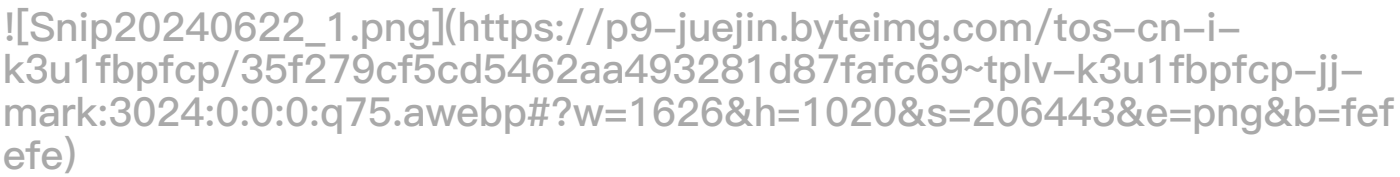
/localhost:8080/ws`，开启连接，连接成功后即可发生信息。



终端信息如下：

```
2024/06/22 20:18:58 server startup at 127.0.0.1:8080
2024/06/22 20:19:10 连接成功
2024/06/22 20:19:14 recv: 您好啊
2024/06/22 20:19:21 recv: 我是AA
```

此时我们已经实现了一个简单的websocket双向通信功能，可以发现websocket的使用总体分成三步：

1. 借用http处理请求
2. 升级请求为websocket连接
3. 利用返回的websocket连接发送和接收信息

3. 一个聊天服务器demo

虽然上面我们通过websocket实现了简单的双向通信，但是实际这上没有什么用 —— 每次回复的消息都只是回显信息而已；我们能否实现一个真正可以聊天的demo呢?

我们来分析下，上面我们已经能做到双向通信了，只是需要解决收到一个消息后推送给其他客户端的问题，而这个问题需要解决两点：

1. 服务器要能区分不同的客户端
2. 当收到消息后，需要能知道转发给哪个客户端

第一点很容易解决，我们只需要给每个客户端分配一个唯一的id即可，然后通过id来区分客户端。第二点，我们可以约定一个消息格式，比如：`id:msg`,这样收到消息后就知道转发给哪个客户端。

1. 初步实现聊天

下面看具体代码实现

```
// mian.go
package main

import (
”log”
”net/http”
”strconv”
”strings”

”github.com/gorilla/websocket”
)

var (
// 全局连接 id – conn 用于区分不同客户端
globleConn = make(map[int16](*websocket.Conn))
// 当前最大id
maxKey int16 = 1

upgrader = websocket.Upgrader{CheckOrigin: func(r *http.Request) bool
{
return true // 允许跨域
}}
)

func serveHome(w http.ResponseWriter, r *http.Request) {
ws, err := upgrader.Upgrade(w, r, nil)
if err != nil {
log.Println(err)
return
}
defer ws.Close()
addGlobalConn(ws)

for {
```

```
_, message, err := ws.ReadMessage()
if err != nil {
log.Println(”接收信息发生错误： ”, err)
return
}
log.Printf(”recv: %s”, message)
// 解析出要发送的客户端
conn, msg := parseMessage(message)
if conn != nil {
pushMessage(conn, msg)
}
}
}

// 记录conn到GlobalConn
func addGlobalConn(ws *websocket.Conn) {
globleConn[int16(maxKey)] = ws
log.Printf(”======编号： %d连接加入成功”, maxKey)
maxKey += 1
}

// 客户端消息解析出id和消息
// 返回conn 和 消息
func parseMessage(message []byte) (*websocket.Conn, string) {
// 假设消息格式为 ”receiver:message”
messageParts := strings.SplitN(string(message), ”:”, 2)
if len(messageParts) != 2 {
return nil, ””
} else {
re } else {
return nil, ””
}
}
}

// 推送消息到客户端
func pushMessage(ws *websocket.Conn, message string) {
err := ws.WriteMessage(websocket.TextMessage, []byte(message))
if err != nil {
log.Println(”write:”, err)
}
log.Printf(”send: %s成功”, message)
}

func main() {
http.HandleFunc(”/ws”, serveHome)
log.Println(”server startup at 127.0.0.1:8080”)
log.Fatal(http.ListenAndServe(”127.0.0.1:8080”, nil))
```

}

```

我们优化后再运行测试，下面是终端测试结果

```
2024/06/22 22:46:52 server startup at 127.0.0.1:8080
2024/06/22 22:47:05 ======编号：1连接加入成功
2024/06/22 22:47:07 ======编号：2连接加入成功
2024/06/22 22:47:10 1号 发送心跳成功
2024/06/22 22:47:12 2号 发送心跳成功
2024/06/22 22:47:15 1号 发送心跳成功
2024/06/22 22:47:17 2号 发送心跳成功
2024/06/22 22:47:19 recv: 2:2号你好 我是1号
2024/06/22 22:47:19 send: 2号你好 我是1号成功
2024/06/22 22:47:20 1号 发送心跳成功
2024/06/22 22:47:22 2号 发送心跳成功
2024/06/22 22:47:25 1号 发送心跳成功
2024/06/22 22:47:27 2号 发送心跳成功
2024/06/22 22:47:30 1号 发送心跳成功
2024/06/22 22:47:30 recv: 1:1号我收到了你的消息
2024/06/22 22:47:30 send: 1号我收到了你的消息成功
2024/06/22 22:47:32 2号 发送心跳成功
2024/06/22 22:47:35 1号 发送心跳成功
2024/06/22 22:47:37 2号 发送心跳成功
2024/06/22 22:47:38 recv: 2:我们聊聊吧
2024/06/22 22:47:38 send: 我们聊聊吧成功
2024/06/22 22:47:40 1号 发送心跳成功
2024/06/22 22:47:42 2号 发送心跳成功
2024/06/22 22:47:45 recv: 1:好呀
2024/06/22 22:47:45 send: 好呀成功
2024/06/22 22:47:45 1号 发送心跳成功
2024/06/22 22:47:47 2号 发送心跳成功
2024/06/22 22:47:49 1号 读取发生错误websocket: close 1000 (normal): Active closure of the user:
2024/06/22 22:47:50 1号 心跳发送失败: websocket: close sent
2024/06/22 22:47:52 2号 发送心跳成功
2024/06/22 22:47:54 1号 读取发生错误websocket: close 1000 (normal): Active closure of the user:
2024/06/22 22:47:55 1号 心跳发送失败: websocket: close sent
2024/06/22 22:47:57 2号 发送心跳成功
2024/06/22 22:47:59 1号 读取发生错误websocket: close 1000 (normal): Active closure of the user:
2024/06/22 22:48:00 1号 心跳发送失败: websocket: close sent
2024/06/22 22:48:00 1号 心跳失败次数达到最大值（3次）停止发送心跳
```
```

，发送取消信号
2024/06/22 22:48:02 2号 发送心跳成功

```

### 4. 最后

上面的代码只是一个基本的实现，并不完善，还有许多地方要优化；考虑到篇幅和入门使用的原因，没有必要面面俱到，希望能起到抛砖引玉的作用供大家参考。

原文链接: https://juejin.cn/post/7382980041398501411
```