# webSocket共享session研究报告

## 背景

ECSC中存在向客户端推送消息的需求，需要引入webSocket技术。但是在之前的研究中发现spring的webSocketSession类不支持序列化方法，会影响到集群共享Session，所以需要深入研究webSocket如何共享Session。

## 可能的解决方案

###### 升级spring版本

首先想到的是更换springMVC版本，看看在新版中有没有支持序列化。然后升级springMVC到4.3.0版本，发现webSocketSession依然不支持序列化。

###### SpringSession

在升级版本不可行之后，开始在网上搜索相关信息，最终在stackoverflow上看到有人遇到同样的问题（http://stackoverflow.com/questions/28890425/can-websocket-session-be-serialized-or-specifically-persisted-to-a-database），评论中提到SpringSession可能能够解决这个问题。

在阅读了SpringSession的文档和使用之后，发现SpringSession可以管理HTTPSession和webSocketSession，并且能够和redis集成，但是依然无法实现集群多个服务共享*webSocketSession。*

###### SocketIO

在SpringSession方案也没走通之后，开始尝试使用nodejs提供webSocket服务。首先使用express创建了一个简单的web服务。之后引入socketIO（http://socket.io/）提供webSocket服务，在成功简易的聊天室服务之后，开始尝试使用redis共享Session，首先引入了nodejs的redis模块，成功连接上了redis服务之后，在github上找到了socketIO使用redis的适配器socket.io-redis（https://github.com/socketio/socket.io-redis）。成功引入之后，起了两个服务，经测试两个服务可以互相通信，Session能够共享。

**总结：**

|  |  |  |
| --- | --- | --- |
| 可能的方案 | 验证方法 | 结论 |
| 升级spring版本 | 升级到4.3.0 | webSocketSession不支持序列化 |
| SpringSession | 搭建一个简单的服务，并引入SpringSession | 不能再多个服务中共享webSocketSession |
| Nodejs | 搭建了一个聊天室的服务 | 能够使用redis共享Session |

## 可行的方案

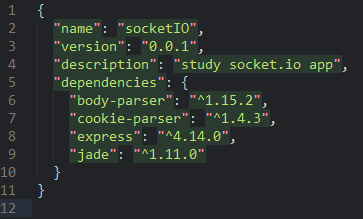
###### nodeJS+express+socketIO+redis

使用nodeJS+socketIO创建webSocket的服务，在ECSC的页面中引入socketIO的客户端脚本，连接到webSocket的服务。其他后台服务通过访问webSocket服务的接口向客户端推送消息。

下面详述一下创建建议聊天室服务的步骤：

**一、创建一个web服务**

首先创建package.json



使用npm install 安装依赖的npm模块

在根目录下创建app.js文件

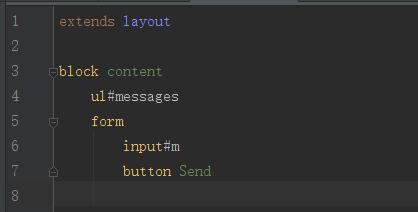


这里使用了jade作为html模板引擎，页面模板放在views目录下，路由定义在routes目录下，静态资源放在public目录下，使用express提供静态资源服务。

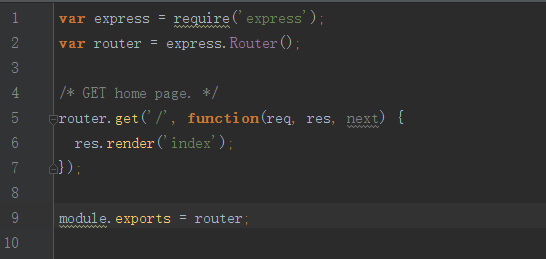
views/layout.jade



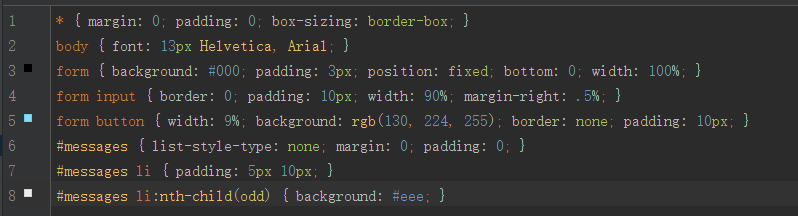
views/index.jade



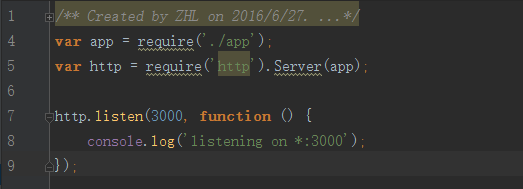
routes/index.js



public/style/style.css



创建启动脚本，index.js文件

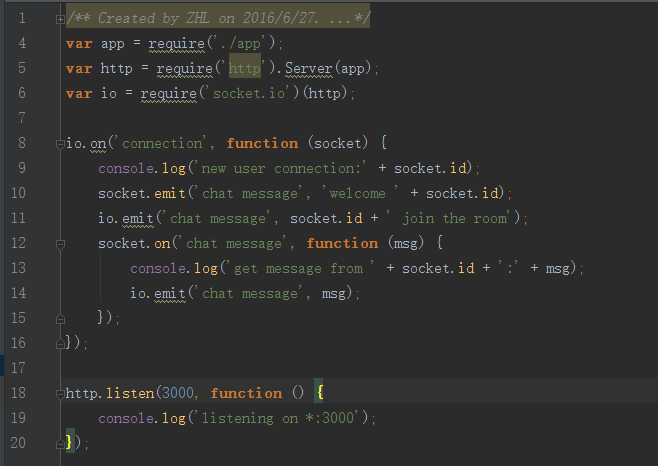


使用node index.js命令启动服务，访问localhost：3000可以看到建议的聊天室的界面，至此，我们已经创建了一个简单的web服务。接下来要引入socketIO来完成聊天室的功能。

**二、引入socketIO**

首先在服务端引入socketIO，使用命令npm install –save socket.io

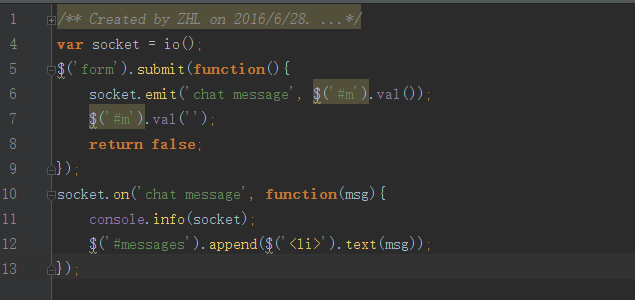
安装好socket.io包之后，修改启动脚本index.js



然后我们还需要在客户端引入socketIO，下载socketIO（与服务端版本一致）的客户端脚本，放在public/lib目录下



创建public/script/chat.js文件，完成聊天室页面逻辑

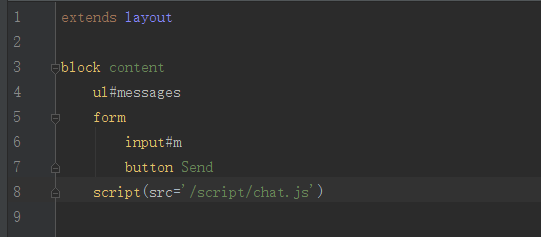


在页面中引入新建的js文件

Views/layout.jade



Views/index.jade

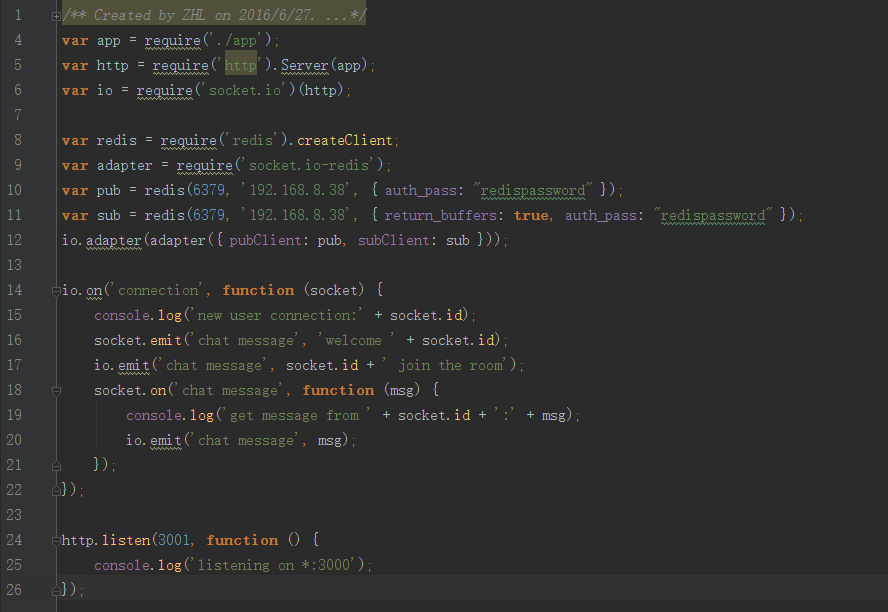


重启服务，至此，我们完成了聊天室的功能。接下来需要引入redis实现Session共享。

**三、实现Session共享**

使用命令npm install –save redis socket.io-redis 安装redis包和socketIO的redis适配包。

安装成功后修改启动脚本index.js文件



重启服务，至此，我们已经完成了Session的共享，可以通过修改端口号启动多个服务进行测试。