# 序列化记录及使用序列化保存场景

该文档为初步学习UE4序列化以及将其应用于自己的小游戏中对场景进行保存时所做的一些总结与记录，只是序列化的九牛一毛，并且肯定会有理解错误和不足的地方，请多指正。

网上都查得到的概念性东西以及偏官话的基础部分，在此都不做累赘，会比较初步的带过。

## 序列化说明

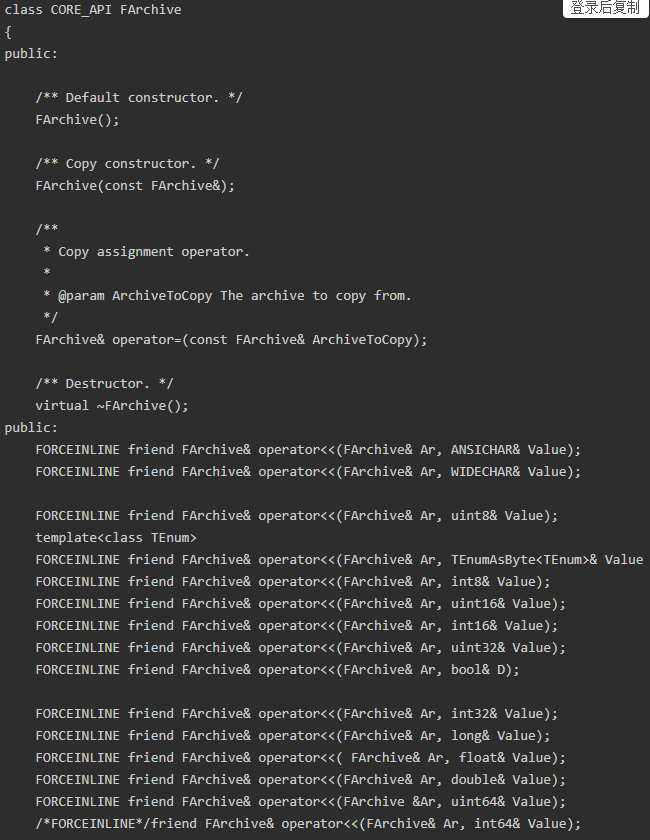
序列化是指将对象转换成字节流，从而存储对象或将对象传输到内存、数据库或文件等的过程。 它的主要用途是保存对象的状态，以便能够在需要时重新创建对象。 反向过程称为“反序列化”。 （通俗来说就是保存和读取的过程分别为序列化和反序列化）

Ue4的序列化使用了访问者模式(Vistor Pattern)，将序列化的存档接口抽象化，其中FArchive为访问者， 其它UObject实现了void Serialize( FArchive& Ar )，接口的类为被访问者。FArchive可以是磁盘文件访问, 内存统计，对象统计等功能。

序列化中最为重要的结构即是FArchive类，该类有众多的继承：

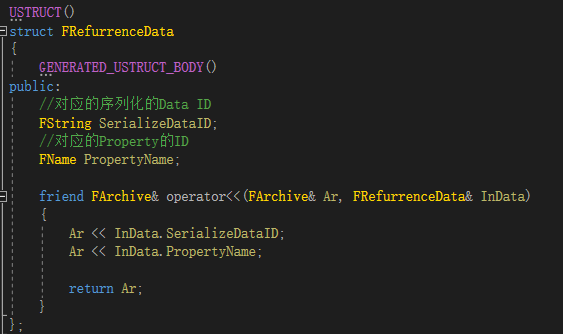


从FArchive的代码中可以看到（该代码只展示了接口），



该类已经实现了一些基本类型的存取，通过重定义<<操作符进行存取。

当我们自定义结构需要序列化时，也必须要重载操作符，例如

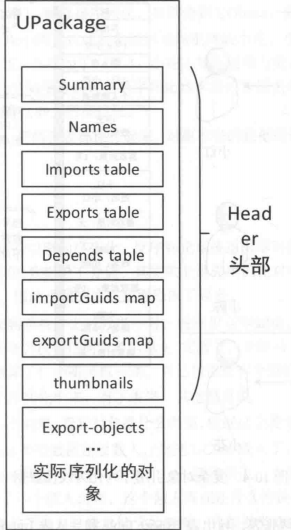


### 初步分析序列化与反序列化

序列化与反序列化都是通过Serialize来进行的，序列化的过程通俗点来说，函数会遍历该UObject上被标识为UPROPERTY()的对象，并将其写为2进制的数据进行存储。

#### UPackage对象简易说明

在我们游戏中生成的蓝图也好，地图也好，都会存储为uasset或是umap格式资源，而每一个资源对应的就是一个UPackage，存储一个Upackage时，会将该资源下对应的所有对象存储到该包中，一个包的结构如下图所示：



File Summary 文件头信息。

Name Table 包中对象的名字表。

Import Table 存放被该包中对象引用的其它包中的对象信息(路径名和类型)。

Export Table 该包中的对象信息(路径名和类型)。

Export Objects 所有Export Table中对象的实际数据。

有时候我们FindObject会出现找不到的情况，就是包没有加载，此时可以使用LoadObject进行包的加载。

#### 序列化反序列化

有了包的概念之后，我们以场景举例，对序列化与反序列化进行分析。

##### 序列化

在包最前方有两张表，导出表 Export Table和导入表 Import Table，前者相当于该类自身的属性名单，后者相当于外部引用的属性名单，例如引用的其他类指针等。

当序列化一个对象时，如果遇上UObject指针，不会直接记录地址值，而是记录下一个编号，也就是会对UPROPERTY打上标记，标记在序列化过程中非常重要，他会标记一个属性是否有是其余对象的引用，是否需要存储等等等等。

如果遇到对象为外部的包引用，则会进行记录与标记，存放导ImportTable中。

##### 反序列化

当反序列化时，遍历包中的对象列表，对每一个对象进行反序列化。

先根据GetClass函数获取当前类的信息，通过GetOuter函数获取Outer。这个Outer实际上指定了当前UObject会被当作为哪一个对象的子对象进行序列化。

判断当前等待序列化的对象的类UClass的信息是否被载入，没有的话:

a. 预载入当前类的信息；

b. 预载入当前类的默认对象CDO的信息；

也就是说，会先New一个该类型的空白对象放在那里，接着载入名字，设置类对象信息等等，之后开始载入对象的所有脚本成员变量信息，此时因为对象已经创建出来了，所以可以根据反射查找到需要设置数据的UPROPERTY，对应的函数为

SerializeScriptProperties，序列化在类中定义的对象属性。

当对象信息中碰到了UObject类型的成员变量时，根据包中记录的导出表 Export Table和导入表 Import Table，判断是包内的OJB还是包外的OBJ：

如果是包内的，查找该对象是否已经被初始化，如果已经初始化完成，则直接将指针对应替换，指向已经生成的对象。如果没有，则先New一个对应类型的UObject放在那里，设置好指针，等待该UObject序列化完成。

如果该对象在包外，则判断该包是否已经载入内存，如果没有则把包载入内存，如果已经载入完成，则直接在该包内进行查找并且完成指针替换。

此过程中，会不断的调用各个Object的Serialize函数进行指针替换，最终完成了一个包的反序列化。

#### 实例分析与问题分析

有这样的情况，我们在自己的代码中对对象进行序列化保存，如下所示：

