链滴

Spring boot + Vue 实现 WebSocket

作者：Parker

原文链接：https://ld246.com/article/1589027967675

来源网站：链滴

来，搞事情，pom文件加上依赖

```
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-websocket</artifactId>
        <version>4.3.8.RELEASE</version>
</dependency>
```

主要有三个文件

1. WebSocketsConfig.java (WebSocket配置文件)

2. MyWebSocketInterceptor.java (可以看做是一个拦截器，beforeHandshake(websocket握手前), afterHandshake(握手之后))

3. WebSocketPushHandler.java（Websocket一些事件）

WebSocketsConfig.java

```
package sucblog.config;


import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.socket.WebSocketHandler;
import org.springframework.web.socket.config.annotation.EnableWebSocket;
import org.springframework.web.socket.config.annotation.WebSocketConfigurer;
import org.springframework.web.socket.config.annotation.WebSocketHandlerRegistry;
import org.springframework.web.socket.server.HandshakeInterceptor;
import sucblog.websocket.MyWebSocketInterceptor;
import sucblog.websocket.WebSocketPushHandler;

@Configuration
@EnableWebSocket
public class WebSocketsConfig implements WebSocketConfigurer {

    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(createWebSocketPushHandler(), "/webSocketServer")
                .addInterceptors(createHandshakeInterceptor()).setAllowedOrigins("*");
        registry.addHandler(createWebSocketPushHandler(), "/sockjs/webSocketServer")
                .addInterceptors(createHandshakeInterceptor()).withSockJS();
    }


    /**
     * 握手拦截器
     * @return
     */
    @Bean
    public HandshakeInterceptor createHandshakeInterceptor() {
        return new MyWebSocketInterceptor();
    }

    @Bean
```

```
    public WebSocketHandler createWebSocketPushHandler() {
        return new WebSocketPushHandler();
    }
}
```

MyWebSocketInterceptor

```
package sucblog.websocket;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.server.ServerHttpRequest;
import org.springframework.http.server.ServerHttpResponse;
import org.springframework.http.server.ServletServerHttpRequest;
import org.springframework.web.socket.WebSocketHandler;
import org.springframework.web.socket.server.HandshakeInterceptor;

import java.util.Map;

public class MyWebSocketInterceptor implements HandshakeInterceptor {
    private Logger logger = LoggerFactory.getLogger(this.getClass());

    /**
     * 在握手之前执行该方法，继续握手返回true ,中断握手返回false，通过map参数设置WebSocket
ession的属性
     * @param request
     * @param response
     * @param webSocketHandler
     * @param map
     * @return
     * @throws Exception
     */
    @Override
    public boolean beforeHandshake(ServerHttpRequest request, ServerHttpResponse respons
, WebSocketHandler webSocketHandler, Map<String, Object> map) throws Exception {
        logger.info("xxx用户建立连接...");
        if (request instanceof ServletServerHttpRequest) {
            String userId = ((ServletServerHttpRequest) request).getServletRequest().getParameter
"userId");
            map.put("userId", userId);
            logger.info("用户唯一标识：" + userId);
        }
        return true;
    }


    /**
     * 在握手之后执行该方法，无论是否握手成功都致命了响应的状态码和响应头
     * @param serverHttpRequest
     * @param serverHttpResponse
     * @param webSocketHandler
     * @param e
     */
```

```
    @Override
    public void afterHandshake(ServerHttpRequest serverHttpRequest, ServerHttpResponse se
verHttpResponse, WebSocketHandler webSocketHandler, Exception e) {

    }
}
```

WebSocketPushHandler

```
package sucblog.websocket;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.socket.CloseStatus;
import org.springframework.web.socket.TextMessage;
import org.springframework.web.socket.WebSocketSession;
import org.springframework.web.socket.handler.TextWebSocketHandler;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class WebSocketPushHandler extends TextWebSocketHandler {
    private Logger logger = LoggerFactory.getLogger(this.getClass());
    private static final List<WebSocketSession> userList = new ArrayList<>();


    /**
     * 用户进入 系统监听
     * @param session
     * @throws Exception
     */
    @Override
    public void afterConnectionEstablished(WebSocketSession session) throws Exception {
        logger.info(session.getAttributes() + "用户进入系统。。。");
        logger.info("用户信息:" + session.getAttributes());
        Map<String, Object> map = session.getAttributes();
        for (String key : map.keySet()) {
            logger.info("key:" + key + "and value: " + map.get(key));
        }
        userList.add(session);
        sendMessagesToUsers(session);
    }


    /**
     * 处理用户请求
     * @param session
     * @param message
     */
    @Override
    protected void handleTextMessage(WebSocketSession session, TextMessage message) thr
```

```
ws Exception{
        logger.info("系统处理["+session.getAttributes().get("userId")+"]用户的请求信息==>" + me
sage.getPayload());
        sendMessageToUser((String) session.getAttributes().get("userId"), message);
    }


    /**
     * 用户退出后的处理
     * @param session
     * @param status
     * @throws Exception
     */
    @Override
    public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws E
ception {
        if (session.isOpen()) {
            session.close();
        }
        userList.remove(session);
        logger.info("xxx用户退出系统");
    }


    /**
     * 自定义函数
     * 给所有在线用户发送消息
     * @param
     */
    public void sendMessagesToUsers(WebSocketSession session) {
        String msg = "欢迎'" + session.getAttributes().get("userId");
        TextMessage message = new TextMessage(msg);
        for (WebSocketSession user : userList) {
            try {
                // isOpen() 在线就发送
                if (user.isOpen()) {
                    user.sendMessage(message);
                }
            } catch (IOException e) {
                e.printStackTrace();
                logger.error(e.getLocalizedMessage());
            }
        }
    }


    public void sendMessageToUser(String userId, TextMessage message) {
        for (WebSocketSession user : userList) {
            try {
                if (user.isOpen()) {
                    user.sendMessage(message);
                }
            } catch (IOException e) {
                e.printStackTrace();
```

```
                logger.error(e.getLocalizedMessage());
            }
        }
    }
}
```

前端代码

```
<template>
    <div style="margin: 150px auto; width: 500px;box-shadow: 0 0 20px #33333350; border-ra
ius: 10px; padding: 20px;">
      <h2>websocket学习学习</h2>
      <h3>学海无涯，回头是岸</h3>
      <div style="margin-top: 20px; text-align: left">
        <input type="text" v-model="msg">
        <button @click="websocketsend(msg)">发送</button>
        <div v-if="bmsg">
          <ul>
            <li v-for="(item, i) in bmsg" :key="i">
              <strong :class="{services:item.user==='服务器'}">{{item.user}}</strong>
              回复时间：{{item.time}}
              <div>
                {{item.content}}
              </div>
            </li>
          </ul>
      </div>
      </div>
    </div>
</template>

<script>
export default {
  name: 'KolOnlineChat',
  data() {
    return {
      websock: null,
        msg: null,
        bmsg: [],
    }
  },
  mounted() {
    this.initWebSocket()
  },
  destroyed() {
    this.websock.close()  // 离开路由断开websocket连接
  },
  methods: {
    initWebSocket() { // 初始化websocket
      const wsuri = "wss://localhost:9999/webSocketServer?userId=parker"
      this.websock = new WebSocket(wsuri)
      this.websock.onmessage = this.websocketonmessage
      this.websock.onopen = this.websocketonopen
```

```
      this.websock.onerror = this.websocketonerror
      this.websock.onclose = this.websocketclose
    },
    websocketonopen() { //连接建立之后执行send方法发送数据
      // let actions = {'test': '12345'}
      // this.websocketsend(JSON.stringify(actions))
    },
    websocketonerror(){//连接建立失败重连
      this.initWebSocket()
    },
    websocketonmessage(e){ //数据接收
      const redata = JSON.stringify(e.data)
      this.arrOpear(redata, 'services')
    },
    websocketsend(Data){//数据发送
      this.websock.send(Data)
      this.arrOpear(Data, 'localhost')
    },
    websocketclose(e){  //关闭
      console.log('断开连接',e)
    },
    arrOpear(msg, type) {
      let _b = {
          content: msg,
          user: type === 'services'? '服务器' : '我',
          time: new Date().toLocaleTimeString()
      }
      this.bmsg.push(_b)
    }
  }
}
</script>

<style scoped lang="less">
strong {
  color: #03be01;
}
strong.services {
  color: orangered;
}
</style>
```

运行截图：

原文链接：Spring boot + Vue 实现 WebSocket