# 说明

## 主旨

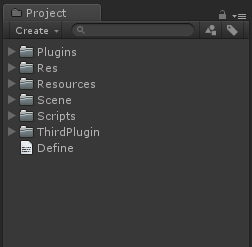
xyframework 主要是为了开发, app 或者 小游戏的一个简易集成方案，作者本人从项目中脱离这个方案时候本着，没有万能的方案，只有更合适。现在作者本人使用该方案已经开发了2款 app和小游戏，稳定上线，主要是为了使app快速开发和后期稳定维护。本文档只要是为了快速熟悉大概的东西，只能算是起步的，想深入了解，请看具体的详细教程。

## 内置和功能

内置了Xlua 2.12版本，DOTween , BestHttp 插件。

包含了简单mvc模式管理主ui (UGUI)，资源管理策略，热更方案，文件下载，任务管理等还有一些简便的操作方式。

## 目录结构



1.0

Pluguins : unity 放置各个平台 插件包的地方

Res: xyframework 放置待用的 美术和策划 资源的主目录地方

Resources: unity 放置app内置资源的地方

Scene: 放置场景文件的地方

Scripts: 放置脚本的地方(主要是C#脚本)

ThirdPlugin: 放置插件和一些库文件

#### 目录解析

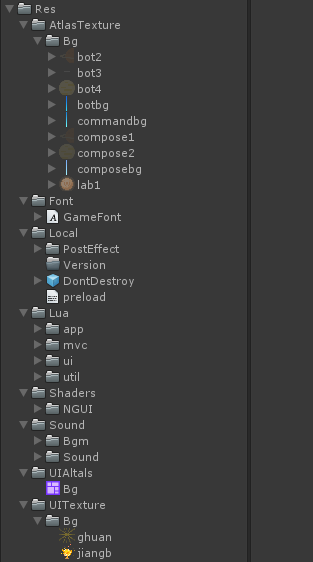
Res结构: 

图1.1

AtlasTexture: 存放小图片(可制作图集的图片)图片格式(不是指真实资源后缀，指的的是在unity图片设置界面设置的)为Sprite（图1.2），对应打包的图集在UIAltals 文件夹中,一个文件夹对应一个spritealtas，具体看framework中的spritealtas设置。

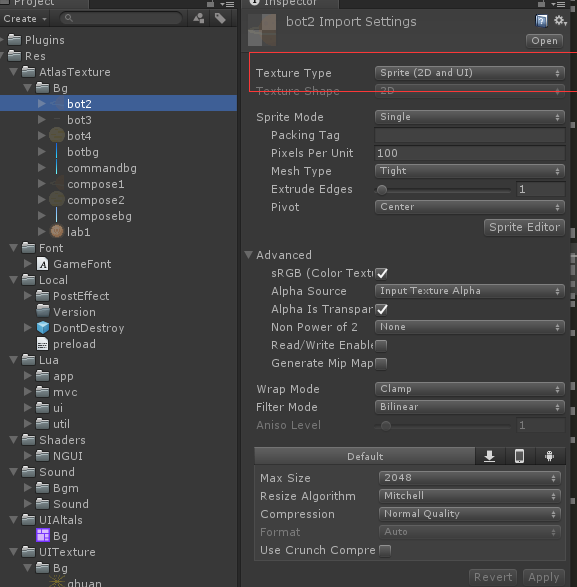


图1.2

Font: 一目了然就可以看出是存放字体的文件夹，现在framework中使用的字体就是现在GameFont这个字体，可以根据自己需求更换。

Local: 主要是存放一些只想被依赖的信息文件和文件的版本信息(热更相关后面会说)等等。如：DontDestroy这个prefab只有在打包app的时候被Game场景依赖跟随了app。这里这个prefab就无法进行资源更新了，可以根据自己需求是否参与热更(后面会说)。

Lua: 就是存放xlua 脚本的地方了。app 主要存放一些app层面相关的lua脚本，mvc存放了mvc的base脚本，ui就是ui相关的脚本，uitl只要是存放一些工具相关，后面还会生成data文件夹存放excel导出的数据。另外还可以自己添加一些需求相关的东西自己新建文件夹，比如游戏AI相关等

Shaders: 存放shader的文件夹，这里没有用到shader，就随意copy了一份NGUI的shader

Sound: 存放音效和BGM 等声音文件

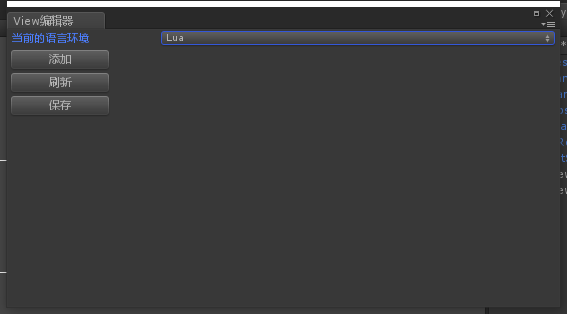
UIAltals: 刚刚已经说了存放 AtlasTexture 中的打包成的 spritealtals

UITexture: 存放图片较大的图片

#### 功能介绍

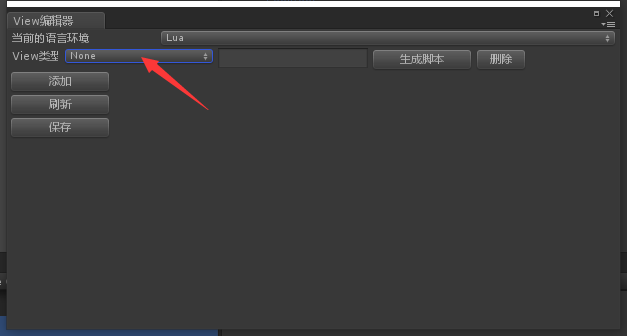
1. View编辑器

XYFramework->View编辑器

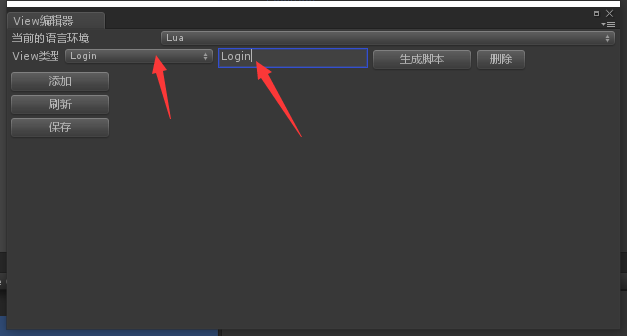


主要是为了方便快捷的建立一个简单的View,语言现在只支持Lua 后期考虑添加C#，因为使用的xlua这种方式，暂时还没考虑ILRuntime。

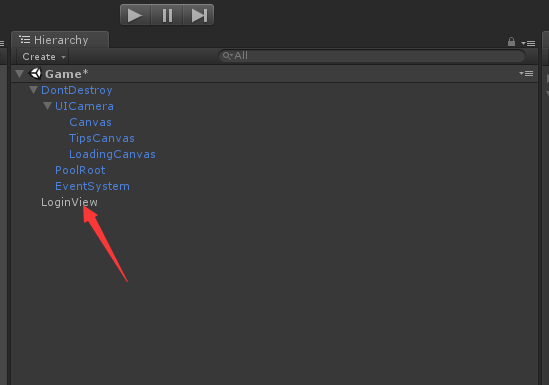
找到EnumScript.cs脚本 UIID 枚举，这里可以随意添加更改，对应



在空处填写你要添加的View名字

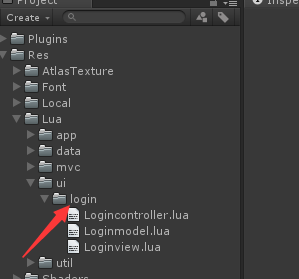


点击生成脚本

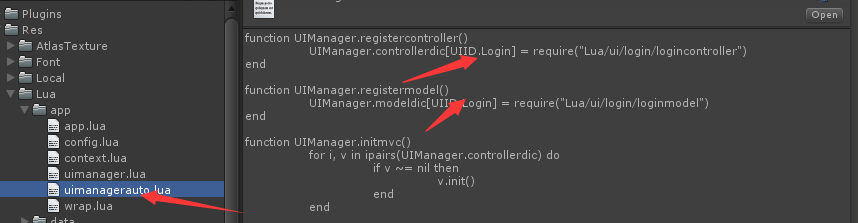
场景就会生成对应的view的Gameobject

面板信息如下

对应的lua文件所在，会帮你建好基本的mvc模式的lua脚本。



并会自动帮写写入



主要是为了注册Controller和Model,作用后面详细教程会介绍

这里是很简单的mvc基本结构当然也能存在C->N个view的情况这里可以根据自己的情况自己扩展比如:Loginview 有 Itemview，bulabulaview等等 这里只须在LoginController注册下相应的view就好了

Controller 里面主要是逻辑处理(数据和页面直接交互)

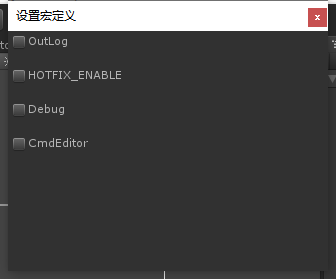
Model 里面主要是数据记录(数据表数据以及网络数据)

View 里面主要是承接C#层Xluamonobehavior脚本中的脚本和页面相关组件并自动生成

为了后期的维护和代码更新迭代可以让整个团队遵循这个结构，虽然代码写的是多些，但是整体效果及后期维护更或者人员交接也能快速接手都能很快的定位问题和转接。经验之谈，望采纳。

1. 设置宏定义工具

XYFramework->设置宏定义工具



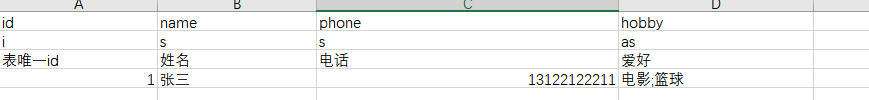
主要是为了在开发过程中切换宏定义，和出包完全无关，只是为了开发方便，对应Assets/Define.txt，想要加什么宏可以自行在这个文本里面添加。

1. Excel 导出lua脚本或者json数据

XYFramework->ExcelToData->生成Lua脚本

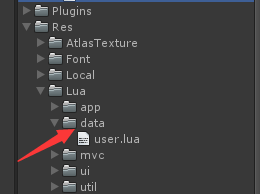
为了使Excel生成对应的lua脚本 Excel文件夹在 ../Excel/ 中

例子 User表结构

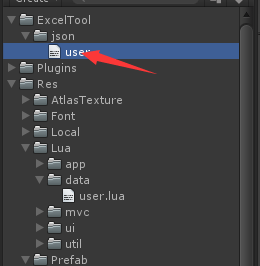


第一行 数据变量 第二行 数据类型(可支持很多种,因为是lua或者json,基本原则就是 i为number 类型，s 为string类型，前面加a 如：ai，as就是相对应的数组数据，分隔符为分号;) 第三行往后就是真实的数据结构

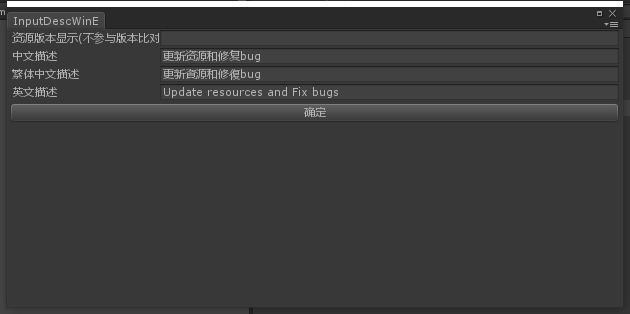
对应生成的lua文件目录



对应生成的json文件在，因为暂时没有用到json数据暂时放这了，想改路径可以随时更改在代码中ExcelExportDataEditor.cs



1. AssetBundle 相关,这里只说用法具体看详细教程



版本号:这东西时候给用户看的一个文字信息而已不要在意 不参与资源文件任何判断，

3语言描述：这个会被打包到资源版本信息文件里面也是展示给用户用的

1. AssetBundle->打包-> (三种打包)

1. 差异打包资源(主要就是为了比对变化的资源进行打包，没有的资源进行删除处理)
2. 强制打包，顾名思义就是强制打包不关心资源
3. 打包lua脚本，就是只是打包脚本文件

产生的资源在在../ ABFiles/ 文件夹中

2.序列化prefab

就是为了拆分prefab的资源依赖，自己建立依赖，使用自己的资源管理(原理是引用计数)，现在只支持Image，RawImage，SButtonUGI，Animator对应的资源数据，后期考虑 如Mesh，Audioclip等

对应的目录：Resources/ Prefab/

产生的目录：Res/ Prefab/

3.复制到服务器资源文件夹

文件夹所在目录为../../ ServerResData/ 中，这里是服务器所需要的分离出来的资源，使用了git来进行管理(当然也可以用svn看自己熟练度)，无视版本号等麻烦操作，直接使用git操作服务器需要热更的资源和脚本

1. 出包

Win Android IOS

分别导出 win平台，Android 平台 IOS平台相应的工程或者app包

具体看详细教程

1. 资源和lua脚本热更。

步骤4中已经说了一些AB文件的东西,其中每次打包资源时候都会产生版本信息，

基本原理：增改删，就是net资源信息同app内部和app缓存 资源信息同时比对，产生比对列表，然后按需下载。

具体脚本在VersionManager.cs中

1. 文件下载对应脚本 DownloadManager.cs 是基于C#Besthttp来做的，主要是因为bestHttp比较成熟的C#下载工具，bug基本没有，功能强大自带异步不会影响unity主线程。

总结：本方案比较简单，上手只要稍微有些lua和c#基础就能看懂。

缺点:因为是从项目中分离出来，代码前期看起来可能有点乱，后期会更新。细致功能点涵盖比较少，没有强链接支持，没有特定交互协议，在网络支持上较弱，后期会加强网络部分功能(考虑protobuff 无GC版 同时支持lua版本，帧同步的一些基本操作等等)，还有后期资源更新可能还会考虑配合Jenkins做成自动化。可能还会尝试一些AI的东西希望能用ECS来做。

优点：快速上手，做App和弱连接的小游戏足够，基本大功能点都有。自动化的东西比较多。