



链滴

gin 框架中结合 gorilla 实现 websocket

作者: [450370050](#)

原文链接: <https://ld246.com/article/1524469717858>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

gin框架写rest接口特别6，性能也很好。
本文介绍一下gin和gorilla结合创建websocket

1. gin

github地址: <https://github.com/gin-gonic/gin>

安装gin的代码包

```
go get github.com/gin-gonic/gin

package main

import (
    "github.com/gin-gonic/gin"
)

func main() {
    bindAddress := "localhost:2303"
    r := gin.Default()
    //监听 get请求 /test路径
    r.GET("/test", func(c *gin.Context) {
        c.JSON(200, []string{"123", "321"})
    })
    r.Run(bindAddress)
}
```

上面是个简单的gin请求例子，更多使用方式请查看官方介绍

2. gorilla

gorilla拥有很多类库，这次我们使用的是它的webSocket

```
go get github.com/gorilla/websocket

package main

import (
    "github.com/gin-gonic/gin"
    "github.com/gorilla/websocket"
    "net/http"
)

var upGrader = websocket.Upgrader{
    CheckOrigin: func (r *http.Request) bool {
        return true
    },
}

//webSocket请求ping 返回pong
func ping(c *gin.Context) {
```

```

//升级get请求为websocket协议
ws, err := upGrader.Upgrade(c.Writer, c.Request, nil)
if err != nil {
    return
}
defer ws.Close()
for {
    //读取ws中的数据
    mt, message, err := ws.ReadMessage()
    if err != nil {
        break
    }
    if string(message) == "ping" {
        message = []byte("pong")
    }
    //写入ws数据
    err = ws.WriteMessage(mt, message)
    if err != nil {
        break
    }
}
}

func main() {
    bindAddress := "localhost:2303"
    r := gin.Default()
    r.GET("/ping", ping)
    r.Run(bindAddress)
}

```

3. js代码

```

<script>
var ws = new WebSocket("ws://localhost:2303/ping");
//连接打开时触发
ws.onopen = function(evt) {
    console.log("Connection open ...");
    ws.send("Hello WebSockets!");
};
//接收到消息时触发
ws.onmessage = function(evt) {
    console.log("Received Message: " + evt.data);
};
//连接关闭时触发
ws.onclose = function(evt) {
    console.log("Connection closed.");
};

</script>

```

4. 测试

浏览器访问html代码，查看浏览器调试模式中的Console输出

Connection open ...
Received Message: Hello WebSockets!