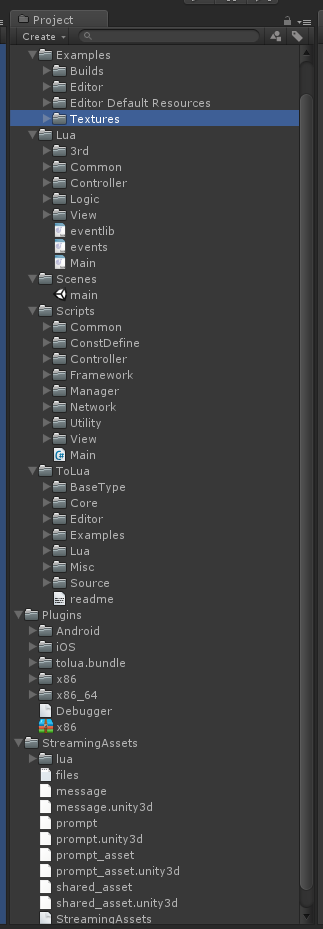
# 框架工程目录说明



1. **Examples** ：框架自带的Demo例子，如果只需要框架的同学，里面的资源可以删除掉。   
   —Builds：里面都是一些NGUI/UGUI定义的图集啊、Prefab等资源。用于生成assetbundle而准备的资源。   
   —Editor：里面是例子用到的一个新手引导步骤演示的编辑器脚本。   
   —Editor Default Resource：目录是新手引导步骤对话框用的的图片资源。   
   —Rsources：例子里面用于演示的一个内建的GUI容器的Prefab。   
   —Textures：里面是Buidls目录里面图集的原图文件。
2. **Scenes**：里面一个main场景文件也是主场景
3. **Lua**：框架自带的Lua源码目录，用户自定义的Lua脚本也就是放在这里面，最后打包的时候，打包脚本会将其按目录结构生成到StreaminAssets(服务器更新的内容都放这里面)目录里面去，然后在将其上传到游戏的Web服务器上面，用于准备被每个游戏客户端下载更新他们本地的Lua脚本。达到热更目的。   
   —3rd：里面是第三方的一些插件lua、实例源码文件，比如：cjson、pbc、pblua、sproto等。   
   —Common：公用的lua文件目录，如define.lua文件，一些变量声明，全局配置等，functions.lua常用函数库，通讯的protocal.lua协议文件。   
   —Controller：控制器目录，它不依赖于某一个Lua面板，它是独立存活在Luavm中的一个操作类，操作数据、控制面板显示而已。   
   —Logic：目录里面存放的是一些管理器类，比如GameManager游戏管理器、NetworkManager网络管理器，如果你有新的管理器可以放到里面。   
   —View：这是面板的视图层，里面都是一些被Unity调用的面板的变量，走的是Unity GameObject的生命周期的事件调用。

总结：基本上，我们只需要在 Controller 和 View 添加自己的lua脚本

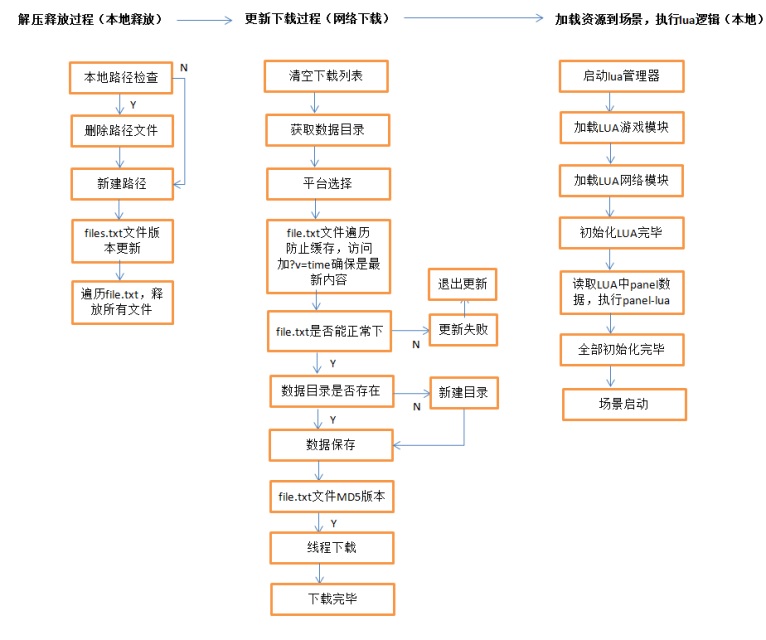
1. **Plugins**：ulua底层库所在的目录，里面存放的是不同平台的底层库，之所以ulua效率高，就是它是纯c的lua虚拟机，而不是c#解释型的。   
   —Andriod：**[安卓](http://lib.csdn.net/base/android" \o "Android知识库" \t "_blank)**lua虚拟机底层库，里面分为armv7-a与Intel x86平台。   
   —**[iOS](http://lib.csdn.net/base/ios" \t "_blank" \o "iOS知识库)**：里面就是苹果lua虚拟机底层库。   
   —ulua.bundle：里面是Mac机器的底层库。   
   —x86：里面是Win32/Linux32位机器的lua虚拟机底层库。   
   —x86\_64：里面是Win64/Linux64位机器的lua虚拟机底层库。

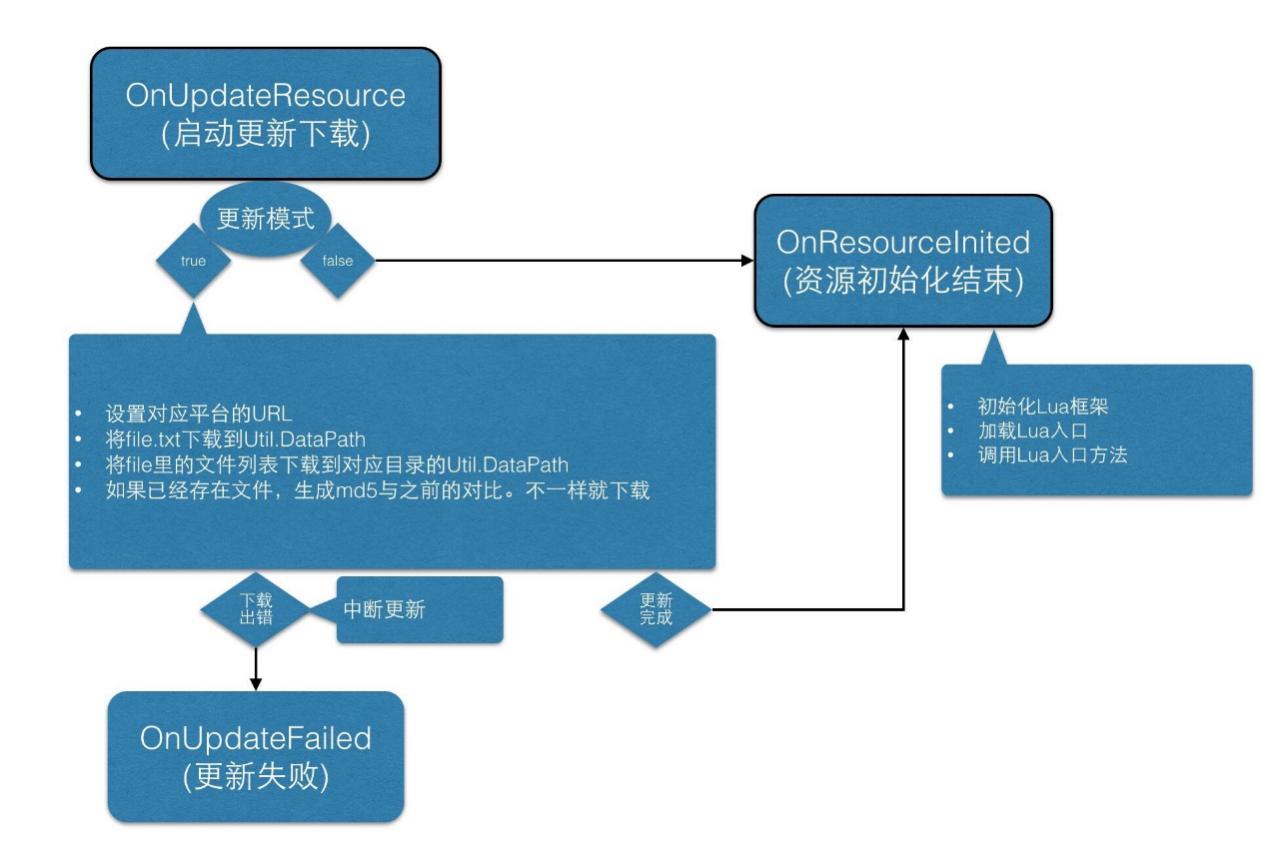
—Cstring, Debugger：其他使用的类库

当我们打包工程时，需要将其他平台的底层库删除，才能打包，比如打包apk，需要将 IOS，x86等其他删除

1. **Scripts**：框架的C#脚本层，之所以这个目录跟lua目录都放在最外层，为了让用户一眼都能找到，明白是什么。   
   —Common：框架的公用定义类。LuaLoader（跟lua加载有关的类）、与luavm通知unity游戏对象的“LuaBehaviour”桥类。   
   —ConstDefine：常量定义目录，AppConst（应用常量）ManagerName（管理器名称）NotiConst（通知常量，用于mvc消息通知）。   
   —Controller：控制器目录，分为StartUpCommand启动控制器，跟常用逻辑控制器。框架接收到启动命令后，直接在启动命令里面注册所有的管理器类。   
   —Framework：经过修改过的PureMVC的框架文件。   
   —Manager：Unity提供基础功能的管理器类，音乐、面板、线程、资源等众多管理器。   
   —Network：网络的常用辅助类，ByteBuffer字节操作封装类，网络协议类，转换器类。   
   —Utility：常用工具类。   
   —View：C#用的PureMVC的视图层。
2. **ToLua(低版本的可能是uLua)**：ulua/cstolua的核心目录，里面还有经过我们修缮后ulua的基础使用例子，用户初学者最佳。   
   —Core：顾名思义，ulua的核心目录，所有c#与lua的交互都是通过它进行调度的。   
   —Editor：这是供cstolua去反射定义Wrap文件列表的工具类目录。   
   —Examples：经过我们修改增加后ulua自带的例子。   
   —Source：这个是cstolua的核心目录，里面有Base核心目录，与动态生成用于存放LuaWrap类的缓存目录。

# 热更新流程说明





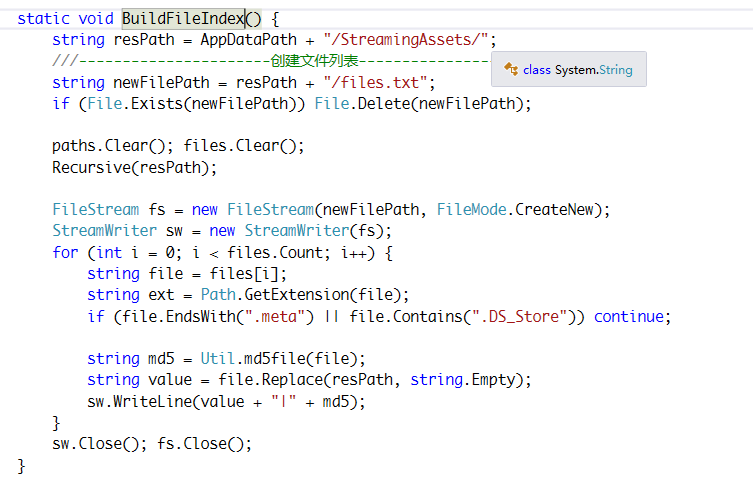
1. 解包：将资源打包到StreamingAssets中，因为Unity打包生成Apk（安卓）/Ipa（苹果）的时候，此目录会原样地打进安装包中，游戏客户端框架可以通过代码读取到里面的资源，并且把里面的资源复制到玩家的手机本地存储里面打包的资源分为素材资源与代码资源，这两部分的打包，框架都集成了，你可以直接修改里面的脚本逻辑适应自己的游戏项目，我们打开ulua/Editor/Packager.cs打包脚本。这里面根据不同的平台打包相应的资源。
2. 打包：打包的资源分为素材资源与代码资源，这两部分的打包，框架都集成了，你可以直接修改里面的脚本逻辑适应自己的游戏项目，我们打开ulua/Editor/Packager.cs打包脚本。这里面根据不同的平台打包相应的资源。

++

（1）在StreamingAssets目录下面新建lua目录，用于存放编码后的lua文件。   
（2）遍历Lua目录下面所有的lua文件，并且根据目录结构创建相应目录树。   
（3）如果AppConst.LuaEncode 设置了编码开关，就启动编码操作，否则直接复制源码过去。



（4）通过代码打包prefab和所用到的资源。



（5）遍历StreamingAssets目录，计算lua文件和ab包MD5值，生成file.txt文件

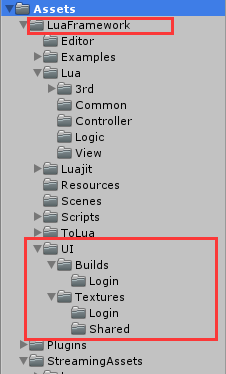
1. 更新：流程参考上述图片

Scripts/Manager/GameManager.cs文件，定位到IEnumerator OnUpdateResource()

（1）它一上来就做些初始化的操作，找到更新的地址URL等。   
（2）请求更新列表文件files.txt，因为里面存放了上面生成的目录结构及其MD5/CRC的信息。   
（3）分析Web服务器上的files.txt文件内容，然后遍历检查本地的文件结构是否完整、MD5是否匹配。   
（4）如果MD5不匹配，或者本地文件不存在，就开始创建一个下载请求，并且将它传递给线程，请求线程下载。   
（5）本地协程会每一帧查询线程下载完后是否将下载的文件名存放到下载文件列表中，如果找到，继续下载下一个，直至全部下载完成。   
其实当全部更新完成后，此时手机存储的文件结构及其内容已经是最新的了，客户端程序便可以顺利启动，完成了热更的最后一环。

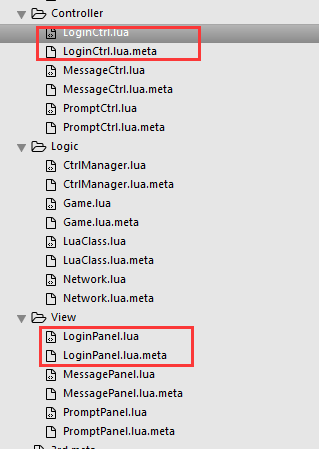
# 使用

1. 制作登陆界面

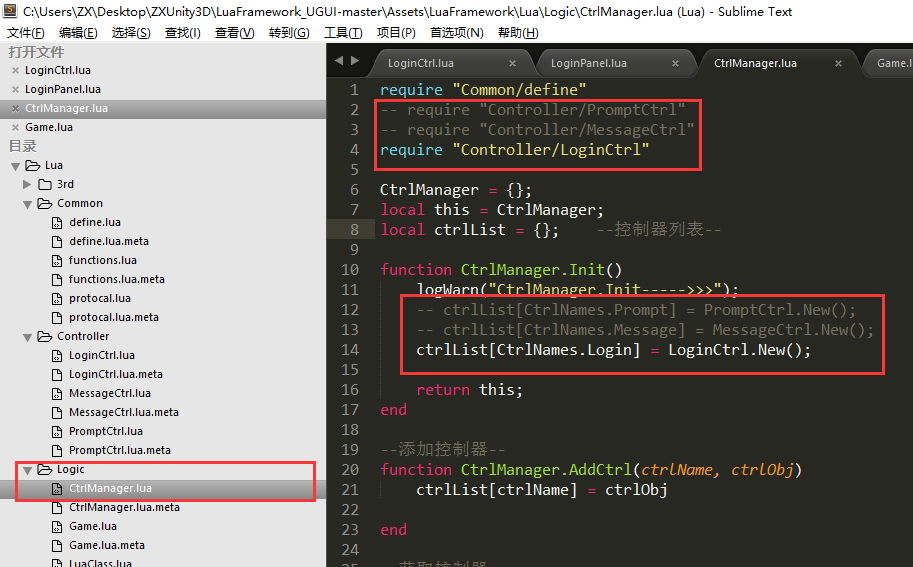


在工程中参考Examples制作如下目录，将制作登录界面的资源导入其中，并制作UI界面，做成预设物

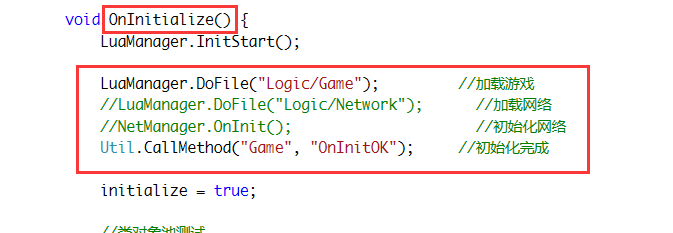
1. 添加Lua脚本：在Lua/Controller 和Lua/View 中 分别创建 LoginCtrl和LoginPanel ，可参考示例脚本，其中脚本和之前预设物的命名规范一定要注意。在这里，我们使用sublimne来进行lua脚本的编写

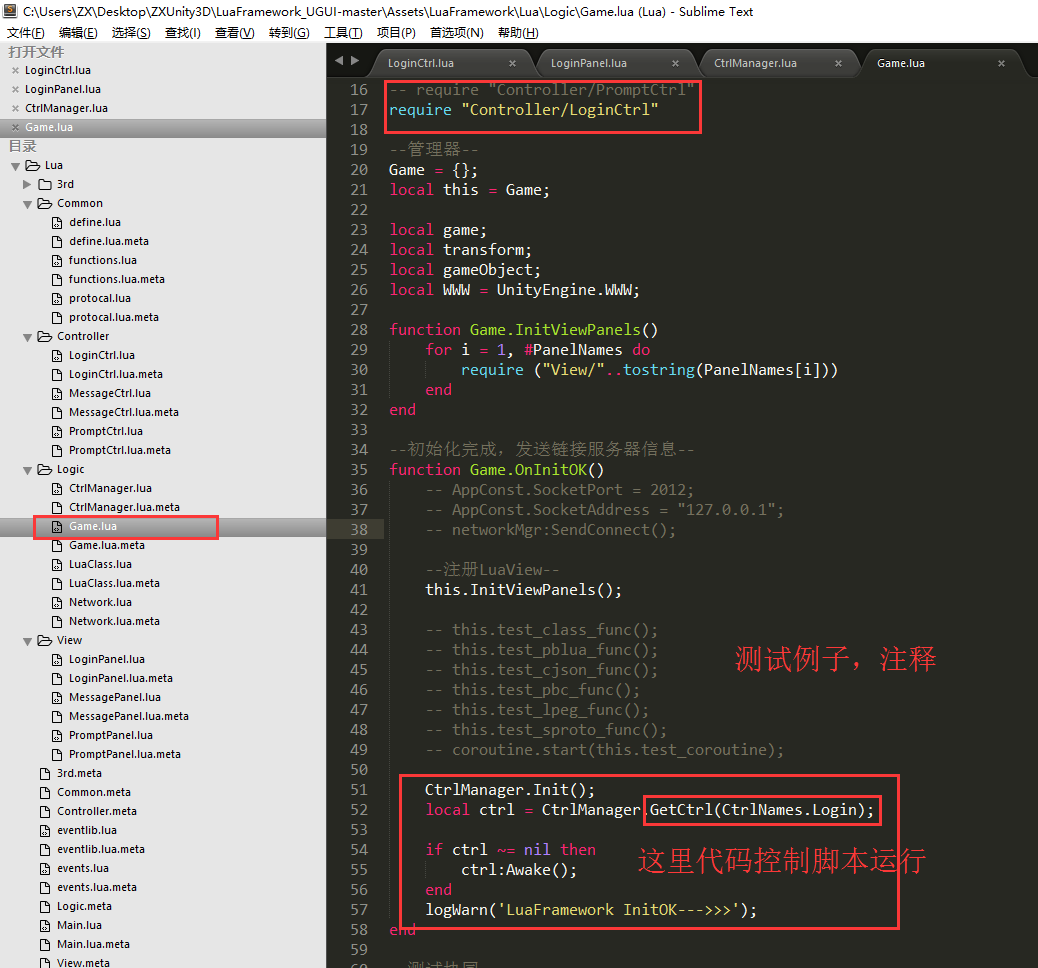


1. 在Logic/CtrlManager.lua 中 Init方法中，添加 Login，并注释例子代码

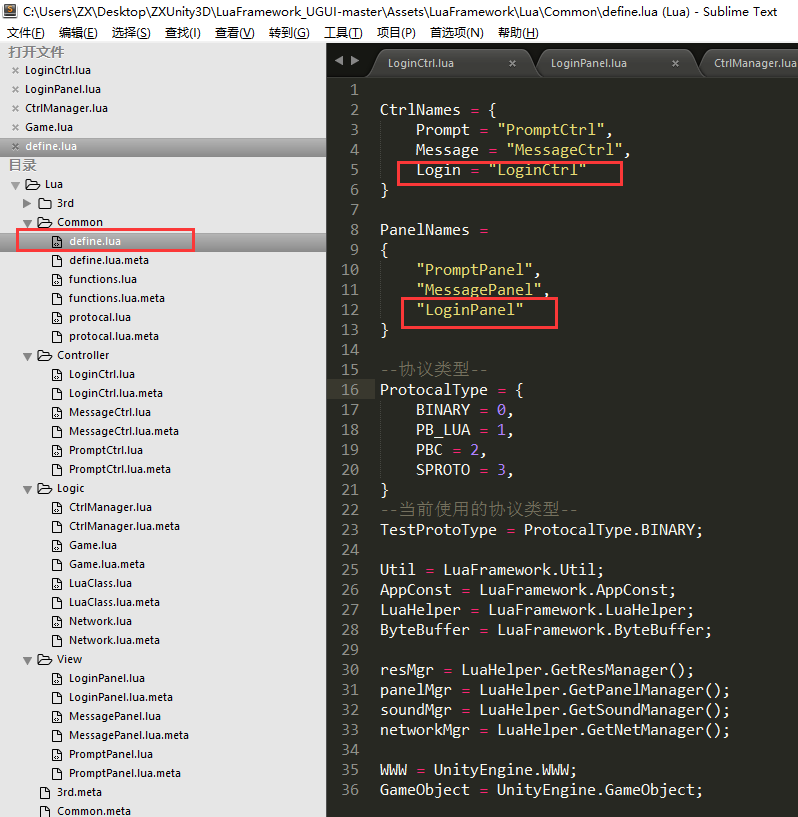


1. 在Logic/Game.lua 中 完成配置，Game.lua中的OnInitOk方法，作为lua脚本的主入口。是由C#脚本 GameManager控制的



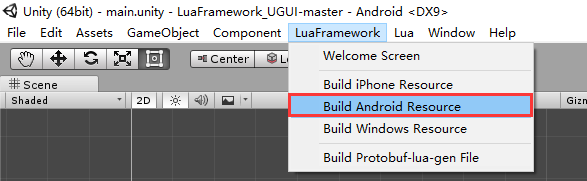


1. 在Common/define.lua 中完成配置

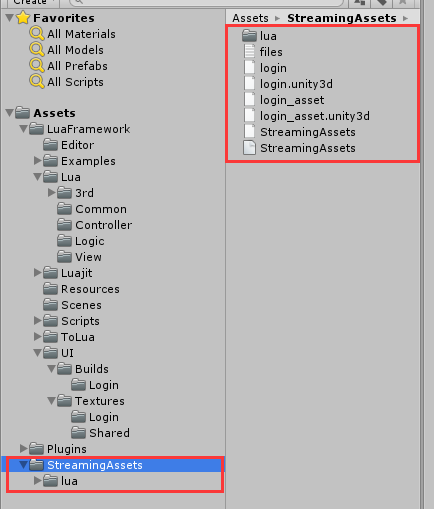


1. 经过上述几步，配置基本完成，我们需要根据平台，进行资源打包。在Packager.cs中，通过代码进行设置，然后生成资源





可以看到在StreamingAssets中，打包好的文件，已经生成了



1. 运行游戏，可以看到login界面出现
2. 在main场景中，创建名为GlobalGenerator的空物体，可看到更新信息