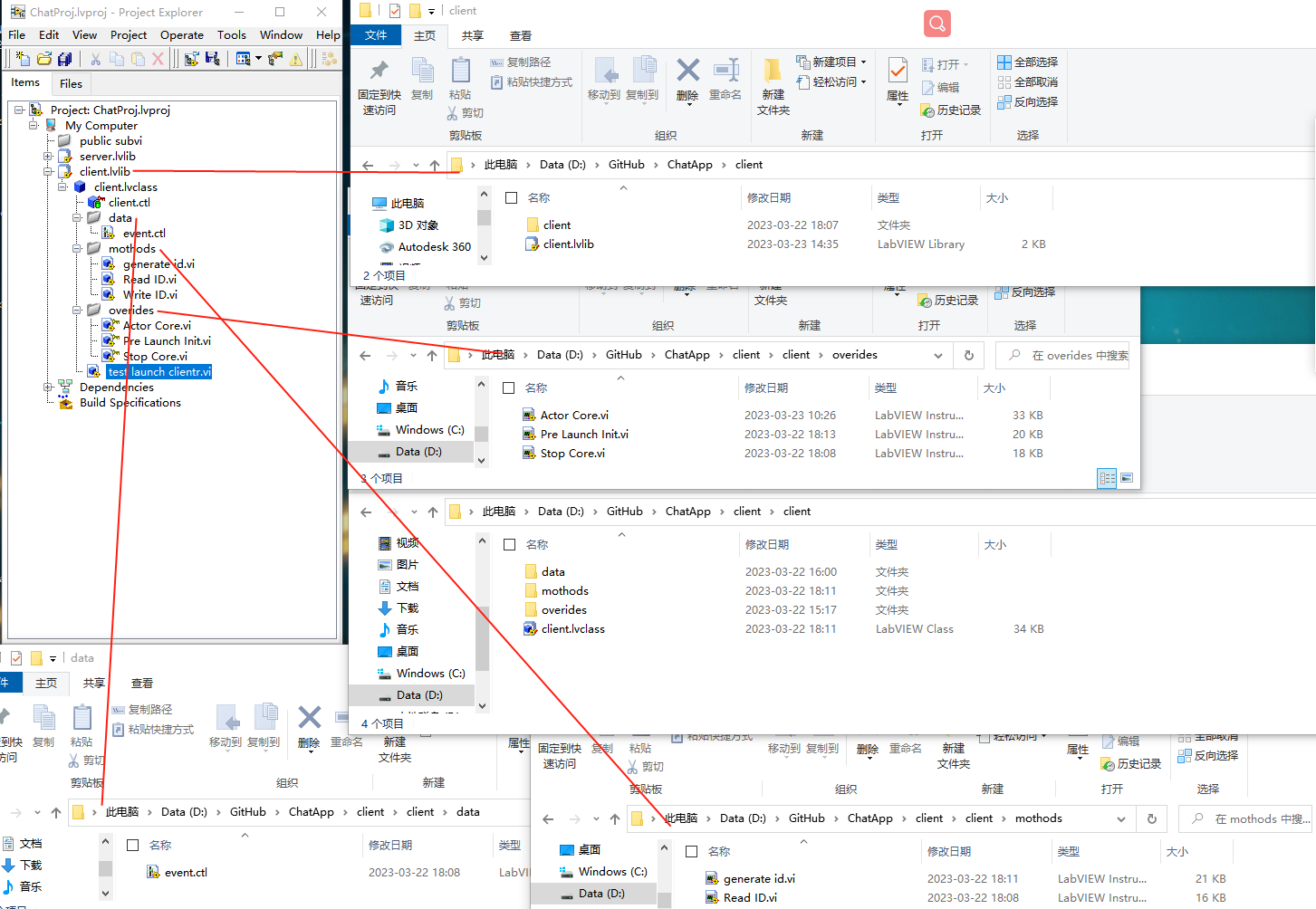
#### LabVIEW Actor Framwork (2)\_\_\_\_\_\_\_\_ 边学边做server&client

回顾下初始需求：

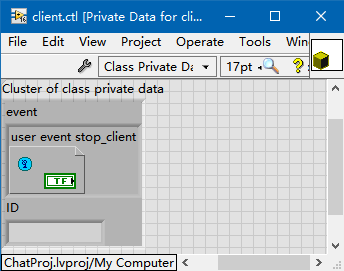
现在要做一个类似聊天的demo，一个server端，若干个client端；首先是server启动，通过server可以打开若干个client端，然后每个client可以独立给server发送数据。server收到数据打印出来带有client id的信息。

 上节我们已经完成了server 这个actor的开发，并完成了单元测试；

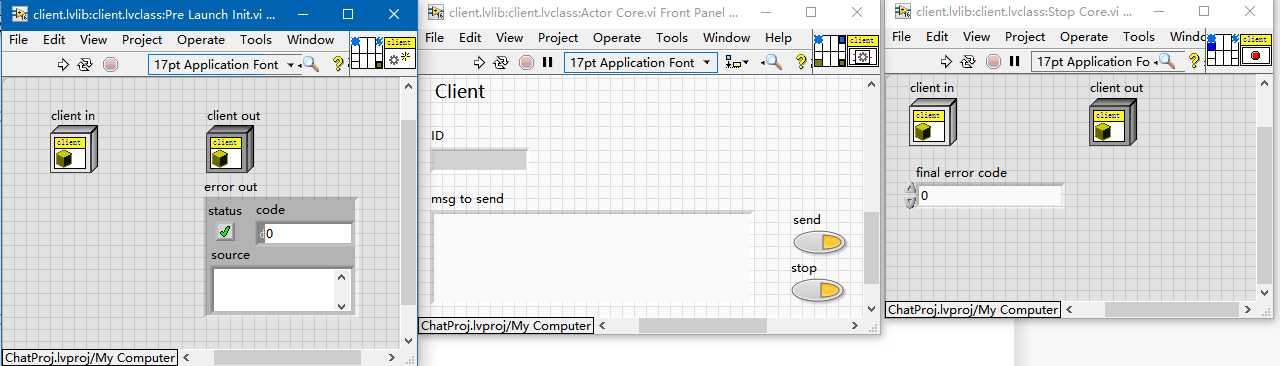
接下来我们就要编写client这个actor了；有了server的经验，这个依葫芦画瓢就行。Client类的结构与在硬盘中的存储位置，项目中的虚拟文件夹映射关系：



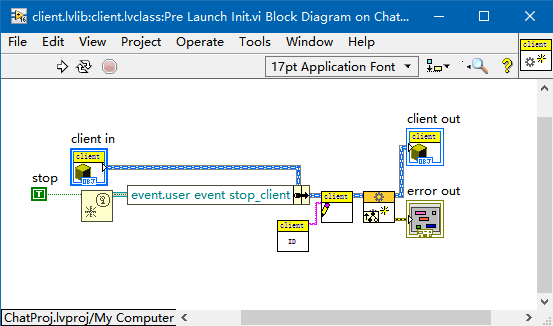
Client的私有数据包含2个，1是stop停止事件，1个是id，这个是为了后面区分不同的client的id；



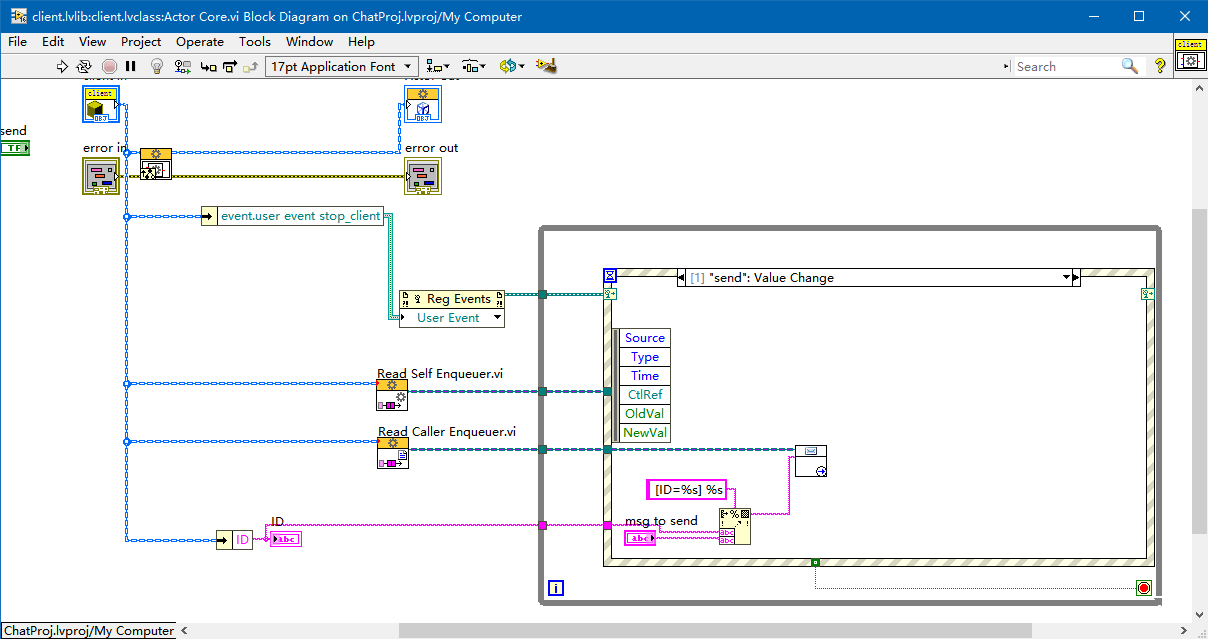
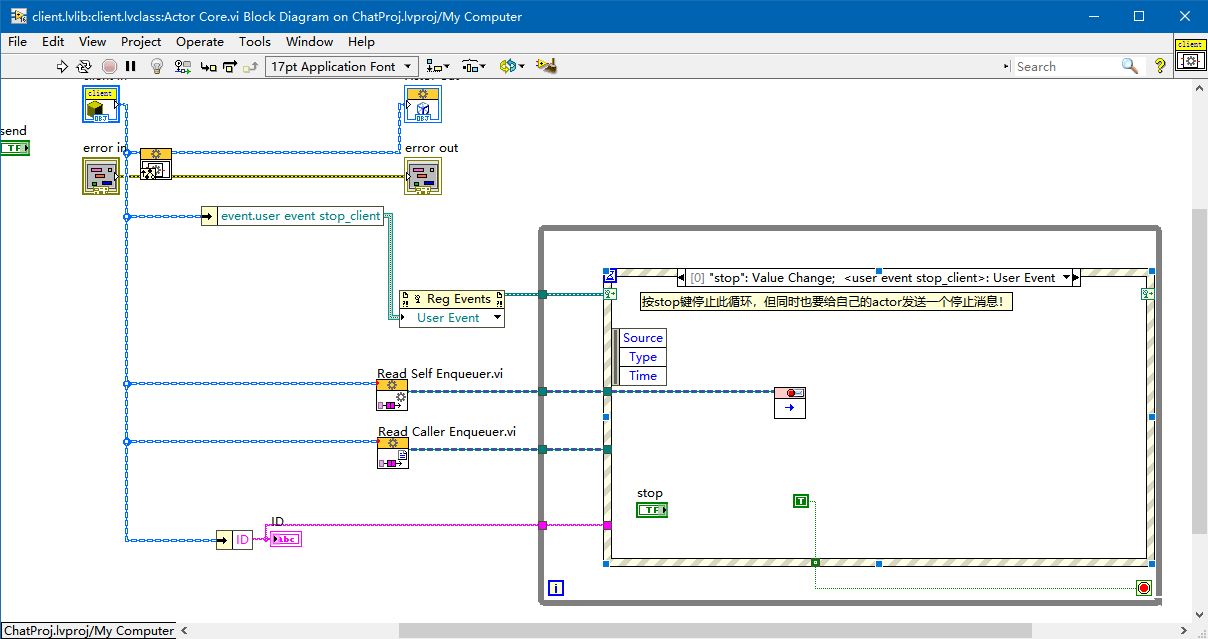
重写的3个vi：



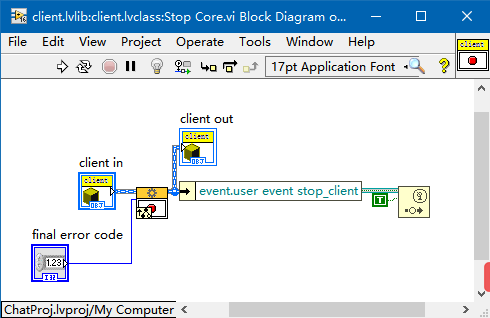
client.lvlib:client.lvclass:Pre Launch Init.vi程序框图：



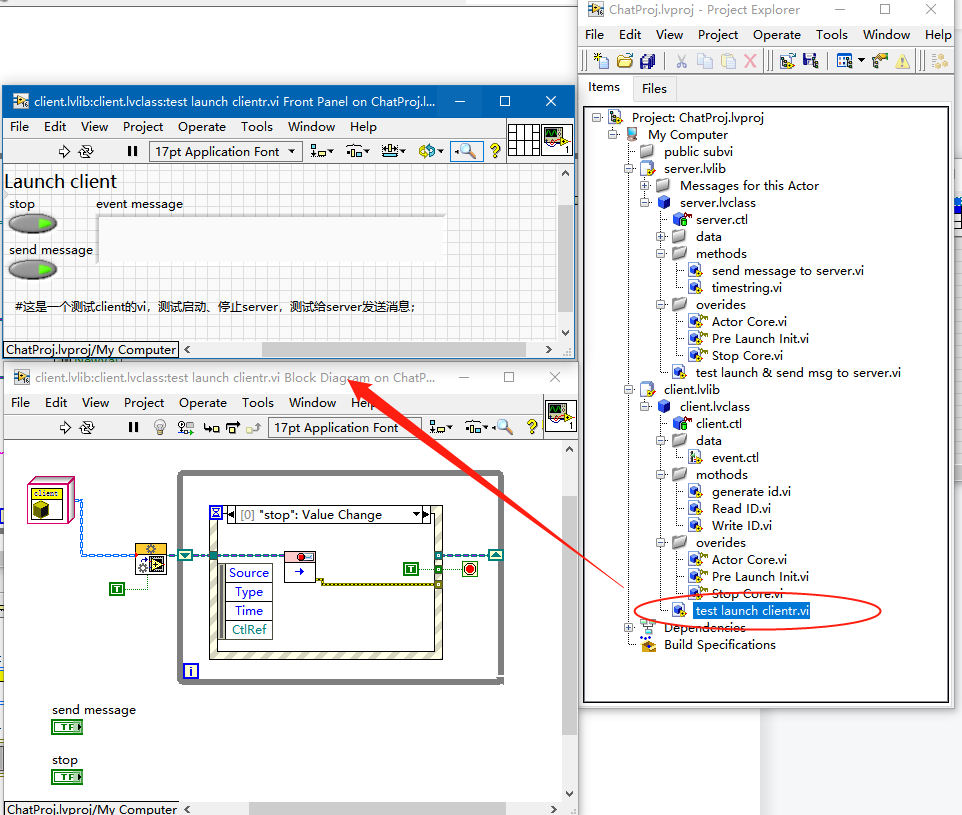
client.lvlib:client.lvclass:Actor Core.vi程序框图：



client.lvlib:client.lvclass:Stop Core.vi程序框图：



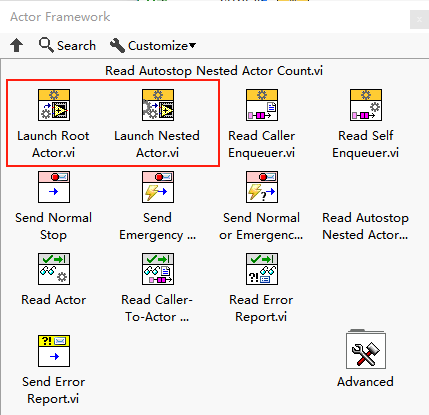
同样的，对于cilent也要进行单元测试，测试vi：



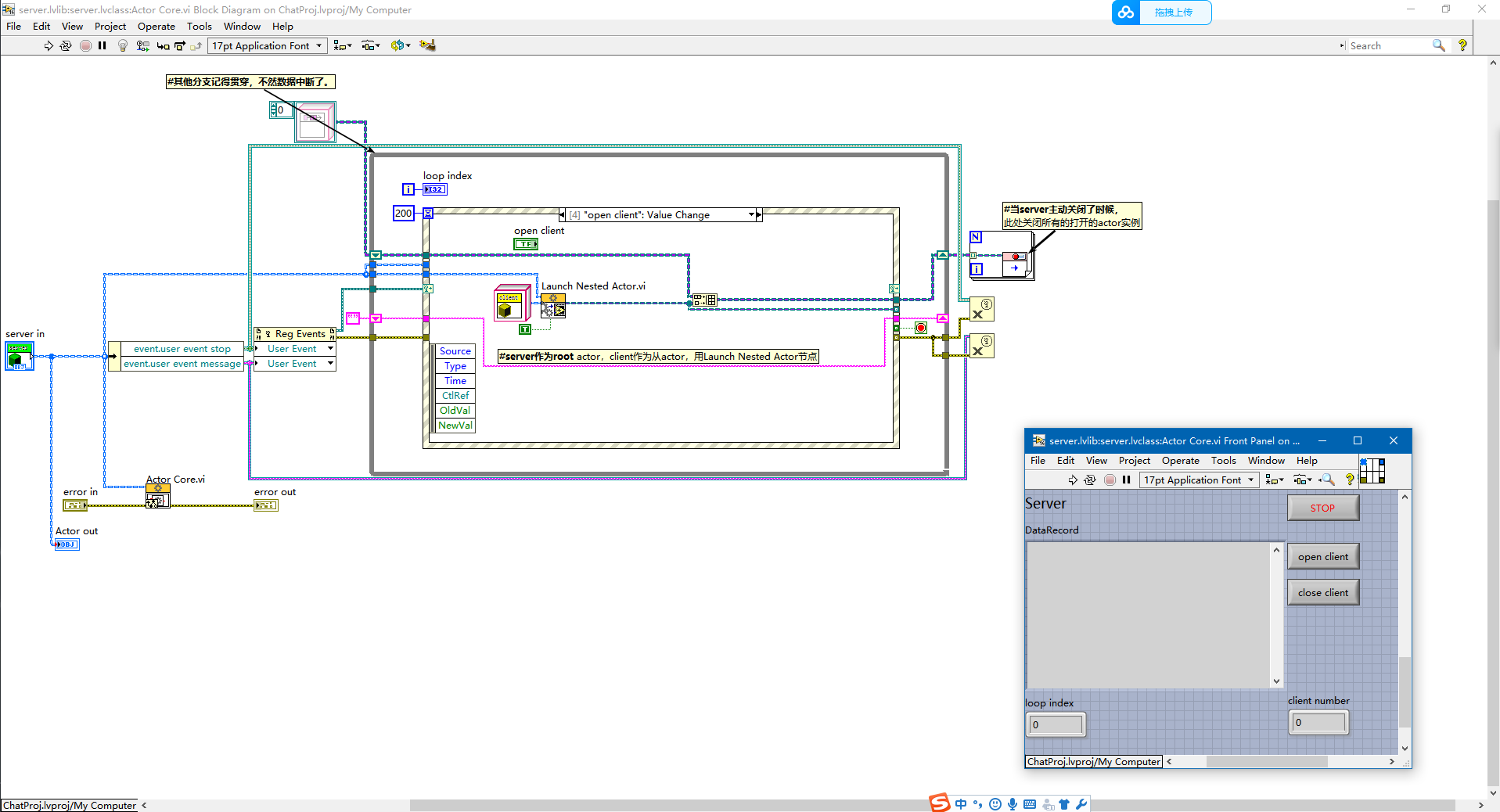
测试client的2个功能：1是启动，2是停止；

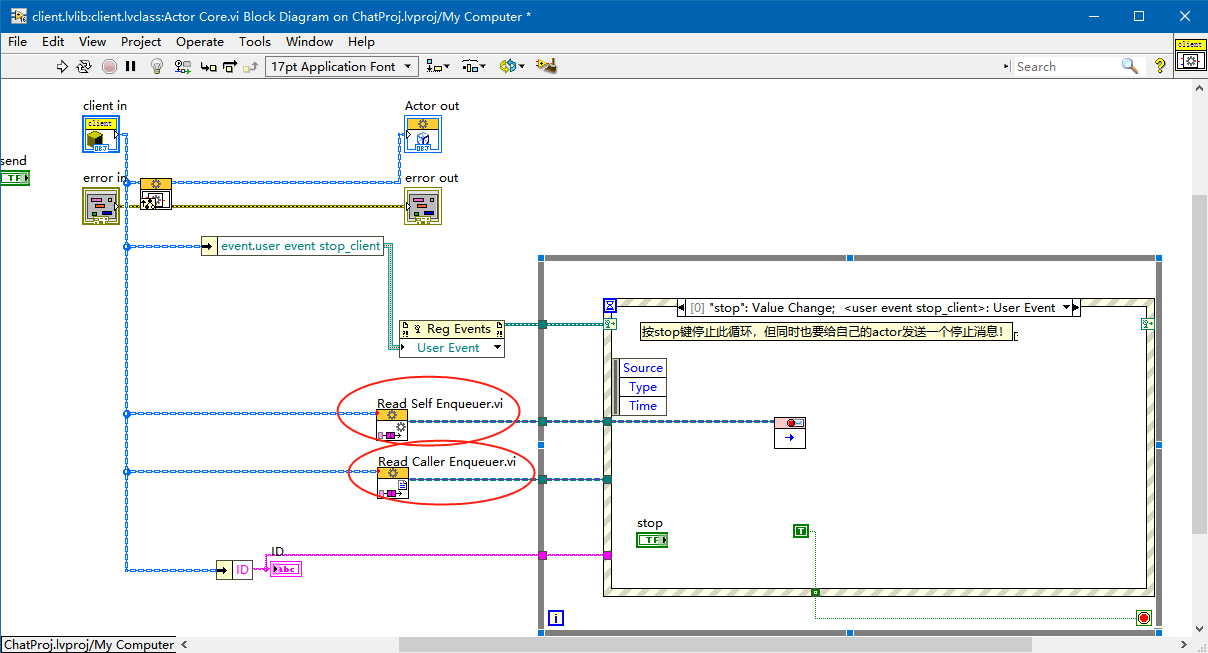
接下来第三个功能，如何让client给server发送数据呢？

这里得首先了解Root 和 Nested 两个概念。

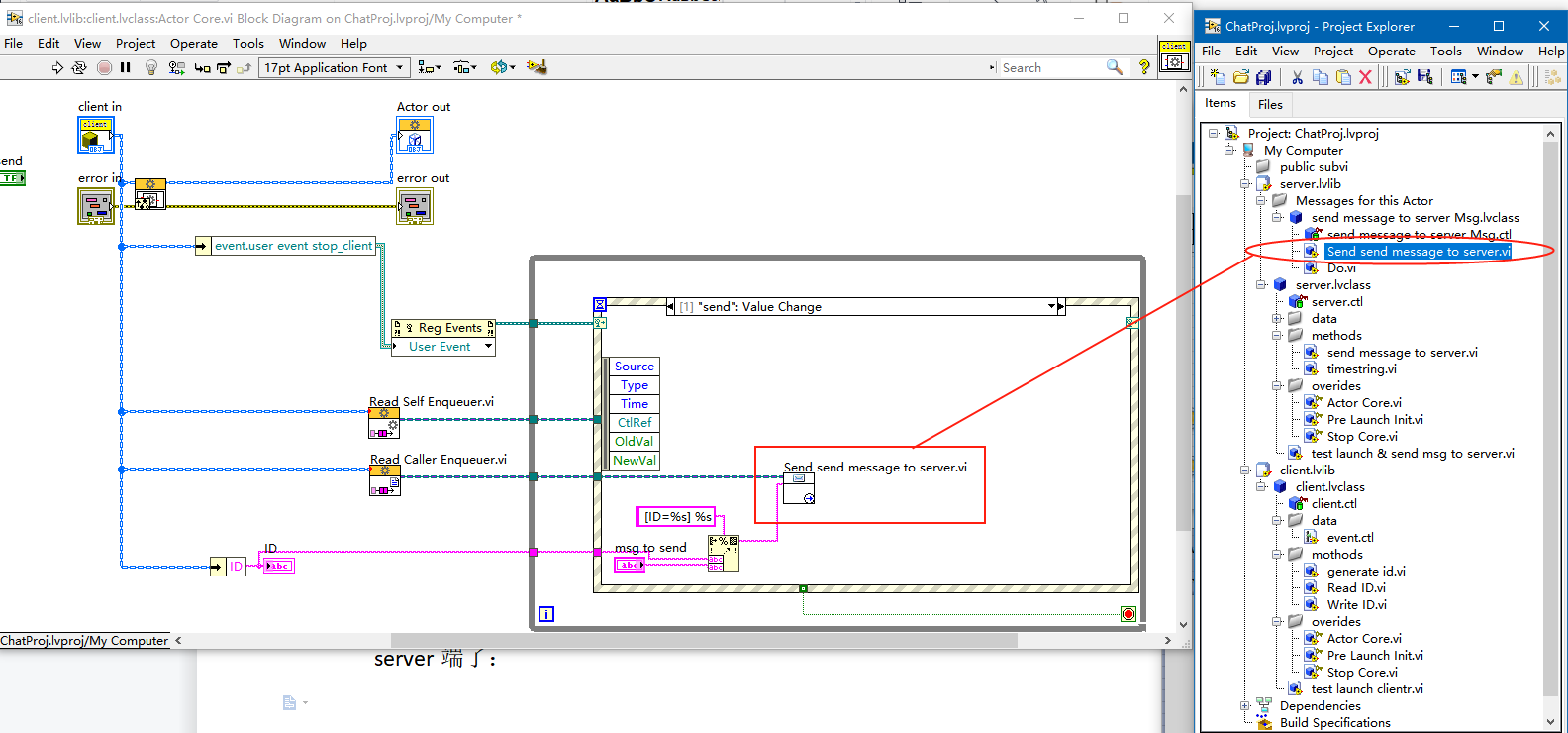


上节我们启动server actor是用的就是launch root actor这个函数，刚单元测试client也是。现在我们需要去server的actor core.vi里面去增加2个按钮的功能，打开和关闭client。注意此处，open一个新的client actor时用的便是Launch Nested Acotor这个节点，此节点区别在于用一个actor去启动另一个actor，这样两个actor之间便形成了caller（调用与被调用的关系）。

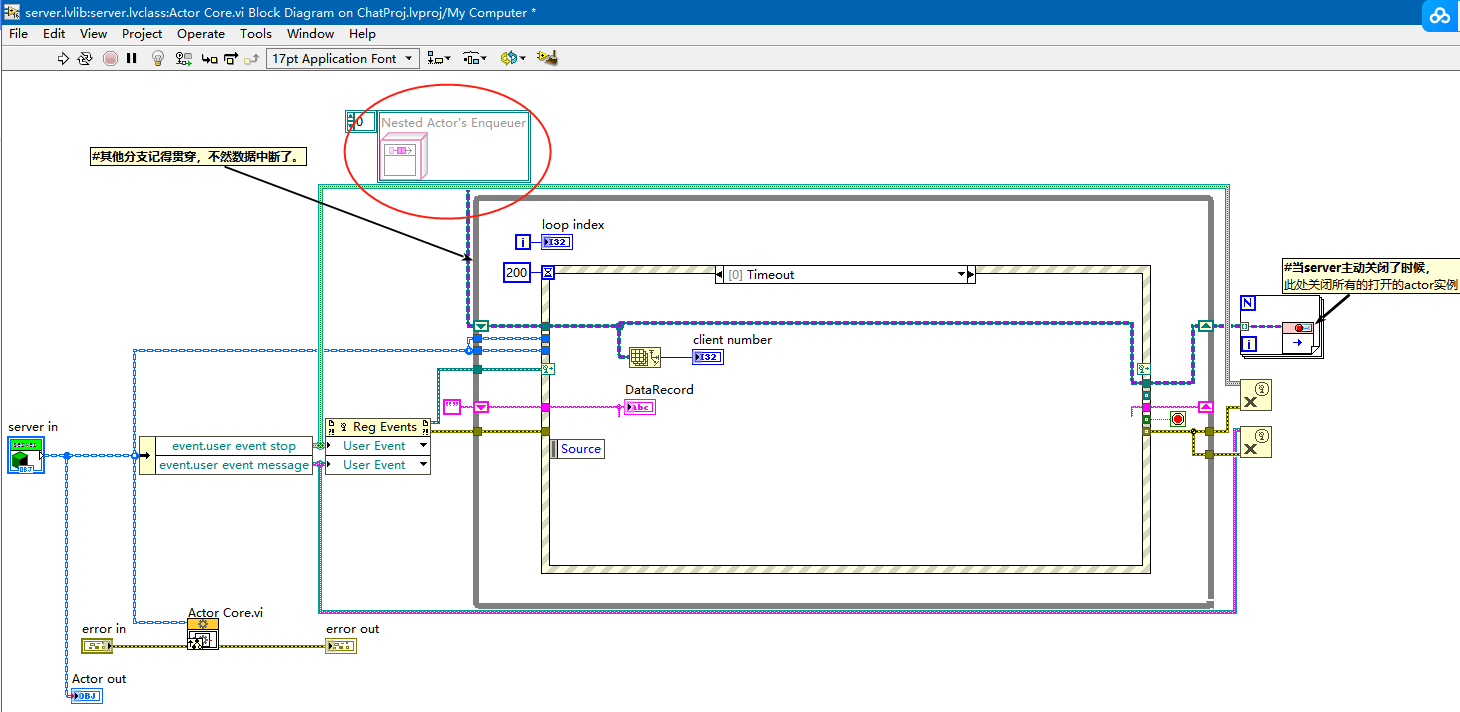


此时，回到client的actor core.vi看到通过下图红色两个节点，一个获取到自己的消息队列，一个可以获取到root actor此处便是server的消息队列。

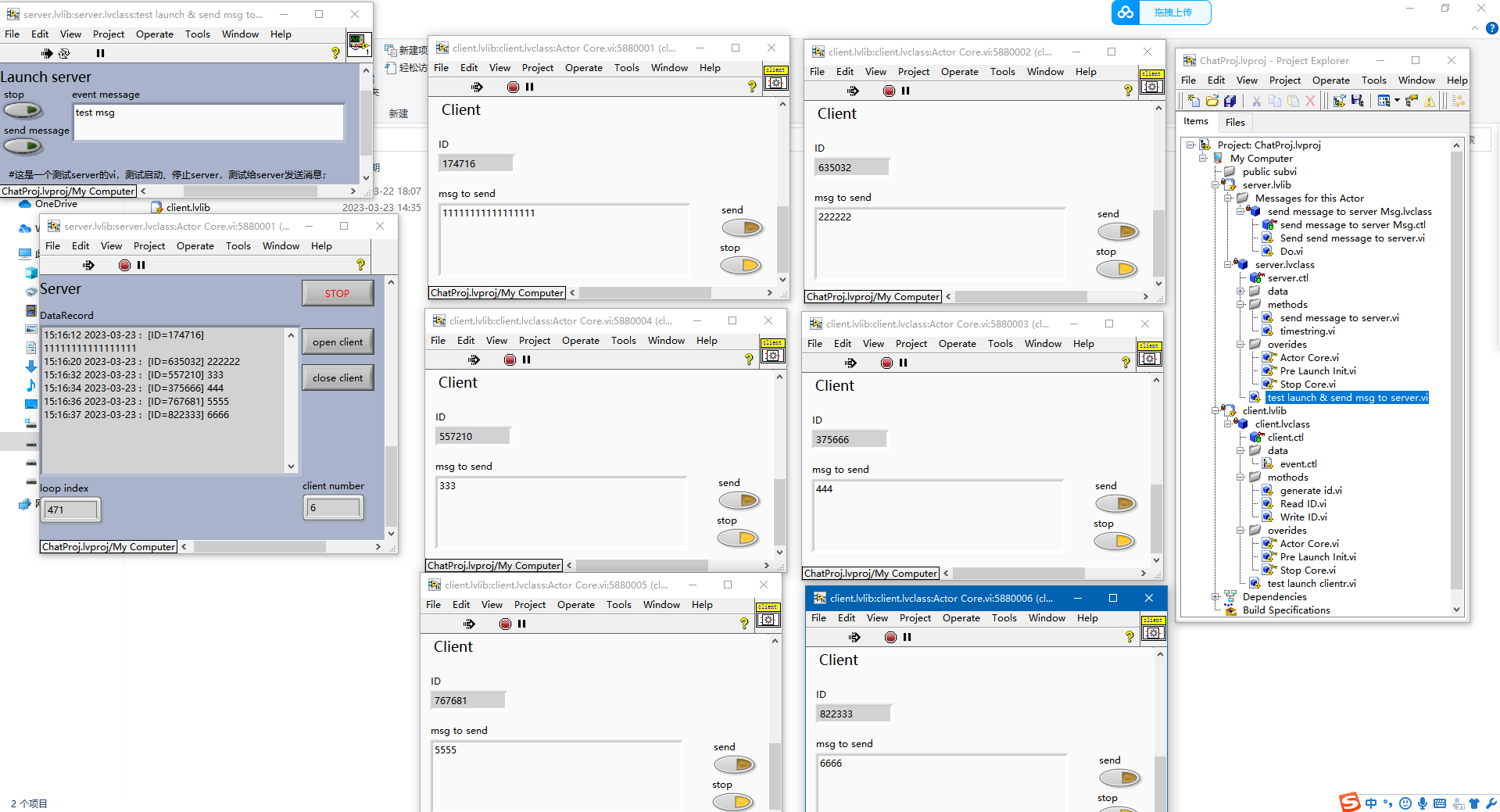
Cilent便可以通过获取到的server队列通过server消息队列方法发送信息给server端了：



剩下便是一些释放，显示cilent数量等小问题修补一下；client的数量便是通过一个client的实例化的对象（全部存在一个一维数组中）个数了。



最后留一张整个demo执行的效果图：



——

看一千遍，不如自己动手一遍。实践出真知，动手自己编写一次这个例子，才会真正理解与体会Actor Framwork的关键与精妙。

大家加油！