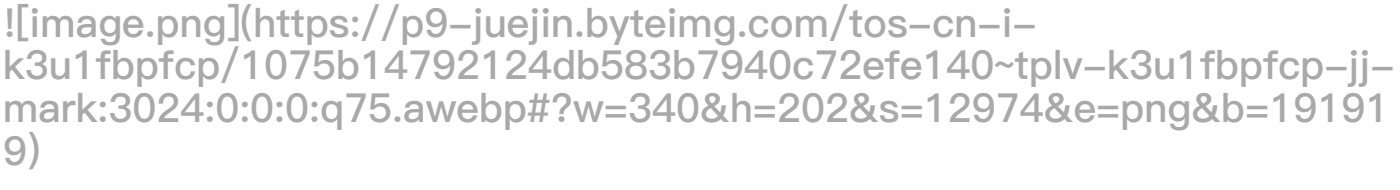
第一章节: 创建WebSocket Server & Client

创建Go工程

在VSCode中创建工程go-im, 并创建src文件夹



打开Terminal进入go-im项目下

```
go mod init aoki.com/go-im
```

安装依赖

初始化项目后安装gin和gorilla-websocket依赖包

```
go get -u github.com/gin-gonic/gin
go get -u github.com/gorilla/websocket
```

编写WebSocket服务端

先创建最简单的WS服务端，在src目录下创建main文件夹, 在main文件夹下创建server.go, 代码如下:

```
package main

import (
”log”
”net/http”

”github.com/gin-gonic/gin”
”github.com/gorilla/websocket”
)

var upgrader = websocket.Upgrader{
    // Allow Cross Origin
    CheckOrigin: func(r *http.Request) bool {
        return true
    },
}

// 处理WS
func ws(c *gin.Context) {
    // 升级为WS协议
    conn, err := upgrader.Upgrade(c.Writer, c.Request, nil)
    if err != nil {
        panic(err)
    }
    defer conn.Close()
    log.Printf(”A New Connect... \n”)
    // 循环读取数据
    for {
        mt, message, err := conn.ReadMessage()
        if err != nil {
            log.Printf(”Read Message Failed... \n”, err)
            break
        }
        // 读到什么往WS客户端发什么
        log.Printf(”Received Message %s... \n”, message)
        err = conn.WriteMessage(mt, message)
        if err != nil {
            log.Printf(”Write Message Failed... \n”, err)
            break
        }
    }
}

// 启动Websocket server
```

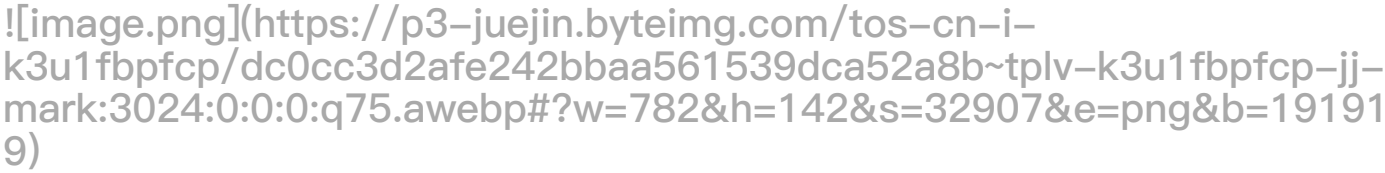
```
func main() {
    server := gin.Default()
    server.GET("/ws", ws)
    server.Run("localhost:8848")
}
```

启动Server

在Terminal中启动Websocket server

```
go run ./src/main/server.go
```

可以在Ternimal看到以下日志输出，表示启动成功



创建客户端

在src/main下面创建client.go, 创建WebSocket客户端并连接Server, 尝试从Terminal读取内容并通过WebSocket发送

```
package main

import (
    "bufio"
    "fmt"
    "log"
    "net/http"
    "os"
```

```
    "github.com/gorilla/websocket"
)

func main() {
    uri := "ws://localhost:8848/ws"
    conn, _, err := websocket.DefaultDialer.Dial(uri, nil)
    if err != nil {
        log.Fatal("dial:", err)
    }
    defer conn.Close()

    go func() {
        for {
            _, message, err := conn.ReadMessage()
            if err != nil {
                fmt.Println("Read WS message failed:", err)
                return
            }
            log.Println("Received: ", string(message))
        }
    }()

    scanner := bufio.NewScanner(os.Stdin)
    for scanner.Scan() {
        line := scanner.Text()
        if err := conn.WriteMessage(websocket.TextMessage, []byte(line));
err != nil {
            log.Println("write failed:", err)
        }
    }
    if err := scanner.Err(); err != nil {
        log.Fatal(err)
    }
}
```

启动Client

新开一个Terminal然后输入命令启动Client

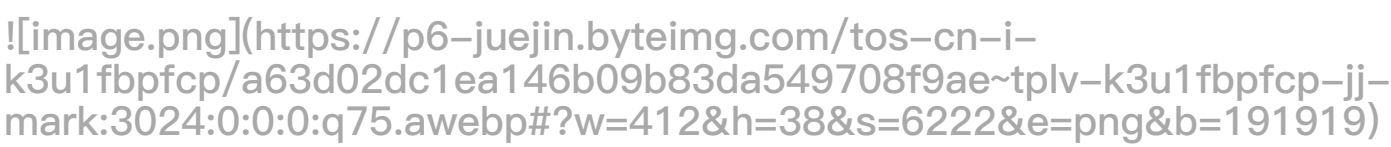
```
go run ./src/main/client.go
```

```

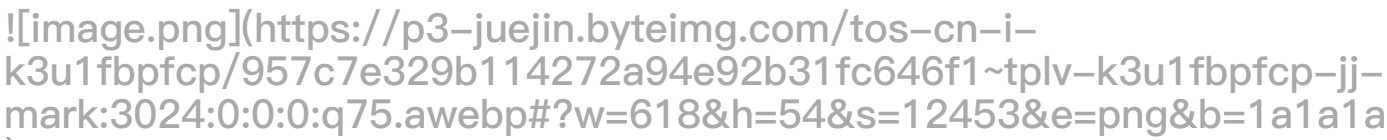
在Server端的Terminal中可以看到日志输出


通信聊天
---

在Client端的Terminal中输入内容, 可以看到日志输出




在Server端的Terminal中也可以看到相应的日志




可以看到客户端与服务端通信没有任何问题，但是我们最终需要的是用户A与用户B之间进行通信，所以我们的Server端还需要做很多工作。

下一章节将尝试根据WS连接获取到登录用户并以此来区分每个客户端连接。

原文链接: https://juejin.cn/post/7359567256583929890
```