参见git ch7_6

1.什么是websocket

websocket是为浏览器和服务端提供的全双工异步通信的能力.

websocket通过一个socket来实现双工异步通信,直接使用websocket协议开发程序繁琐,我们会使用它的子协议STOMP,STOMP协议使用一个基于帧的格式来定义消息,与http的request和response类似(具有@RequesetMapping的@MessageMapping),

2. 使用

2.1 广播式

广播式即服务端有消息时,会将消息发送给所有连接了当前的endpoint的浏览器.

(1) 配置WebSocket,需要在配置类上使用@EnableWebSocketMessageBroker开启对WebSocket的支持,并用过实现WebSocketMessageBrokerConfigurer接口,来配置WebSocket.

```
@Configuration
@EnableWebSocketMessageBroker//1
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
   @Override
   public void registerStompEndpoints(StompEndpointRegistry registry) { //2
      registry.addEndpoint("/endpointWisely").withSockJS();
      registry.addEndpoint("/endpointChat").withSockJS();//3
   }

   @Override
   public void configureMessageBroker(MessageBrokerRegistry registry) {//4
      registry.enableSimpleBroker("/queue","/topic"); //5
   }
}
```

#1 注解开启使用STOMP协议来传输基于代理的消息,这时控制器支持使用注解@MessageMapping
#2 注册STOMP协议的节点(endpoint),并映射的指定URL
#3 注册一个STOMP的endpoint,并指定使用SockJS协议
#4 配置消息代理
#5 广播式配置一个/topic消息代理

控制器:

```
@Controller
public class WsController {
   @MessageMapping("/welcome")    //1
   @SendTo("/topic/getResponse")  //2
   public WiselyResponse say(WiselyMessage message) throws
InterruptedException {
```

```
        Thread.sleep(3000);
        return new WiselyResponse("Welcome, "+ message.getName() + "!");
    }
}
```

#1 当浏览器向服务端发送请求时,通过@MessageMapping映射/welcome这个地址
#2 当服务端有消息时,会对订阅了@SendTo中的路径的浏览器发送消息
前端页面：参见ch7_6

2.2 点对点

解决消息由谁发送, 消息由谁接收的问题

```
 @Autowired
private SimpMessagingTemplate messagingTemplate; //1

@MessageMapping("/chat")
public void handleChat(Principal principal, String msg) { //2
    if (principal.getName().equals("wyf")) {//3
        messagingTemplate.convertAndSendToUser("wisely",
                "/queue/notifications", principal.getName() + "-send:"
                        + msg);//4
    } else {
        messagingTemplate.convertAndSendToUser("wyf",
                "/queue/notifications", principal.getName() + "-send:"
                        + msg);
    }
}
```

#1 通过SimpMessagingTemplate向浏览器发送消息
#2 在Spring mvc中,可以直接在参数中获得principal,principal中包含当前用户的信息
#3 硬编码,实际项目中需要根据实际情况开发
#4 通过messagingTemplate.convertAndSendToUser向用户发送消息,第一个参数是接收消息的用户,
第二个是浏览器订阅的地址,第三个是消息本身