一、基础server:

1、方法：

启动server：

停止server：

运行server：

初始化server：

1. 属性

名称：

监听的ip：

监听的端口：

1. 链接模块。连接封装和业务绑定

1、方法：

启动连接Start()

停止连接Stop()

获取当前链接的conn对象(套接字)。GetTCPConnection() \*net.TCPConn

得到连接id。GetConnId() unit32

得到客户端链接的地址和端口。RemoteAddr() net.Addr

发送数据的方法Send()。Send(data []byte) error

链接所绑定的处理业务的函数类型。Type HandleFunc func(\*net.TCPConn, []byte, int) error

2、属性：

Socket tcp套接字。Conn \*net.TCPConn

连接的id。ConnID unit32

当前链接的状态(是否已经关闭)。isClosed bool

与当前连接所绑定的处理业务方法。HandleAPI ziface.HandleFunc

等待链接被动退出的channel。ExitChan chan bool

三、router模块

四、全局配置

五、消息封装

1、属性：

2、Setter、Getter方法

获取消息id：GetMsgId() unit32

获取消息长度：GetMsgLen() unit32

获取消息内容：GetMsg() []byte

设置消息id：SetMsgId(unit32)

设置消息长度：SetMsgLen(unit32)

设置消息内容：SetMsg([]byte)

3、定义一个解决tcp沾包问题的封包拆包模块

A、针对msg进行tlv格式的封包：

写msg的长度

写msg的id

写msg的内容

B、针对msg进行tlv格式的拆包：

先读取固定长度的head：消息内容的长度和消息的类型

再根据消息内容的长度，再次进行一次读写，从conn中读取消息的内容

4、将消息封装机制集成到框架中

将msg添加到request属性中

修改链接读取数据的机制

给链接提供一个法宝机制：将发送的消息进行封包再发送

1. 多路由模块
2. 消息管理模块(支持多路由业务api调度管理的)--MsgHandler

属性：集合--消息id和对应的router的关系-map。Apis map(unit32) ziface.IRouter

方法：根据MsgId来索引调度路由方法。Func(mh \*MsgHandle) AddRouter(msgId unit32, router ziface.IRouter)

1. 消息管理模块到zinx框架中

将server模块中的router属性替换成MsgHandler属性

将server之前的AddRouter修改 调用MsgHandler的AddrRouter(msgId uint32, router ziface.IRouter)

1. 将connection模块中Router属性替换成MsgHandler，修改初始化Connection方法
2. Connection的之前调度Router的业务替换成MsgHandler调度，修改StartReader方法
3. 读写协程分离
4. 添加一个Reader和Writer之间通信的channel
5. 添加一个Writer goroutine
6. Reader由之前直接发送给客户端，改成发送给通信Channel
7. 启动Reader和Writer一同工作
8. 消息队列及多任务
9. 创建一个消息队列--MsgHandler消息管理模块--

消息队列--TaskQueue []chan ziface.IRequest

worker工作池的数量

1. 创建多任务worker的工作池并启动

根据workerPoolSize的数量创建Worder，每个worder都用一个go去承载：

阻塞的等待与当前worder对应的channel的消息，一旦有消息到来，worker应该处理当前消息对应的业务

1. 将之前的发送消息，改成把消息发送个消息队列和worker工作池来处理

九、链接管理

1、创建一个链接管理模块

属性：

Connection集合-map：connections map[uint32] ziface.IConnection

针对map的互斥锁--connLock

方法：

添加连接。func(connMgr \*ConnManager) Add(conn ziface.IConnection)

删除连接。func(connMgr \*ConnManager) Remove(conn ziface.IConnection)

根据连接id查找对应的链接.func(connMgr \*ConnManager) Get(connId uint32) (conn ziface.IConnection, error)

总连接个数。func(connMgr \*ConnManager) Len() int

清理全部的链接。func(connMgr \*ConnManager) ClearConn()

1. 将链接管理模块集成到框架中

将ConnManager加入Server模块中。给server添加一个ConnMgr属性，修改NewServer方法，加入ConnMgr初始化，判断当前的连接数量是否已经超出最大值MaxConn

每次成功与客户端建立连接后，添加链接到ConnManager中。

每次与客户端链接断开后，将链接从ConnManager中删除

1. 给框架提供创建连接之后、销毁连接之前所要处理的一些业务提供给用户能工注册Hook函数

添加属性：

改server创建链接之后自动调用Hook函数--OnConnStart func(conn ziface.IConnection)

该Server销毁链接之前自动调用改的Hook函数--OnConnStop func(conn ziface.IConnection)

添加的方法：

注册OnConnStart钩子函数的方法--func(s \*Server) SetOnConnStart(hookFunc func(connection ziface.IConnection))

注册OnConnStop钩子函数的方法--func(s \*Server) SetOnConnStop(hookFunc func(connection ziface.IConnection))

调用OnConnStart钩子函数的方法--func(s \*Server) CallOnConnStart(conn ziface.IConnection)

调用OnConnStop钩子函数的方法--func(s \*Server) CallOnConnStop(conn ziface.IConnection)

在Conn创建之后调用OnConnStart，在conn.Start()中调用

在Conn创建之后调用OnConnStop，在conn.Stop()中调用

十、链接属性配置

给Connection模块添加可以配置属性的功能

新增的属性：

链接属性集合--ma。Property map(string) interface{}

保护链接属性的锁。 propertyLock sys.RWMutex

新增的方法：

设置连接属性--func(c \*Connection) SetProperty(key string, val interface{})

获取连接属性---func(c \*Connection) GetProperty(key string, val interface{}, error)

移除连接属性---func(c \*Connection) RemoveProperty(key string)

server.go

message.go

datapack.go

connection.go

request.go