（找了半天没有找到最好的代码，可能是我没有最好的 ，就用了最近我写得一个自我感觉还不错的代码）

大致实现思路：

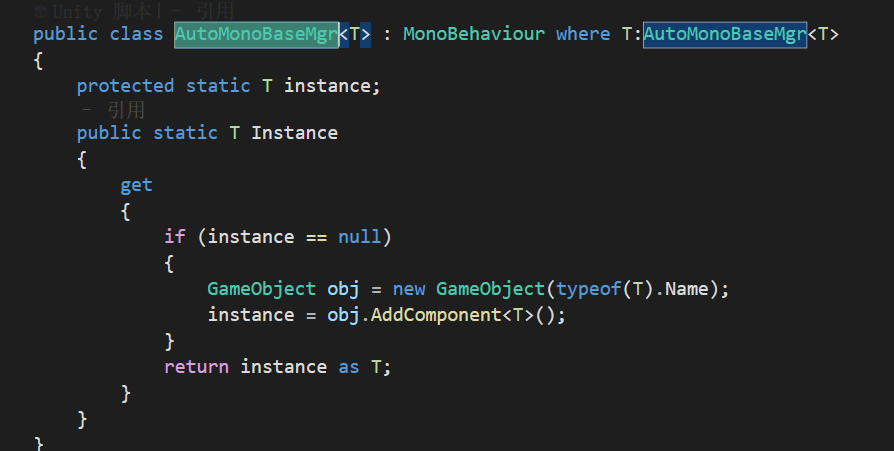
1. 用一个单例来给外部调用，来下载ab包（不重复的资源，本地没有的新版本资源，删除不需要的资源）然后通过学习和思考了解到了要用到ab包对比文件。
2. 设计ab包对比文件。
3. 创建ab包对比文件。
4. 上传ab包对比文件到服务端。
5. 下载ab包对比文件到客户端，存储ab包对比文件信息。
6. 解析ab包对比文件更新下载列表。
7. 根据下载列表来跟新远端的ab包资源。

ABUpdateMgr是一个AB包管理器类：通过它来管理AB包的上传下载。



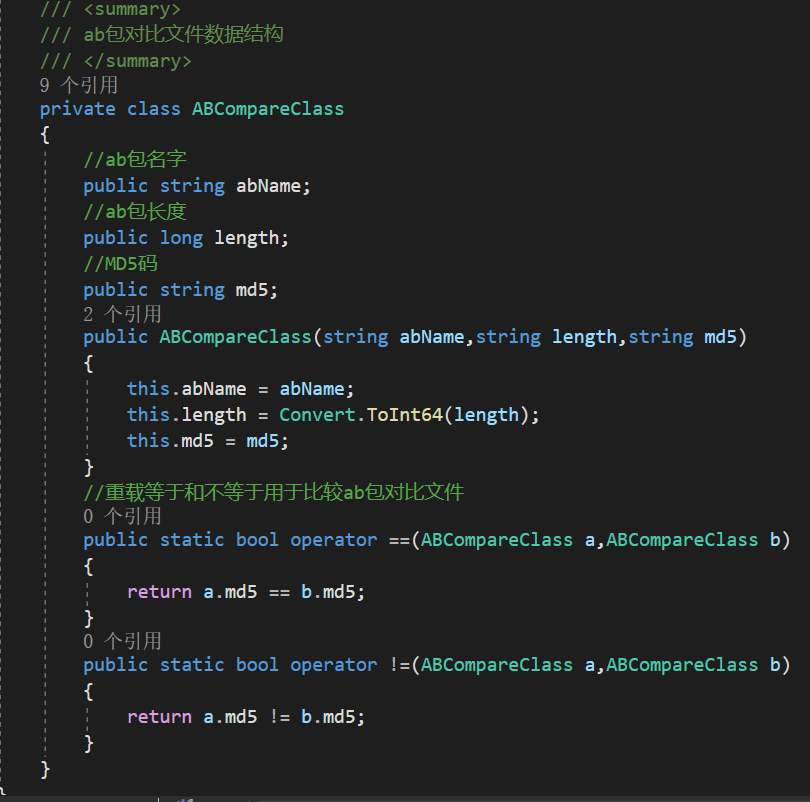
首先它得是个单例类，因为它在游戏中唯一并且方便外部调用它内部的方法。

单例模板类如下：



这是一个mono的单例模板，就是使用需要mono中的方法的单例，在使用时自动创建一个GameObject挂载到场景上。因为我制作的ab包管理需要使用协程方法所以用这种单例。

首先我们需要一个ab包对比文件类，因为我们需要一个对比文件在上传下载时去判断是否需要更新远端的ab包或者删除本地的ab包。



AB包对比文件包含：

AB 包的名字、长度、MD5码

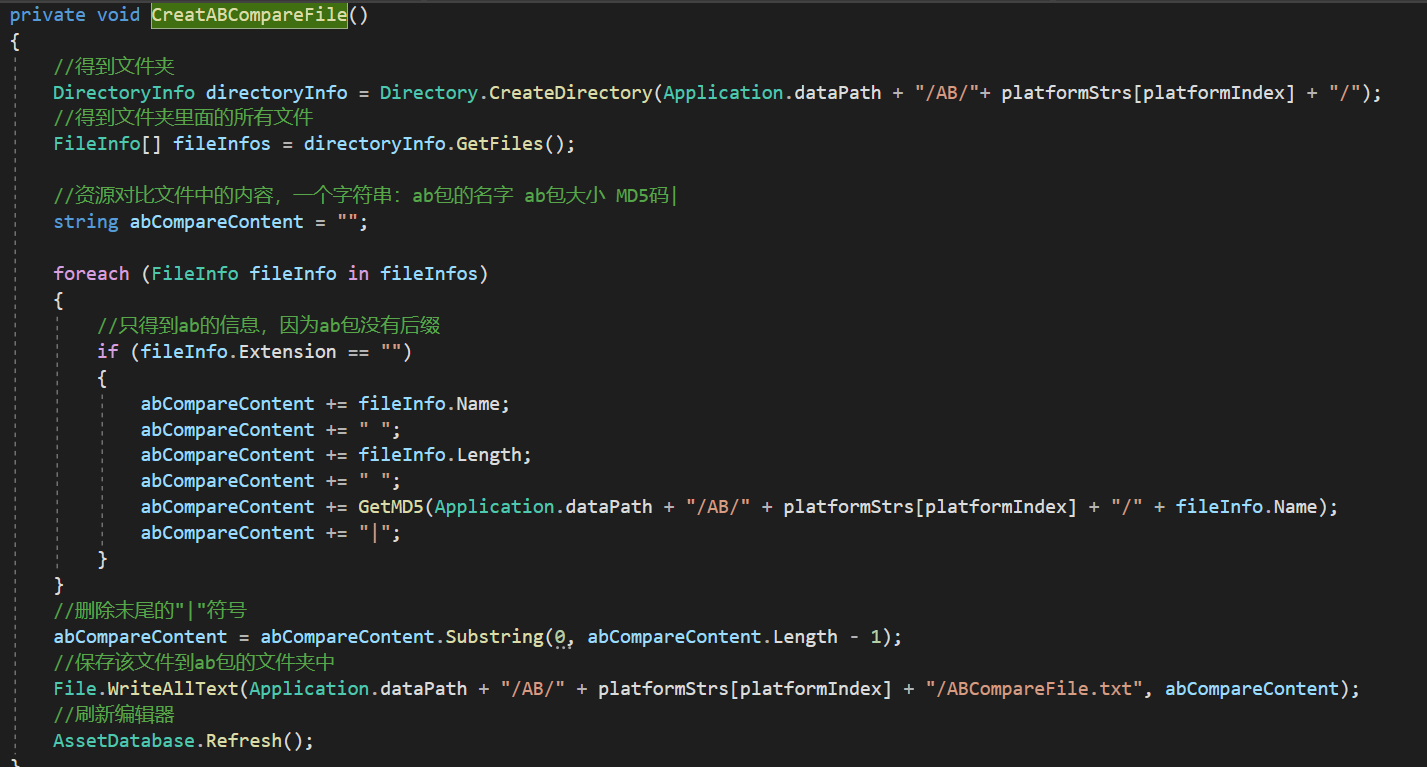
因为通过资源名和字节的长度我们无法判断资源是否更新，所以我们需要利用MD5的唯一性，来判断资源的更新。MD5码是由一个算法得到的唯一的结果，我们可以利用MD5码作为文件的唯一标识。同时我们要重装等于和不等于符号方便之后直接用类来对比文件。

我的ab包对比文件设计成一个字符串文件（用16进制存储，因为更短）：

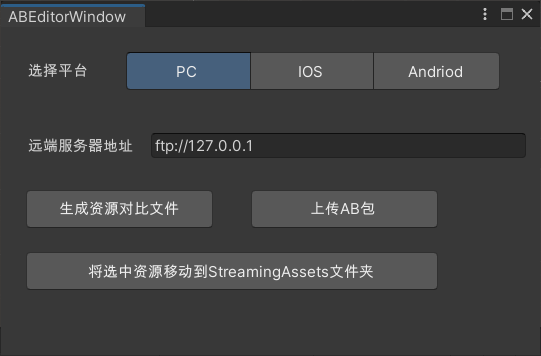
名字和长度和md5码通过空格分隔，不同的ab包通过|来分隔。

结构大概长这样：ab包1名字 长度 此文件的MD5码|ab包2名字 长度 此文件的MD5码|ab包3名字 长度 此文件的MD5码.........

在读取字符串的时候直接在读bytes的时候直接通过string中的Split方法来对应解析出其中的数据。ab包对比文件就是根据我们ab包目录下面的ab包来生成的



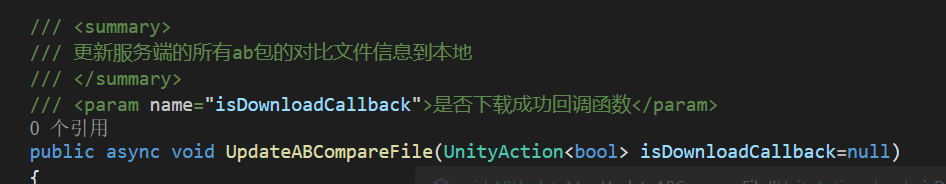
然后我们要生成ab包对比文件，所以我就又写了一个另外的脚本，实现了一个小工具。大概长这样：



然后就可以生成ab包对比文件然后上传到远端服务器

在每次我们登陆游戏需要跟新资源的时候，我们就可以把这个远端的ab包对比文件下载下来。

所以我书写了ABUpdateMgr里面的这个函数。

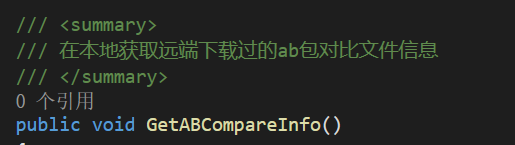


为了不使在下载的时候卡，所以我用了异步函数，同时给一个回调函数给外部判断是否下载成功，下载成功后能执行什么步骤（一般就是通过ab包对比文件的信息解析出来需要下载什么ab包来更新下载列表，然后发现没有什么ab包来删除本地ab包，然后根据下载列表从远端下载需要的ab包文件）。更新好的ab包对比文件信息用一个字典来存：

private Dictionary<string, ABCompareClass> abCompareDic = new Dictionary<string, ABCompareClass>();

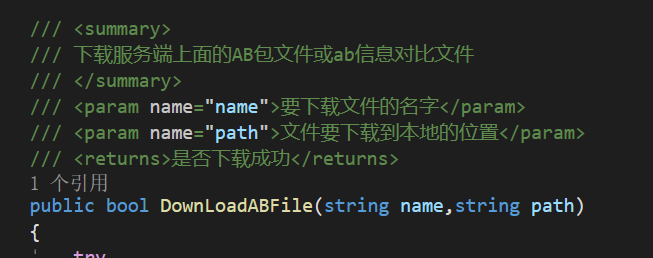
时间复杂度分析：因为是下载文件，一般只下载一次所以应该是0（n），虽然在外部还写了一个循环但是最多循环3次（下载不成功的时候重新下载）所以是0（n）。

空间复杂度分析：因为读数据需要用到bytes数组和字符串数组所以说应该是0（n）



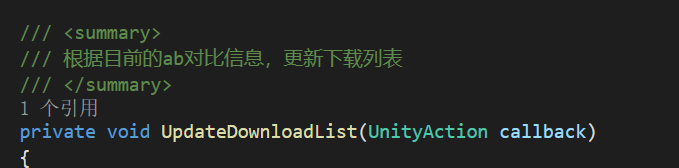
时间复杂度O(n)，空间复杂度O(n)

实际封装好的下载函数（执行下载操作）

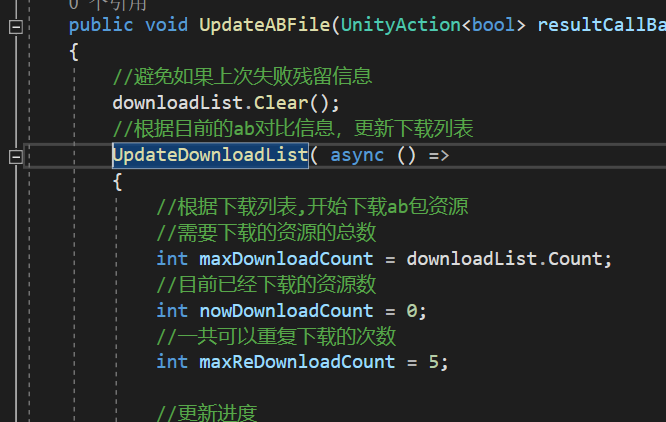


时间复杂度O(n)，空间复杂度O(n)

我们已经根据我们ab包文件夹里面的ab包文件信息生成了ab包对比文件，并且已经上传到了远端，而且这个ab包对比文件又被客户端给下载了，所以现在我们需要解析这个ab包对比文件，来跟新下载列表，最后才实际的开始更新ab包。



一般在下载ab包前跟新ab包对比文件中的信息，所以我把它拿到了根据下载流下载ab包的 函数里面。



同时我们通过在这个函数内调用协程配合UnityWebRequest来实现异步操作避免卡顿。我处理了两个文件夹persistentDataPath和streamingAssetsPath，streamingAssetsPath存放的是初始发布版本自带的ab包文件，persistentDataPath存放的是后来下载过的ab包文件，之所以要处理这个文件夹streamingAssetsPath，是因为我们要根据后续开发中的实际来删除或者避免下载游戏中已经存在的资源。

最后我们把所有的要下载的ab包的名字都存放到private List<string> downloadList=new List<string>();这个数据结构中。这个过程时间复杂度和空间复杂度都是O(n)。

最后有了下载列表我写了这个函数来根据下载列表中的信息下载ab包

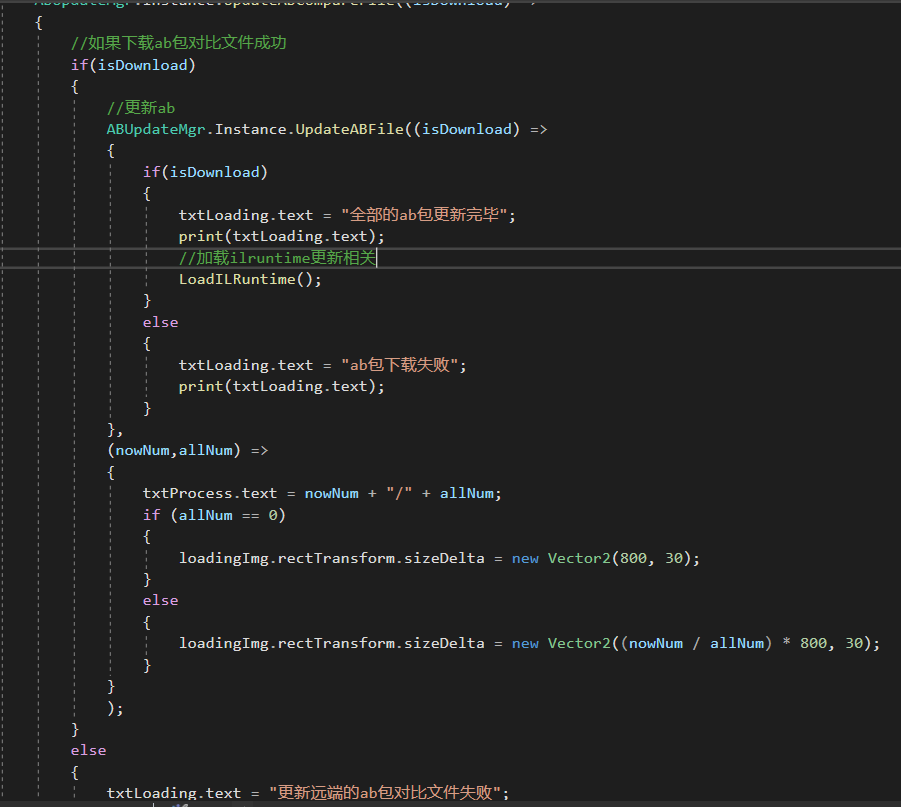


第一个参数是一个回调函数，还是用来判断是否跟新成功和处理函数执行后的逻辑，和上述的函数的参数类似。

第二个参数也是一个回调函数，主要用于在外面跟新下载的进度，比如在游戏登录界面跟新游戏的时候，用来做进度条。

时间复杂度O(n)，空间复杂度O(n)。

具体实现调用：



至此就完成了这个脚本的开发。