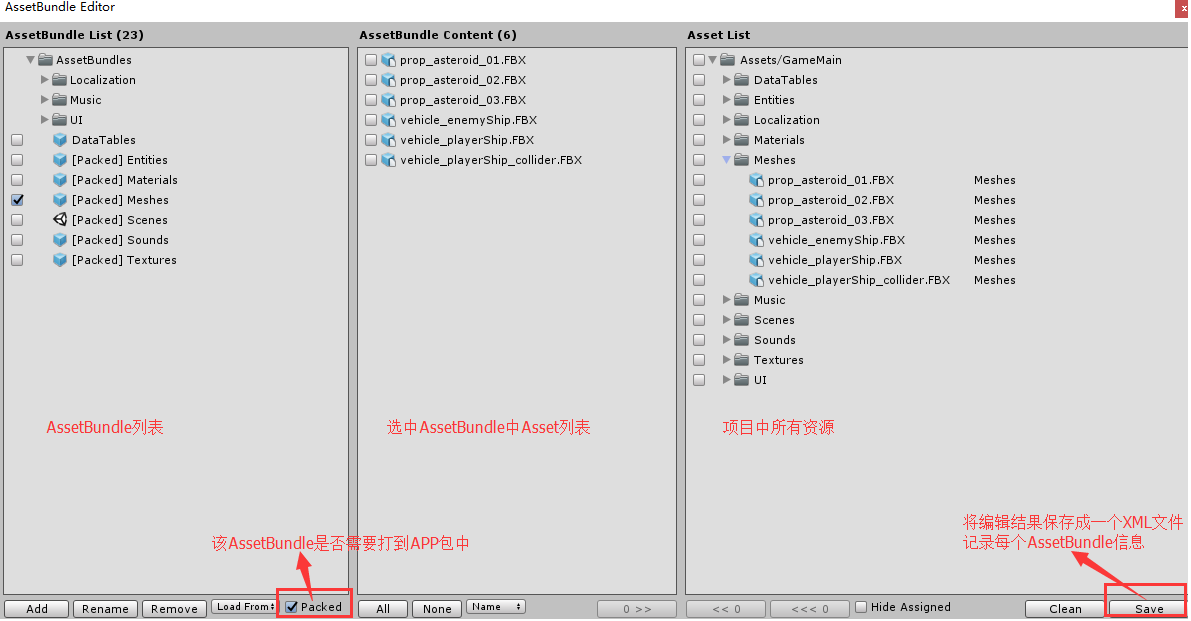
**利用AssetBundle方式的一个资源热更新解决方案（UnityGameFramework：http://gameframework.cn/）：**

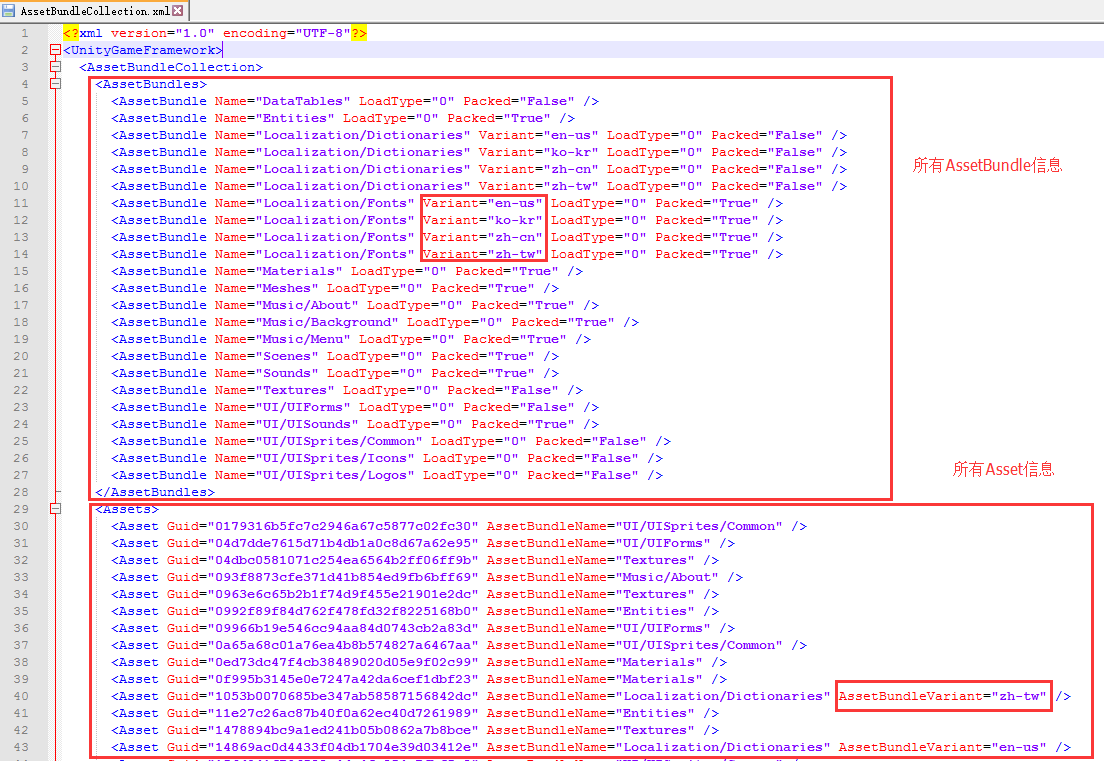
**一：资源打包**

**1：AssetBundle编辑**

第一次为项目编辑AssetBundle资源，需要为项目创建一个配置文件：

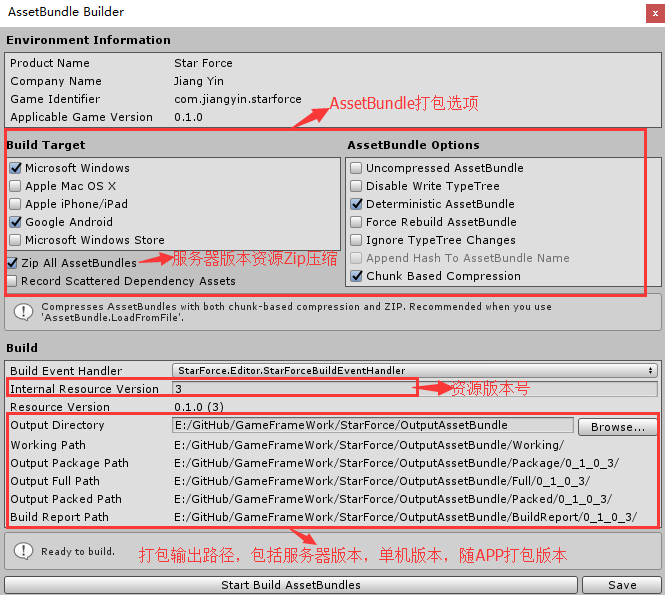




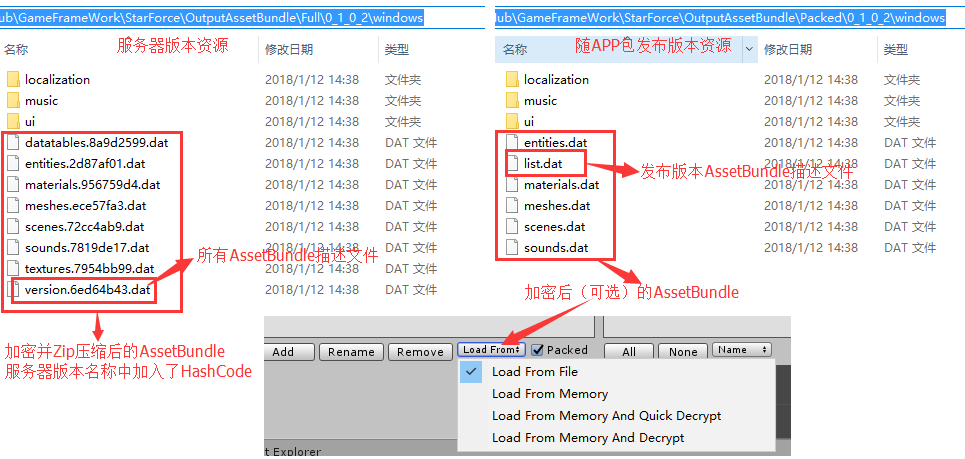


**2：AssetBundle创建**

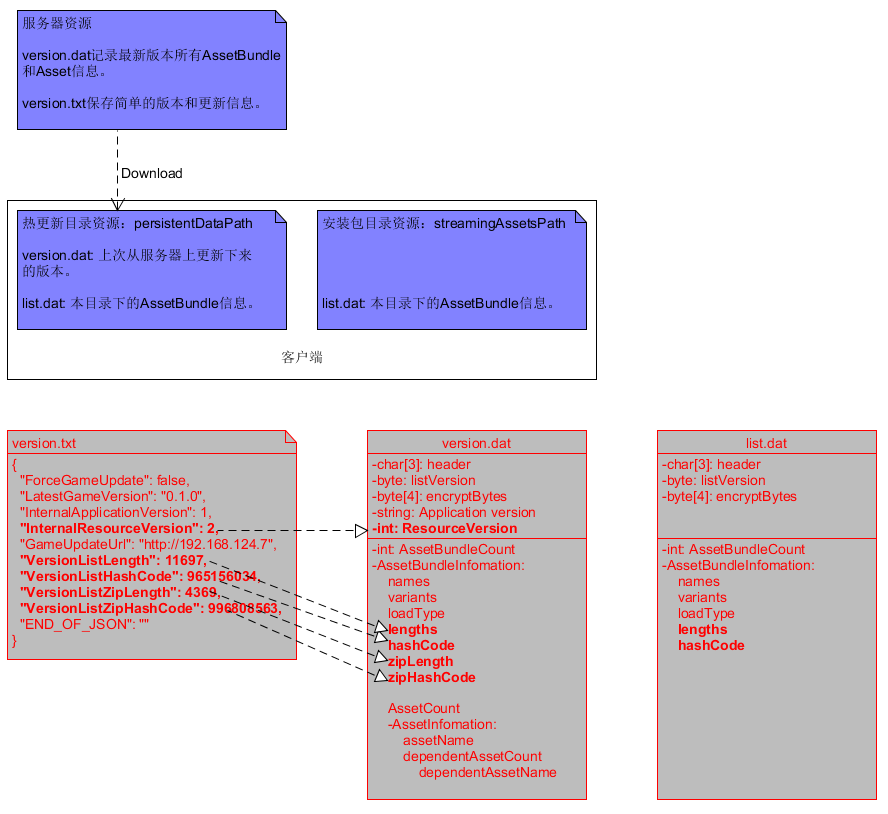




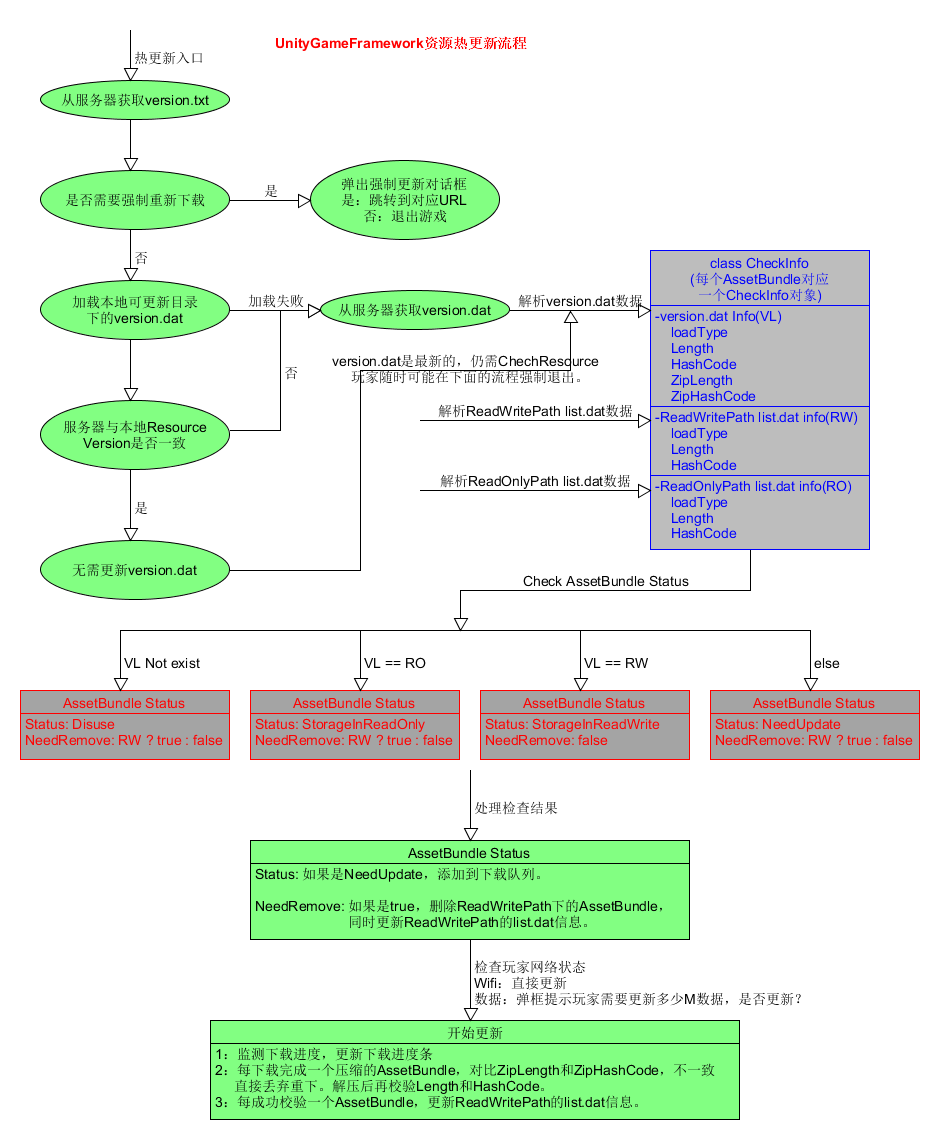
Build完成后，Full目录下对应平台资源上传到服务器上用于更新，Packed目录下对应平台资源拷贝到StreamingAssets目录下构建APP：



**二：资源更新**



资源热更新流程：



**三：Runtime资源管理：**

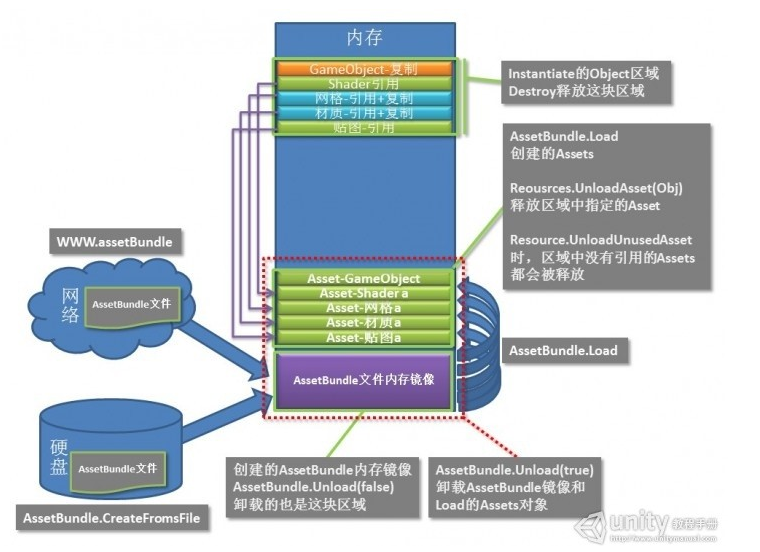
经过上面的资源检查和更新流程，我们构建出了哪些信息：

* 每个AssetBundle的信息，包括LoadType，Length，HashCode（如果选择了AssetBundle加密，秘钥就是该AssetBundle的HashCode），StorageInReadOnly（AssetBundle文件位置）。
* 每个Asset的信息，属于哪个AssetBundle。
* 每个Asset的Dependency信息，依赖哪些资源，分为属于某个AssetBundle的资源，和零散的资源。

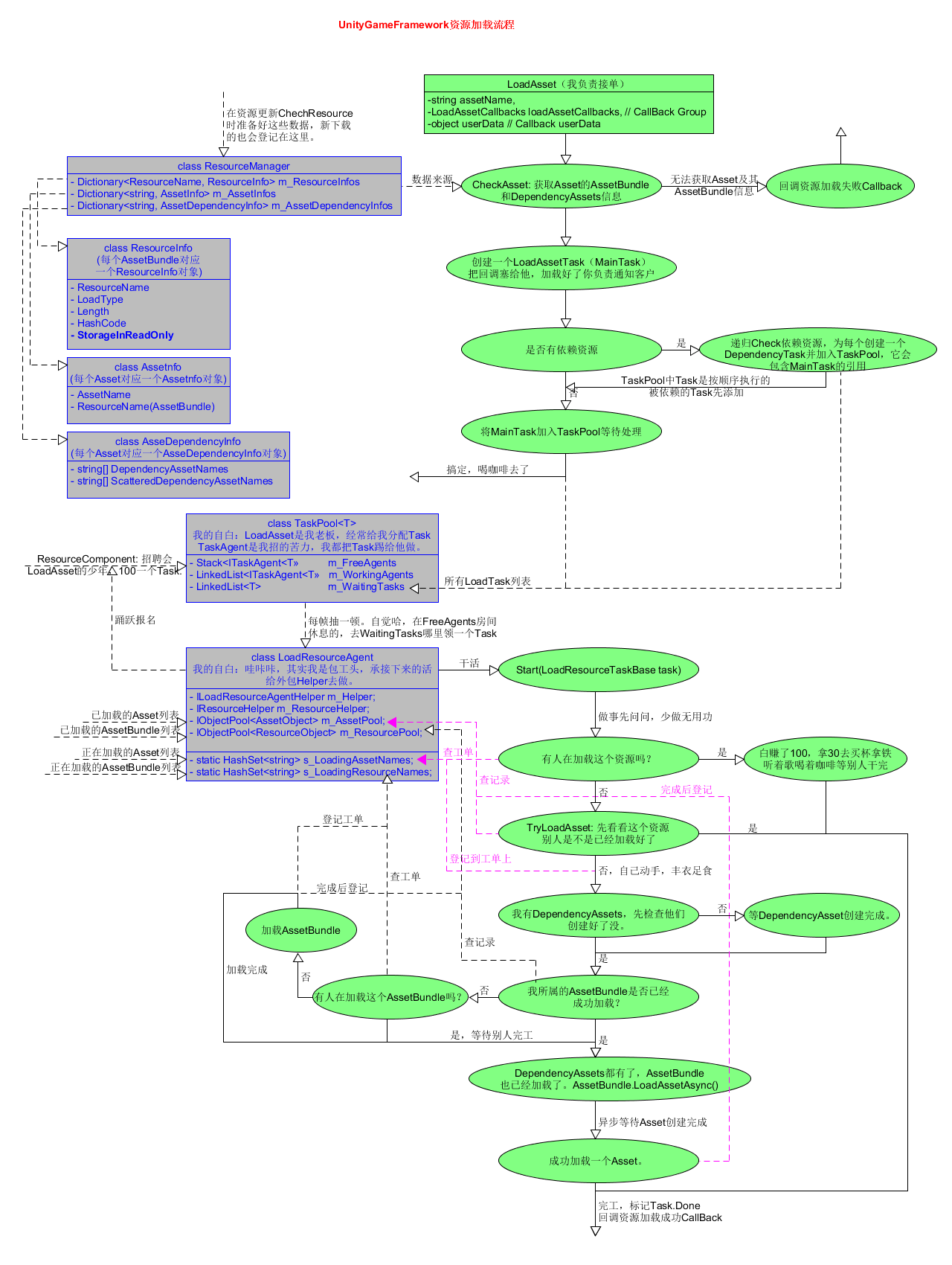
有了这些信息，是否发现我们要加载一个资源，要做的工作已经比较清楚了：

1. 检查该Asset依赖的资源是否已经加载，没有先加载依赖资源。这是一个递归的过程，依赖的资源可能依赖其他资源。
2. 检查该Asset所属AssetBundle是否加载，没有加载之。
3. AssetBundle.LoadAssetAsync(）加载该Asset。

在正式进入资源加载流程之前，先了解一下Unity资源的管理，比如Resources.Load或者AssetBundle.LoadAsset()一个资源和Instaniate一个资源是什么关系（http://www.cnblogs.com/88999660/archive/2013/03/15/2961663.html ）：



资源加载流程：



问题：AssetBundle, AssetBundle创建的Asset，以及Asset Instantiate出来的Object是怎样自动管理的呢？

* 所有Asset和AssetBundle都分别放在一个具有引用计数功能的ObjectPool中。
* 每个LoadAssetTask和LoadDependencyAssetTask成功加载一个Asset和DependencyAsset后，都会将AssetPool中对象引用计数加一。也会将每个Asset属于的AssetBundle的引用计数加一。
* 调用LoadAsset成功后，不用这个Asset了，要相应调用UnloadAsset，UnloadAsset里面会将该Asset和其依赖的DependencyAsset的引用计数减一。
* 如果一个Asset引用计数为0了，那么就会将其Unload，并将其所属的AssetBundle的引用计数减一。
* 如果一个AssetBundle的引用计数为0了，就会调用AssetBundle.Unload(true)将其卸载。

