**http://game.ceeger.com/forum/read.php?tid=13479**

[**[iOS]**](http://game.ceeger.com/forum/index.php?c=thread&fid=27&type=92)**[原创] 在ios android设备上使用 Protobuf (使用dll方式)**

**如果你的工程可以以.Net 2.0 subset模式运行，请看这个帖子中的方法。**

地址：<http://game.ceeger.com/forum/read.php?tid=14359&fid=27>

如果只能以.Net 2.0下运行，就可以继续往下看了。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

protobuf是google定制的一个协议，可以方便地进行数据的序列化和反序列操作。(支持多个平台间通信)

可以用它来进行数据的本地缓存

以及 网络协议的传输 等等。

要了解protobuf协议看下面两个网址：

<http://code.google.com/p/protobuf/>

<https://developers.google.com/protocol-buffers/docs/tutorials>

Unity有两个Protobuf库可以用分别是 protobuf-csharp-port, **protobuf-net** (后者效率更高)

两个库使用都可以在android上运行正常，但是在ios上面会出现下面的异常：

*ExecutionEngineException:*

*Attempting to JIT compile method*

JIT意思就是运行时编译，而ios上是aot (full)编译，因此任何库只要使用jit方式都无法在ios上运行。

要了解jit以及aot，请看下面的网址:

JIT:<http://en.wikipedia.org/wiki/Just-in-time_compilation>

AOT:<http://en.wikipedia.org/wiki/AOT_compiler>

<http://www.mono-project.com/AOT>

经过尝试之后，可以用protobuf-net的**预编译**来在ios以及android设备上运行

接下来按步骤讲解，最后会附上测试工程。

一、 下载protobuf-net 及配置

 到<http://code.google.com/p/protobuf-net/downloads/list> 去下载protobuf-net最新发布版本(写这篇教程的时候最新版本是r668)

 下载之后解压到你的硬盘的某个目录 ， 最好是将 **Precompile\precompile.exe** 以及 **ProtoGen\protogen.exe** 两个文件加入到环境变量中 ， 之后方便使用。

 我们还需要CoreOnly\ios中的三个文件，之后会用到，而且最终会放到unity工程下。

二、生成cs代码

 使用protogen 命令行来生成代码 (precompile以及protogen都只能在windows下运行，在mac上可能可以通过mono来运行这个exe，但是没有试过。)

**使用例子如下**：

 protogen -i:Test1.proto -i:Test2.proto -i:Test3.proto -o:Output.cs -ns:com.fbmly.model

-i 是输入文件,可以有多个

-o 输出的cs文件, 只能有一个..如果-i有多个 会将所有的代码生成到这一个cs文件当中

-ns 命名空间   最好使用，如果不使用每次生成的默认命名空间是proto的文件名。

通常来说 只要把所有的数据结构生成到一个cs就行了（要不然使用多个-i ， 要不就把所有protobuf定义写在一个proto文件中），这样之后之后操作比较方便一些。

经过这一步你就得到了生成之后的cs文件。

三、编译dll库

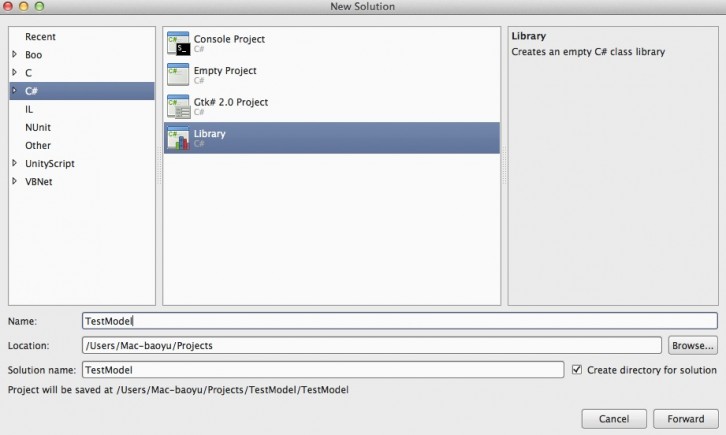
使用MonoDevelop工具 将上一步生成的cs文件编译成dll库.

1.创建一个新的工程 (File->New->Solution)

2.在弹出的对话框中选择 C# 再选择 Library.

3.在下方填写好工程的名字 (这个名字是生成dll的名字,所以要起好)

4.点击forward (之后还要点一次ok)



5.删掉默认生成的MyClass.cs文件，有必要的话在AssemblyInfo.cs里填写一些版权信息。

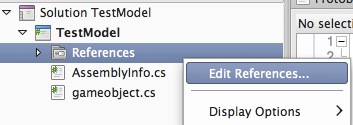
6.将上一步用protogen生成的cs文件 加入到工程 (在左边工程名 上右键 就可以添加)

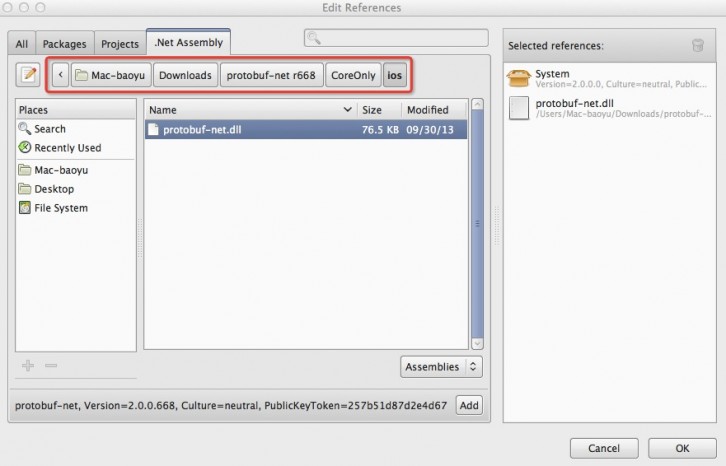
7.添加protobuf-net引用库

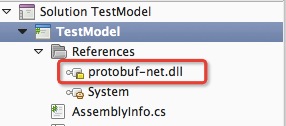
   点击工程的References->Edit References

   然后选择刚才的CoreOnly\ios下的dll库 ， 双击库文件就可以加到右边

   最后检查下References里面如果有protobuf-net.dll就正确了。

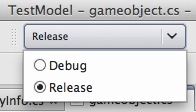






8.都做好后, 点击Build->Build All就可以在当前工程生成dll了

   不要忘记 debug 和 release的选择 ，测试完后应该重新拿release再编译一次的。



9. 左下角提示成功 就 会在当前工程目录下生成dll文件了 (我这里是 TestModel.dll)

这个dll存放的是你所有的数据结构。

四、预编译序列化库

 上一步生成了数据的dll文件，这一步需要用到上一步的dll文件来生成专门序列化的dll文件。

**首先要保证TestModel.dll和编译时所使用的库在同一目录** ( 上一步生成的dll文件目录，会自动把protobuf-net库文件也复制过来 , 所以只需要记住这一点就行了。 比如你要是复制到其他目录进行操作， 这点会很重要 )

然后打开命令行， 定位到dll工程目录下的bin\Debug 或 bin\Release目录

然后执行以下命令（**修改成你所设置的名字**）：

precompile TestModel.dll -o:ProtobufSerializer.dll -t:com.fbmly.ProtobufSerializer

TestModel.dll就是上一步生成的dll文件，-o是生成的文件名  -t是在ProtobufSerialize.dll中所生成的序列化类的类名 支持命名空间，不用命名空间就直接写类名就可以了。

 如果最后有 All Done 提示， 就代表生成成功了。

[复制代码](javascript:;)

|  |
| --- |
| void OnGUI( ) { if ( GUILayout.Button ( "Test" , GUILayout.Width ( 200f ) , GUILayout.Height ( 200f ) ) ) { Fbmly data = new Fbmly ( ); data.mmm = 10; ProtobufSerializer serializer = new ProtobufSerializer ( ); //Serialize byte[] buffer = null; using ( MemoryStream m = new MemoryStream ( ) ) { serializer.Serialize ( m , data ); m.Position = 0; int length = (int)m.Length; buffer = new byte[length]; m.Read(buffer, 0 ,length); } Fbmly newData = null; //Deserialize using ( MemoryStream m = new MemoryStream ( buffer ) ) { newData = serializer.Deserialize ( m , null , typeof ( Fbmly ) ) as Fbmly; } Debug.Log ( "newData.mmm=" + newData.mmm ); } } |

测试工程已经在touch4上测试通过，并且工程设置为.Net 2.0 Subset

如果各位在真机上出现错误，不妨设置到.Net 2.0试试。

这种方式已经在游戏项目中验证过了，复杂的数据也是没有问题的（游戏项目用了500多个proto定义） ， ios和android都可用。

最后附上工程包：里面附了一个工具类，可以序列化的时候使用, 用的时候只需要修改一下就可以了，因为里面维护了一个序列化类的单例。