ET

deepwaterooo

February 3, 2023

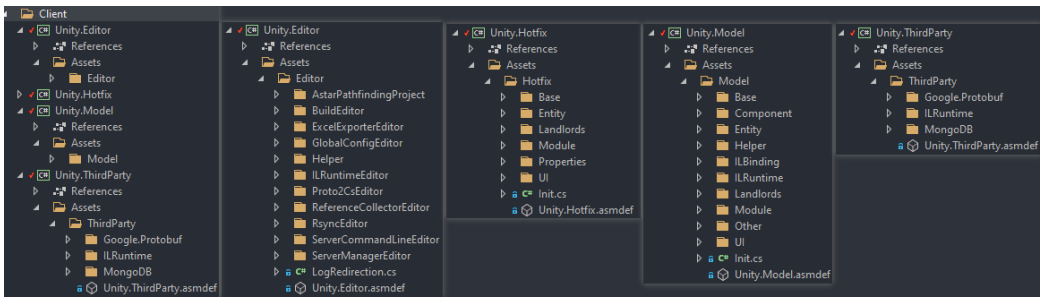
Contents

1 ET 框架框架设计模块功能等

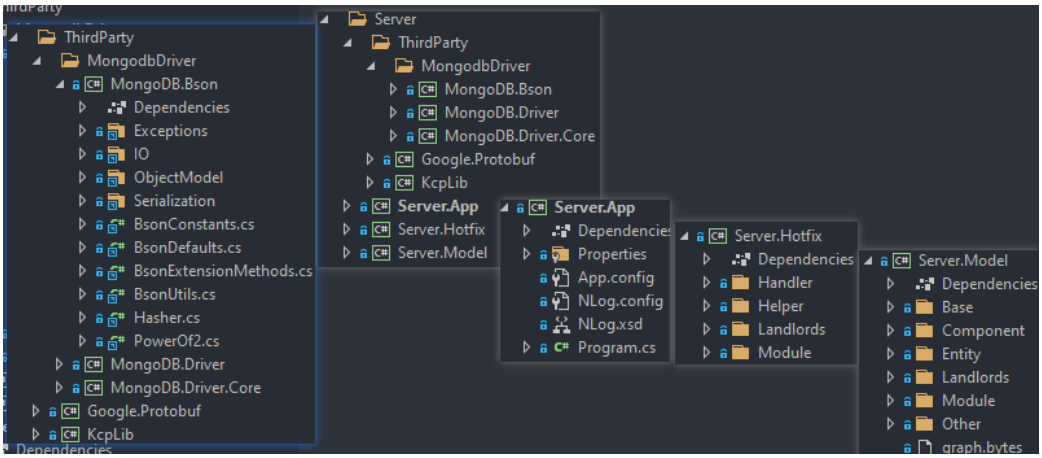
1.1 一个双端框架的 12 个项目，目录结构

- 就不知道这 12 个项目是如何组织构建起来的

1.1.1 客户端的四个项目



1.1.2 服务器端的 8 个项目：



- ET-EUI 整理出了一个小小的登录系统，如果这个系统能够运行得通，配置简单顺利，也就基本上也算是达到自己小小服务器的需要了。运行一下试试看。

1.2 模块功能管理

1.2.1 ET 框架中事件使用的规则和注意事项

- **事件结构体的定义**必须在 Model 或 ModelView 下，**事件的订阅和发布**必须在 Hotfix 或 HotfixView 下（是否为 View 层根据是否需要 UnityAPI 决定）
- 在 ET 框架中 **View 层可以直接调用逻辑层的代码**，而 **逻辑层不允许直接调用 View 层的代码**，所以逻辑层想要和 View 层交互只能使用抛出事件的方式，让 View 层进行订阅进行相应的处理。

1.2.2 ET 框架下 ETTask 的异步编程

- 开发早期都是使用协程或者多线程进行程序异步的实现，
- 在 C#5.0 之后引入了 Tasync await 等关键字可以轻松优美的实现程序的异步进行。
- Task 是 C# 异步编程库的异步功能基础类型，包含了多线程的使用。
- 而 ET 框架主打的是 **多进程单线程**，所以使用 ETTask 对 Task 进行了封装，使其不用考虑多线程的共享难题，更易于使用。

1.2.3 协程

- 协程其实就是创建一段程序辅助主线程的运行，注意 **协程不是多线程，其仍运行在主线程当中，其只是将一个函数按照一定的条件分解成若干块，穿插在主线程的各个位置中运行**。
- async 和 await 关键字
  - async 是修饰函数的关键字，被修饰的函数称为异步函数，只有被此关键字修饰的函数才能使用 await 关键字
  - await 关键字后跟一些表达式 (一般是返回值为 ETTask 的函数)，在异步函数运行到此时会立即跳出函数，继续执行原逻辑。
  - await 返回前会做出保证，保证 await 后表达式的任务完成后，执行点会跳回异步函数中，继续向后执行剩余代码
  - 若 await 后表达式正常返回，可用变量接收表达式的返回值，表达式的返回值类型为定义表达式 ETTask<> 的泛型

2 小小服务器：要怎么才能开始动手试图去实现这个小服务器呢？

2.1 如果适配 ET 框架，现游戏可能哪些模块版块存在问题

- 我也觉得 ET 框架可能不太适合我现在的游戏（也就是说，把我的游戏完全适配成 ET 框架来开发，原本只需要一个小小服务器，完全适配为 ET 框架，就把问题弄得狠复杂了。。。），
- 使用 ET 框架，我所有的安卓基础就会被抛到九宵去外，不再关安卓 SDK NDK 什么事儿了。。。。。是对自己太大的损耗。而我原本还可以简单封装实现的安卓录屏，游戏内使用安卓 SDK 相关功能模块录屏游戏过程等，会被全部废掉，损失太大不值得。我觉得我就只要个文件服务器加个数据库而已。
- 原因是：我现在还想不通若是用 ET 框架来实现自己游戏的（服务器与客户端双端都可以热更新），我该如何实现我的方块砖 10 个按钮上的点击事件，射线检测？它的 ILRuntime 热更新程序域里对射线检测包的组件安装可能会成为自己狠大的问题，因为还不是狠懂里面的内部原理. 这个模块重构的原理是: 把射线检测, 如果必要一定要, 封装成如 ET 中任何其它可装载卸载的组件一样的装载到某个必要场景上去.
  - ET 里有个检测鼠标左右键位置的帮助类, 但跟射线检测应该还是相差狠远的. 而游戏场景里面有一个 OperaCompoent，这个组件会实时监听按键的点击并且将点击的位置发送给服务器，服务器再传送回客户端
- 所以, 现在对这个框架, 最深的感触是: 盲人摸象, 摸每部分细节似乎都快清楚了, 却无法组装从一个更高的层面上来理解和把握框架设计, 无法吃透, 在大框架功能模块上犯难, 在网上再搜一搜
- 我可以把两组按钮画面同样做成预设与热更新资源包，射线检测同样会成为可装载可卸载的组件，可是再后面射线检测到的物体逻辑，感觉有点儿复杂了
-

2.2 如果不适配，怎么弄个服务器带数据库等逻辑？

- 使用部分共享源码的双端（共享的是文件服务器 8 个项目，MongoDB 数据库服务器，Realm 注册登录用，网关服，Location 服，ETTAsk，RPC 消息机制，NetComponent 等自己机对陌生需要练习，而自己的服务器也不可缺省的相关功能）
- 现在知道自己整的不沦不类的服务器所谓的登录，登录的是网页上的认证相关，跟自己真正要实现的游戏里注册登录服保存数据库完全两回事，现在知道了。
- 作用 ET 的头，实现用户注册与登录，适配自己现有游戏的尾，游戏除了入口之外全游戏进热更新程序域里
- 那么自己的现框架架构作尾，全游戏逻辑进热更新域，存在的问题就变成是：
- 我无法再实时动态检查用户上下线顶号之类的, 我只能默认登录过就是登录状态，可是用户下线了，或更严格的说掉线了，服务器并不及时知道，可以通过安卓 SDK 中的按钮登出知道。但是掉网了掉线了呢?（这部分的逻辑可以晚点儿再考虑，把网络请求相关的摸得再熟悉一点儿再弄）
- 再则，ILRuntime 热更新程序域里，我又该如何实现在热更新程序域里网络上载用户的游戏保存进展？这里需要去想和理解，为什么它 ET 框架就可以在热更新程序域里同网络交互，你哪里还没有想明白?
- ET 框架，热更新程序域里装载的组件，只是帮助与外界游戏程序域连通好，真正的网络请求上传下载等是在热更新域外面完成链接式传进去的？感觉对这个大框架没有掌握好，脑袋仍然是在像糊糊一样。。。

- ET 框架，网络的那部分做得还是比较完整的。实现在了各种的封装，涉及大量的网络调用与交互，游戏过程中的交互与更新。但是太多的功能对于自己的游戏来说完全不必要. 所以只想用 ET 的头
- 各种泛型，接口的定义，一二三个参数等的泛型接口定义 (你可以去找一找工程中的各种 ILRuntime 的适配器)，全都是都可以成为热更新域里能够被游戏程序域里识别的原因，所以狠多设计，自带 ILRuntime 的适配性质
- 那么就可以小一点儿一点儿地来, 先弄个登录窗口, 实现服务器的注册登录保存登录信息到数据库, 相对比较小点儿的逻辑. 这个过程中把 MongoDB 数据库的配置等所有连接过程中必要的步骤, 可能出现的问题给解决掉, 就算前进了一小步呀
- 不知道怎么开始，也不知道怎么创建可以🄵套的像是安卓模块库一样的子工程，就只能把小游戏斗地主复制一份了再从它的基础上来改?!!!
- 如果简单一点儿开始，我觉得我应该是可以先把简单点儿的 MongoDB 数据库连接成功，把用户登录相关的逻辑，网络交互的部分，ETTask RPC ACTOR 消息等，哪怕是复制，把这部分弄过去

2.3 ET 框架

- https://blog.csdn.net/qq_33574890/article/details/128244264?spm=1001.2101.3001.6650.1&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EAD_ESQUERY%7Eyljh-1-128244264-blog-123841252.pc_relevant_multi_platform_whitelistv4&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EAD_ESQUERY%7Eyljh-1-128244264-blog-123841252.pc_relevant_multi_platform_whitelistv4&utm_relevant_index=2 上次看看得不是狠懂，这次再看，至少是觉得 UI 的逻辑处理，作者的观点更自然真实一些，放在一个文件一起处理，个人认为更好, 而不是折分成为几个文件

```
class LoginState:State{
^^Ivoid OnEnter(){
^^I^^IUI.Show()
^^I}
^^Ivoid OnLeave(){
^^I^^IUI.Hide()
^^I}
}
```

3 登录协议流程

- 因为登录协议是客户端与服务器通信的，不属于服务器内部协议，所以打开 OuterMessage.proto，里面存放的都是客户端与服务器通信定义的协议数据。
- 比如定义如下，登录协议：

```
message C2G_LoginGate // IRequest
{
^^Iint32 RpcId = 90;
^^Iint64 Key = 1;^^I// 帐号
}

message G2C_LoginGate // IResponse
{
^^Iint32 RpcId = 90;
^^Iint32 Error = 91;
^^Istring Message = 92;
^^Iint64 PlayerId = 1;
}
```

- emacs 里 org-mode exporte-to-pdf 希望有个 latex 选择可以自动将^I 转化为空格，而不是这种字符，晚点儿再弄这个

- 注意点：**没有意识到像是注释一样的片段，这个协议里，会成为标注或是标签**
  - 1. 因为登录是请求-响应类型协议（即发送一条数据，并期望返回一条数据），所以注意对应 C2R_Login 协议带有 “//ResponseType R2C_Login” 标志，在生成协议时，用于标记这个 C2R_Login 请求对应的响应类型为 R2C_Login
  - 2. 因为请求是直接发送给 realm 服的，所以是普通的 IRequest 类型协议，标记为 IRequest
  - 3.R2C_Login 回复类消息结构，因为是 Realm 服发送给客户端的，因此是一个普通 IResponse
  - 4. 注意两个协议类里面都有 RpcId，主要用于发送请求-响应类消息时，发送将自己的 RpcID 发送出去，返回时带回这个值，用于发送方接受到返回协议时，可以找到对应的是哪一个请求协议返回来的。

4 一步一步的进展

- 首先，把斗地方大厅改写为游戏主菜单的三个选项 (如果我只想用 ET 的头，它的头太大了，还是要自己弄个小小的头，小小的服务器，所以暂时就还是考虑自己从头实现一个 MongoDB 的小小服务器比较容易一点儿，不懂的就翻 ET)

- 把这个界面的相关上下文全部适配好：UI 的自动创建生成系统，UI 的按钮点击回调等
- 这里想要找的是：在点击的回调里如何，是否可以卸载装载 UI 组件，还是说必须得去 HotfixView 什么视图层来处理这些逻辑呢?

4.1 UIType.cs: 这种类型的定义好像不止加一个地方，一个地方不够，可是大的框架架构还是没搞明白

```
namespace ETHotfix {

    public static partial class UIType {
        public const string Root = "Root";
        public const string UILogin = "UILogin"; // 注册 登录 界面
        public const string UILobby = "UILobby"; // 主菜单  三选项

// 上面的界面远远不够呀。。。
        public const string UIEducationalMode = "UIEducationalMode";

        public const string UIEducational = "UIEducational";
 // 怎么再把它细化为：三  四  五方格呢？应该是要用同一接口的不同实现，完全重复写三个系统会把人弄死的。。。。
        public const string UIGridThree = "UIGridThree";
        public const string UIGridFour = "UIGridFour";
        public const string UIGridFive = "UIGridFive";

        // 那么就涉及游戏界面的折分：哪些是可以公用，哪些是不得不细化最小粒度的？

        public const string UIClassic = "UIClassic";

        public const string UIChallenge = "UIChallenge";
 // 挑战难度：要定义接口来实现 20-50 个不同的实现了？
    }
}
```

- 安卓 SDK 这个框架其实并不受影响。但本质是所有安卓 SDK 的东西不能够热更新。因为 ET 是网络多人游戏框架的，可能更多的是不适合添加与适配案桌 SDK。这些晚点儿再得结论好了，反正我的案桌 SDK 本质也是可要可不要。如果能够快速掌握一个比较好的双端框架的话
- 不知道若是照这么改下去，得把这个游戏改成是什么花葫芦呢?

5 带 MongoDB 数据库的注册登录用户帐户管理资源文件服务器

- 去找和实现简单的服务器项目，操纵 MongoDB 数据库
- 除了自己的电脑安装有 MongoDB 数据库之外，服务器项目中因为要连接操纵电脑上数据库，可能还需要狠多插件的安装与配置，连接字符串，什么 MongoDBClient 之类的。这样的小项目狠容易就实现了，基本没有任何挑战
- 关于小小 **文件服务器热更新资源包**：
  - ET 里 **客户端的资源包更新逻辑**狠明白了：是 **下载服务器端的热更新资源 MD5 码表文件到客户端；客户端删除所有服务器中不存在的资源包；再一一比对存在于服务器端的资源文件，合成了一个需要下载更新的热更新资源链表；然后异步下载这些客户端落后于服务器的资源包们**。
  - **服务器端的逻辑**：还没有追踪，需要理明白。
    - **文件服务器，热更新资源包**，是如何保存到服务器: ET 是一个双端框架，它服务器里的资源包，也就是存放在服务器的某个目录位置。它是通过 Unity 里的打包工具一键打包保存在服务器的某个特定目录下，并自动生成服务器端各资源包资源文件的 MD4 版本号码表文件 Version.txt。
      - 这里还有点儿糊糊：**双端框架**是说，服务器上也需要存放客户端的所有逻辑，比如客户端从 unity 中一键打包，保在某个特定的目录，用作服务器端的热更新资源包小小文件服务器？如果这样，下面两条就不需要了
      - 所以这里的逻辑是否有点儿仓畜，不过不是重点。如果是自己的服务器，仍得上传呀，我不能云上服务器云上来项目。这个逻辑很容易实现。
      - 同样对应的，如果实现服务器资源包的过滤更新等，小细节都可以自己去实现

* 这里没弄明白的是：**FileServer.exe 是如何生成的？生成逻辑，相关原理在哪里？**为什么示例项目中这个进程必须得运行起来？运行起来，它有个地址，才方便客户端 MD5 码表比对与下载更新必要的客户资源包？
* **服务器又是如何处理下载逻辑的？**它是存放在服务器上的某个目录，它是用 Unity-WebRequestAsync 将目录中特定的文件下载到本地的
* 下面的，因为服务器是简单地将资源包存放在某个目录下，小文件服务器没有涉及任何的不涉及数据库存储，所以逻辑狠简单。下面的用不上
  - MongoDB 数据库是可以下载文件的，看见一个网络上的小例子：可以成为自己的一个参考：https://blog.csdn.net/y526089989/article/details/94452268?spm=1001.2101.3001.6650.15&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-15-94452268-blog-102048135.pc_relevant_3mothn_strategy_and_data_recovery&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7EBlogCommendFromBaidu%7ERate-15-94452268-blog-102048135.pc_relevant_3mothn_strategy_and_data_recovery&utm_relevant_index=16

```
using MongoDB.Bson;
using MongoDB.Driver;
using MongoDB.Driver.GridFS;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ESRTools
{
    public class MongoHelper
    {
        private static GridFSBucket bucket;

        ///新版本方式 旧的不一样
        public static IMongoDatabase GetDatabase()
        {
            //mongodb://user:pwd@127.0.0.1:2222/db
            string connectionString = "mongodb://user:pwd@127.0.0.1:2222";
            string dbName = "dbName";
            // or use a connection string
            var client = new MongoClient(connectionString);
            // or, to connect to a replica set, with auto-discovery of the primary, supply a seed list of
            // var client = new MongoClient("mongodb://localhost:27017,localhost:27018,localhost:27019");
            return client.GetDatabase(dbName);
        }
        public static void DownFile()
        {
            bucket = new GridFSBucket(GetDatabase()); // <<<<<<<<<<<<<<<<<<<<
            var fileid = new ObjectId("");
            bucket.DownloadAsBytes(fileid);
        }
        public static  ObjectId GetUploadFileId(string fileName)
        {
            bucket = new GridFSBucket(GetDatabase());
            var filter = Builders<GridFSFileInfo>.Filter.Eq(x => x.Filename, fileName);
            //eq 方法，就是等于，还有其他的方法，具体看 Mongo 的 api 文档
            var sort = Builders<GridFSFileInfo>.Sort.Descending(x => x.UploadDateTime);
            //按上传时间来倒叙一下
            var options = new GridFSFindOptions
            {
                Limit = 1,
                Sort = sort
            };
            using (var cursor = bucket.Find(filter, options))
            {
                var fileInfo = cursor.ToList().FirstOrDefault();
                if (fileInfo != null && fileInfo.Length > 0)
                {
                    return fileInfo.Id;
                }
                return new ObjectId();
            }
        }
}
```

- **protobuf 进程间消息传递协议:** 加进来。Protocol Buffers 是一种轻便高效的结构化数据存储格式，可以用于结构化数据串行化，或者说序列化。它很适合做数据存储或 RPC 数据交换格式。可用于通讯协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据格式。
- TODO：如何把 ET 的头，关于 ETTask 异步包装的网络请求，关于新用户注册并保存用户信息进 MongoDB 数据库，用户登录，用户信息提取等，以最精简的方式整合到这里面来作为自己的小小服务器
- 感觉上面这些都基本上算是抓清楚了，少数一两个问题（*FileServer.exe 是如何生成的？生成逻辑，相关原理在哪里？*）慢慢解决，下午晚上会试着实现注册登录帐户系统的移植与整合