WebSocket的集成

> JEECG BOOT 增加websocket 旨在服务端主动向客户端推送数据，实现系统向在线用户推送消息，可群发，可对指定用户发送

jeecg boot 集成 websocket 步骤

（1）maven依赖

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
```

（2）WebSocket配置类

```
package org.jeecg.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.socket.server.standard.ServerEndpointExporter;

@Configuration
public class WebSocketConfig {
    /**
     *  注入ServerEndpointExporter,
     *  这个bean会自动注册使用了@ServerEndpoint注解声明的Websocket endpoint
     */
    @Bean
    public ServerEndpointExporter serverEndpointExporter() {
        return new ServerEndpointExporter();
    }

}

```

(3) WebSocket操作类

> 通过该类WebSocket可以进行群推送以及单点推送

```
package org.jeecg.modules.message.websocket;

import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.CopyOnWriteArraySet;

import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.PathParam;
import javax.websocket.server.ServerEndpoint;

import org.springframework.stereotype.Component;

import lombok.extern.slf4j.Slf4j;

@Component
@Slf4j
@ServerEndpoint("/websocket/{userId}")
public class WebSocket {

    private Session session;

    private static CopyOnWriteArraySet<WebSocket> webSockets =new CopyOnWriteArraySet<>();
    private static Map<String,Session> sessionPool = new HashMap<String,Session>();
    @OnOpen
    public void onOpen(Session session, @PathParam(value="userId")String userId) {
        try {
            this.session = session;
            webSockets.add(this);
            sessionPool.put(userId, session);
            log.info("【websocket消息】有新的连接，总数为:"+webSockets.size());
        } catch (Exception e) {
        }
    }

    @OnClose
    public void onClose() {
        try {
            webSockets.remove(this);
            log.info("【websocket消息】连接断开，总数为:"+webSockets.size());
        } catch (Exception e) {
        }
    }

    @OnMessage
    public void onMessage(String message) {
        log.info("【websocket消息】收到客户端消息:"+message);
    }

    // 此为广播消息
    public void sendAllMessage(String message) {
        log.info("【websocket消息】广播消息:"+message);
        for(WebSocket webSocket : webSockets) {
            try {
                if(webSocket.session.isOpen()) {
                    webSocket.session.getAsyncRemote().sendText(message);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    // 此为单点消息
    public void sendOneMessage(String userId, String message) {
        Session session = sessionPool.get(userId);
        if (session != null&&session.isOpen()) {
            try {
                log.info("【websocket消息】 单点消息:"+message);
                session.getAsyncRemote().sendText(message);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    // 此为单点消息(多人)
    public void sendMoreMessage(String[] userIds, String message) {
        for(String userId:userIds) {
            Session session = sessionPool.get(userId);
            if (session != null&&session.isOpen()) {
                try {
                    log.info("【websocket消息】 单点消息:"+message);
                    session.getAsyncRemote().sendText(message);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }

    }

}
```

前端中VUE使用WebSocket

```
<script>
    import store from '@/store/'

    export default {
        data() {
            return {
            }
        },
        mounted() {
              //初始化websocket
              this.initWebSocket()
        },
        destroyed: function () { // 离开页面生命周期函数
              this.websocketclose();
        },
        methods: {
            initWebSocket: function () {
                // WebSocket与普通的请求所用协议有所不同，ws等同于http，wss等同于https
                var userId = store.getters.userInfo.id;
                var url = window._CONFIG['domianURL'].replace("https://","ws://").replace("http://","w
                this.websock = new WebSocket(url);
                this.websock.onopen = this.websocketonopen;
                this.websock.onerror = this.websocketonerror;
                this.websock.onmessage = this.websocketonmessage;
                this.websock.onclose = this.websocketclose;
              },
              websocketonopen: function () {
                console.log("WebSocket连接成功");
              },
              websocketonerror: function (e) {
                console.log("WebSocket连接发生错误");
              },
              websocketonmessage: function (e) {
                var data = eval("(" + e.data + ")");
                 //处理订阅信息
                if(data.cmd == "topic"){
                   //TODO 系统通知

                }else if(data.cmd == "user"){
                   //TODO 用户消息

                }
              },
              websocketclose: function (e) {
                console.log("connection closed (" + e.code + ")");
              }
        }
    }
</script>
```