-

TCPTCPgtcp.Conngtcp.ConnSendRecv

Recv*switch...case...goroutine

> Send*/Recv*gtcp.Conn

https://github.com/gogf/gf/v2/tree/master/.example/net/gtcp/pkg_operations/common

1. types/types.go

   SendPkg/RecvPkg

   JSONJSONActuint8

   ```
   package types


   type Msg struct {
       Act  string //
       Data string //
   }
   ```

2. funcs/funcs.go

   //

```
package funcs


import (
    "encoding/json"
    "fmt"
    "github.com/gogf/gf/v2/net/gtcp"
    "github.com/gogf/gf/.example/net/gtcp/pkg_operations/common
/types"
)


//
func SendPkg(conn *gtcp.Conn, act string, data...string) error {
    s := ""
    if len(data) > 0 {
        s = data[0]
    }
    msg, err := json.Marshal(types.Msg{
        Act  : act,
        Data : s,
    })
    if err != nil {
        panic(err)
    }
    return conn.SendPkg(msg)
}


//
func RecvPkg(conn *gtcp.Conn) (msg *types.Msg, err error) {
    if data, err := conn.RecvPkg(); err != nil {
        return nil, err
    } else {
        msg = &types.Msg{}
        err = json.Unmarshal(data, msg)
        if err != nil {
            return nil, fmt.Errorf("invalid package structure: %s",
err.Error())
        }
        return msg, err
    }
}
```

3. gtcp_common_server.go

   10doexit

```
package main


import (
    "github.com/gogf/gf/v2/net/gtcp"
    "github.com/gogf/gf/v2/os/glog"
    "github.com/gogf/gf/v2/os/gtimer"
    "github.com/gogf/gf/.example/net/gtcp/pkg_operations/common
/funcs"
    "github.com/gogf/gf/.example/net/gtcp/pkg_operations/common
/types"
    "time"
)


func main() {
    gtcp.NewServer("127.0.0.1:8999", func(conn *gtcp.Conn) {
        defer conn.Close()
        // , 10
        gtimer.SetTimeout(10*time.Second, func() {
            funcs.SendPkg(conn, "doexit")
        })
        for {
            msg, err := funcs.RecvPkg(conn)
            if err != nil {
                if err.Error() == "EOF" {
                    glog.Println("client closed")
                }
                break
            }
            switch msg.Act {
                case "hello":     onClientHello(conn, msg)
                case "heartbeat": onClientHeartBeat(conn, msg)
                default:
                    glog.Errorf("invalid message: %v", msg)
                    break
            }
        }
    }).Run()
}


func onClientHello(conn *gtcp.Conn, msg *types.Msg) {
    glog.Printf("hello message from [%s]: %s", conn.RemoteAddr().
String(), msg.Data)
    funcs.SendPkg(conn, msg.Act, "Nice to meet you!")
}


func onClientHeartBeat(conn *gtcp.Conn, msg *types.Msg) {
    glog.Printf("heartbeat from [%s]", conn.RemoteAddr().String())
}
```

4. gtcp_common_client.go

   forSendPkg

   gtimer13hellogtimerTCP

   10doexit


```
package main


import (
    "github.com/gogf/gf/v2/net/gtcp"
    "github.com/gogf/gf/v2/os/glog"
    "github.com/gogf/gf/v2/os/gtimer"
    "github.com/gogf/gf/.example/net/gtcp/pkg_operations/common
/funcs"
    "github.com/gogf/gf/.example/net/gtcp/pkg_operations/common
/types"
    "time"
)


func main() {
    conn, err := gtcp.NewConn("127.0.0.1:8999")
    if err != nil {
        panic(err)
    }
    defer conn.Close()
    //
    gtimer.SetInterval(time.Second, func() {
        if err := funcs.SendPkg(conn, "heartbeat"); err != nil {
            panic(err)
        }
    })
    // , 3hello
    gtimer.SetTimeout(3*time.Second, func() {
        if err := funcs.SendPkg(conn, "hello", "My name's John!");
err != nil {
            panic(err)
        }
    })
    for {
        msg, err := funcs.RecvPkg(conn)
        if err != nil {
            if err.Error() == "EOF" {
                glog.Println("server closed")
            }
            break
        }
        switch msg.Act {
            case "hello":     onServerHello(conn, msg)
            case "doexit":    onServerDoExit(conn, msg)
            case "heartbeat": onServerHeartBeat(conn, msg)
            default:
                glog.Errorf("invalid message: %v", msg)
                break
        }
    }
}


func onServerHello(conn *gtcp.Conn, msg *types.Msg) {
    glog.Printf("hello response message from [%s]: %s", conn.
RemoteAddr().String(), msg.Data)
}


func onServerHeartBeat(conn *gtcp.Conn, msg *types.Msg) {
    glog.Printf("heartbeat from [%s]", conn.RemoteAddr().String())
}


func onServerDoExit(conn *gtcp.Conn, msg *types.Msg) {
    glog.Printf("exit command from [%s]", conn.RemoteAddr().String())
    conn.Close()
}
```

5.

- ```
  2019-05-03 14:59:13.732 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:14.732 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:15.733 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:15.733 hello message from [127.0.0.1:51220]:
My name's John!
  2019-05-03 14:59:16.731 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:17.733 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:18.731 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:19.730 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:20.732 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:21.732 heartbeat from [127.0.0.1:51220]
  2019-05-03 14:59:22.698 client closed
  ```

- ```
  2019-05-03 14:59:15.733 hello response message from
[127.0.0.1:8999]: Nice to meet you!
  2019-05-03 14:59:22.698 exit command from [127.0.0.1:8999]
  ```