채팅방을 위한 WebSocket

https://velog.io/@jyyoun1022/Spring-Web-Socket웹-소켓과-Chatting채팅-2

TIP

sockjs

spring에서 위와 같은 브라우저 문제를 해결하기 위한 방법으로 sockjs를 해결책으로 제공한다. 서버 개발시 일반 웹소켓으로 통신할지 sockjs 호환으로 통신할지 결정할 수 있다. 클라이언트는 sockjs를 통해 서버랑 통신한다.

SockJS는 WebSocket을 지원하지 않는 브라우저에서도 실시간 양방향 통신을 가능하게 하는 JavaScript 라이브러리입니다.

stomp

stomp는 단순 텍스트 지향 메시징 프로토콜이다. spring에 종속적이고, 구독방식으로 사용하고 있다. (가벼워서 많이 사용한다.)

stomp는 웹소켓 위에서 동작하는 프로토콜로써, 클라이언트와 서버가 전송할 메시지 유형, 형식, 내용들을 정의하는 매커니즘이다.

node를 이용할땐 socket.io를 주로 사용하고, spring을 사용할땐 stomp, sockjs를 주로 사용한다.

Need BE

ChatConfig

- `registerStompEndpoints` Override
  - Client에서 websocket연결할 때 사용할 API 경로를 설정해주는 메서드
  - 새로운 핸드셰이크 커넥션을 생성할 때 사용됨
- `configureMessageBroker`
  - `enableSimpleBroker`
    - 메시지 받을 때 관련 경로 설정

        - `"/queue", "/topic"` 이 두 경로가 prefix(api 경로 맨 앞)에 붙은 경우, messageBroker가 잡아서 해당 채팅방을 구독하고 있는 클라이언트에게 메시지를 전달해줌
    - `setApplicationDestinationPrefixes`
        - 메시지 보낼 때 관련 경로 설정
- 도메인이 다른 서버에서도 접속 가능하도록 CORS : setAllowedOprigins(" * ");를 추가해줘야됨.

ChatMessage

- 메시지 타입, 방 번호, 채팅을 보낸 사람, 메세지, 채팅 발송 시간

ChatRoom - (Session으로 이용할것으로 보임)

- HashSet 형태의 Session이 필요하다.⇒ 클라이언트별로 세션을 갖고 있어야한다.

WebSocketHandler

- 웹 소켓 클라이언트로부터 채팅 메세지를 전달받아 채팅 메세지를 객체로 변환
- 전달받은 메시지에 담긴 채팅방 Id 로 발송 대상 채팅방 정보를 조회
- 해당 채팅방에 입장해 있는 모든 클라이언트(Websocket Session) 에게 타입에 따른 메시지 발송
  (중요할듯)
- `afterConnectionEstablished`, `afterConnectionClosed`, `handleMessage` 필요
- `handleMessage`
    - 클라이언트로부터 수신된 WebSocket 메시지를 처리하는 메소드
- `afterConnectionEstablished`
    - 접속됐을때
- `afterConnectionClosed`
    - 접속끊겼을때

React

```
import SockJs from "sockjs-client";
import StompJs from "stompjs";
//stomp와 sockjs 패키지로 깔고 임포트!!

const sock = new SockJs("http://서버주소");
//client 객체 생성 및 서버주소 입력

const stomp = StompJs.over(sock);
//stomp로 감싸기

const stompConnect = () => {
    try {
      stomp.debug = null;
      //웹소켓 연결시 stomp에서 자동으로 connect이 되었다는것을
      //console에 보여주는데 그것을 감추기 위한 debug

      stomp.connect(token, () => {
        stomp.subscribe(
          `서버주소`,
          (data) => {
            const newMessage = JSON.parse(data.body);
            //데이터 파싱
          },
          token
        );
      });
    } catch (err) {

    }
  };

//웹소켓 connect-subscribe 부분

const stompDisConnect = () => {
    try {
      stomp.debug = null;
      stomp.disconnect(() => {
        stomp.unsubscribe("sub-0");
      }, token);
    } catch (err) {

    }
  };
//웹소켓 disconnect-unsubscribe 부분
// 웹소켓을 disconnect을 따로 해주지 않으면 계속 연결되어 있어서 사용하지 않을때는 꼭 연결을 끊어주어야한
다.

const SendMessage = () => {
    stomp.debug = null;
    const data = {
      type: "TALK",
      roomId: roomId,
      sender: sender_nick,
      message: message,
      createdAt: now,
    };
```

```
  //예시 - 데이터 보낼때 json형식을 맞추어 보낸다.
    stomp.send("/pub/chat/message", token, JSON.stringify(data));
  };
//웹소켓 데이터 전송 부분
```