

Chatting Web

Personal project

CHOI WONIK

Process



01
WebSocket

Web Socket을
통해
메세지 전송

02
Producer

kafka 토픽에
브로커로 전송된
메시지 저장

03
Consumer

kafka 토픽에
저장된 메시지를
가져옴

04
메시지 확인

구독한
WebSocket
주소에서 메시지
확인

00

Docker Kafka 설정

Docker container 설정

- **Kafka** 클러스터를 관리하고,
- 브로커 간의 조정을 담당하는 **Zookeeper** 를 설정
- Kafka 브로커 설정을 포함하고 있으며, Zookeeper 에 의존하는 **Kafka** 서비스를 설정

```
C:\Users\whhwwh>docker ps -a
CONTAINER ID        IMAGE
0f6da7f585c0       confluentinc/cp-kafka:latest
acf1b6f14b1a       confluentinc/cp-zookeeper:latest
```

```
version: '3'
services:
  zookeeper:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000

  kafka:
    image: confluentinc/cp-kafka:latest
    container_name: kafka
    ports:
      - "9092:9092"
    depends_on:
      - zookeeper
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: 'zookeeper:2181'
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_INTERNAL:
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092,PLAINTEXT_INTERNAL://k
      KAFKA_AUTO_CREATE_TOPICS_ENABLE: "true"
      KAFKA_DELETE_RETENTION_MS: 86400000 # 1일 후 데이터 삭제(24*60*60*1000)
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
      KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
      KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
```

01

WebSocket 단계

WebSocket 설정

- sub로 시작되는 주소에
- 요청을 구독한 모든 사용자에서 메시지 전달
- pub로 시작되는 경로에 메시지 발행



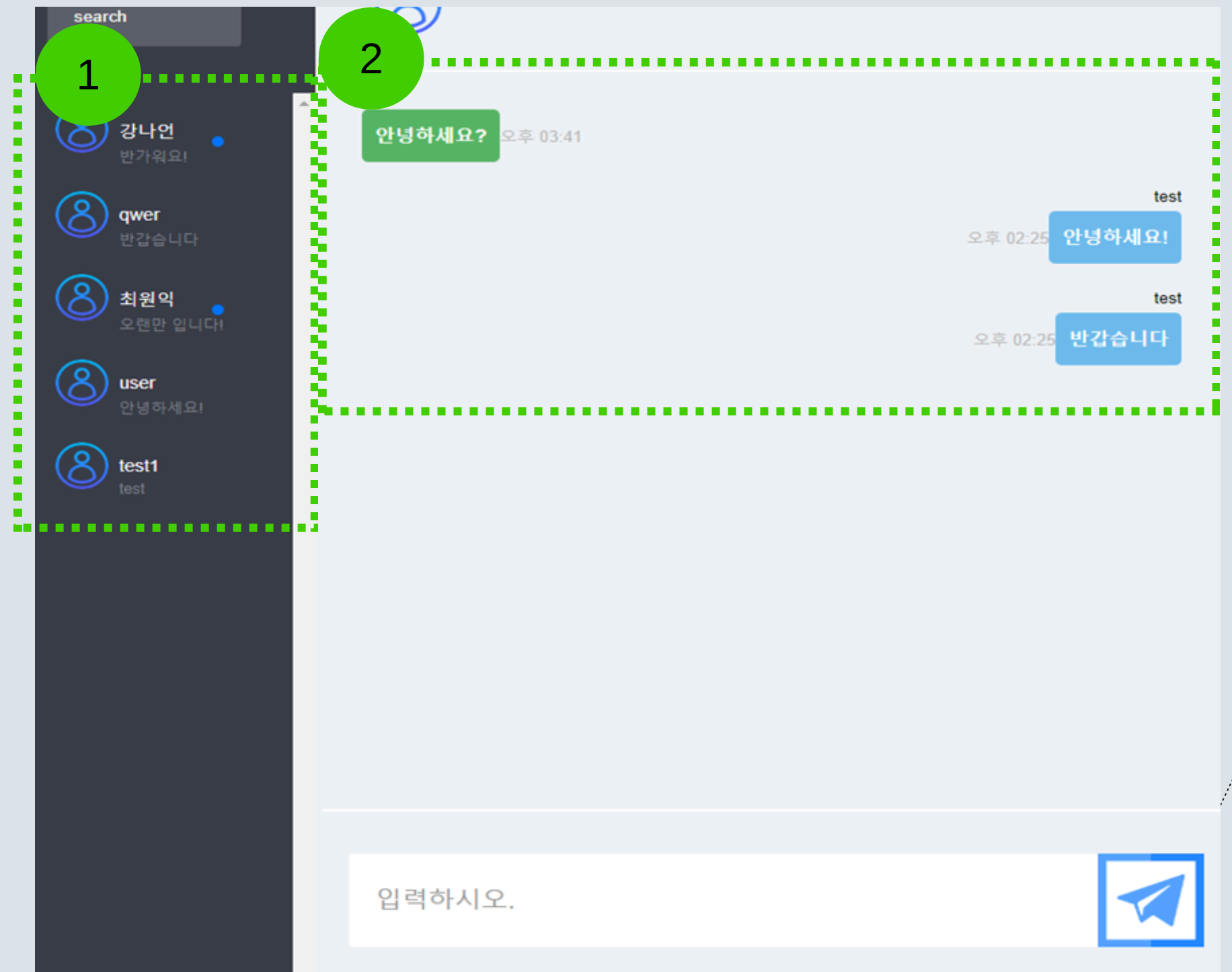
과정 1 Side 채팅

- WebSocket 주소 : `"/sub/chat/side/"+meNum`
meNum'은 member entity에 저장된 내 고유 id로 지정
- `"/sub/chat/side/"+meNum` 주소를 구독하여 메시지를 실시간으로 전달 받음.
- 아직 확인 하지 않은 메시지를 '·'으로 나타냄.

과정 2 Main 채팅

- WebSocket 주소 : `"/sub/chat/room/"+roomId`
- 'roomId'는 member entity에 저장된 내 고유 id와 상대의 고유 id를 섞어 지정 ex) 3&4
- `"/sub/chat/room/"+roomId` 주소를 구독하여 메시지를 실시간으로 전달 받음.

```
@Override
public void configureMessageBroker(MessageBrokerRegistry registry) {
    // 구독한 주소로 메시지를 보낸다.
    registry.setApplicationDestinationPrefixes("/pub");
    // 구독한 주소에서 메시지를 받는다.
    registry.enableSimpleBroker("/sub");
}
```



01

WebSocket 단계

과정 1 서버로 전송

- 주소 : “/pub/chat/massage”
- WebSocket을 통해 클라이언트에서 메시지를 서버로 전송



과정 2 Kafka로 전송

- JSON형식의 메시지를 ChatMessageDTO 객체로 변환
- 서버에서 받은 메시지 kafka로 전송

1

```
BTN=>{
  let $msg = document.getElementById('msg');
  if($msg.value!=""){
    console.log(me + ":" + $msg.value);
    stomp.send(
      '/pub/chat/message',
      {},
      JSON.stringify({roomId: roomId, message: $msg.value, writer: me})
    );
    $msg.value = '';
  }

  $msg.focus();
  return false;
}
```

2

```
@Autowired
private KafkaTemplate<String, ChatMessageDTO> kafkaTemplate;

@MessageMapping("chat/message")
public void sendMessage(@PathVariable("roomId") String roomId ,@RequestBody ChatMessageDTO message) {
    try {
        kafkaTemplate.send(KafkaConstants.KAFKA_TOPIC, message).get();
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}

public class ChatMessageDTO implements Serializable{
    private static final long serialVersionUID = 1L;

    private String roomId;
    private String writer;
    private String message;
    private String timestamp;
}
```


02

Producer 단계

kafka 전송 확인

- **kafka topic**에 저장된 내용을 console로 나타내어 제대로 전송 되었는지 확인

브로커에서 메시지 전달

- topic : kafka-chat
- **Kafka** 토픽에서 읽어들이는 메시지를 처리하는 콜백 메서드
- Kafka에서 받은 메시지를 **ChatMessageDTO**에서 역직렬화 하여 전달

```
C:\Users\whwhh>docker exec -it kafka /bin/bash
[appuser@0f6da7f585c0 ~]$ kafka-console-consumer --bootstrap-server localhost:9092 --t
{"roomId":"3&1","writer":"qwer","message":"안녕하세요?","timestamp":null}
{"roomId":"4&1","writer":"user","message":"안녕하세요!","timestamp":null}
{"roomId":"4&1","writer":"user","message":"제 이름은 최원익 입니다.","timestamp":null}
{"roomId":"4&1","writer":"user","message":"반가워요!","timestamp":null}
{"roomId":"4&1","writer":"user","message":"반가워요!","timestamp":null}
{"roomId":"4&1","writer":"user","message":"ㅎㅇ여","timestamp":null}
{"roomId":"4&1","writer":"user","message":"ㅎㅇ여","timestamp":null}
{"roomId":"4&1","writer":"user","message":"ㅎㅇ","timestamp":null}
{"roomId":"4&1","writer":"user","message":"ㅎㅇ","timestamp":null}
{"roomId":"4&1","writer":"test","message":"안녕하세요!","timestamp":null}
{"roomId":"4&1","writer":"test","message":"완성된 채팅 웹입니다.","timestamp":null}
{"roomId":"4&1","writer":"user","message":"앞으로 잘 부탁드립니다.","timestamp":null}
{"roomId":"4&1","writer":"test","message":"websocket","timestamp":null}
{"roomId":"4&1","writer":"user","message":"안녕하세요!","timestamp":null}
{"roomId":"5&1","writer":"최원익","message":"오랜만 입니다!","timestamp":null}
{"roomId":"3&1","writer":"test","message":"안녕하세요!","timestamp":null}
{"roomId":"3&1","writer":"test","message":"반갑습니다","timestamp":null}
{"roomId":"6&1","writer":"강나연","message":"반가워요!","timestamp":null}
{"roomId":"3&1","writer":"test","message":"test","timestamp":null}
```

```
@KafkaListener(
    topics = KafkaConstants.KAFKA_TOPIC,
    groupId = KafkaConstants.GROUP_ID
)
public void consume(ChatMessageDTO message) {
```

03

Consumer 단계

과정 1 Chat 내용 db에 저장

- 메시지가 기록된 시간생성
- 채팅 내용을 기록한 db에 kafka 토픽에서 읽어온 메시지 저장

과정 2 메시지 내용 전달

- **ChatMessage**로 내용 역직렬화 된 내용
MessageListener에서 메시지 구독 주소로 발송 준비

과정 3 ChattingRoom db에 저장

- 각 유저간에 처음 채팅일 경우 채팅방 생성
- 유저간 채팅한 내역이 존재할 경우 가장 최근 채팅 기록 업데이트

```

public void consume(ChatMessageDTO message) {
    // a:오전/오후 시간대를 알기쉽게 나타냄
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("a hh:mm");
    // 현재시간
    LocalDateTime time = LocalDateTime.now();
    // 메시지 시간을 문자열로 나타낸 후 messageDTO 셋팅
    message.setTimestamp(time.format(formatter).toString());

    // 메시지 처리 로직을 여기에 작성합니다.
    Chat chat = Chat.messageText(message.getRoomId(), message.getWriter(), message.getMessage(), message.getTimestamp());
    kafkaRepository.save(chat);

    // MessageListener.listen(message);

    // 채팅방에 저장
    String roomId = message.getRoomId().replace("&", "");
    roomId = roomId.replace(String.valueOf(memberRepository.findById(message.getWriter()).get().getId()), "");
    Long talkerNum = Long.parseLong(roomId);
    String tlakerName = memberRepository.findById(talkerNum).get().getName();

    // 채팅방의 메시지를 보낸 사람
    ChattingRoom chattingRoomMe = ChattingRoom.createRoom(message.getRoomId(), message.getWriter(), message.getMessage(), time, 1);
    // 채팅방의 메시지를 보낸 사람
    ChattingRoom chattingRoomYou = ChattingRoom.createRoom(message.getRoomId(), tlakerName, message.getMessage(), time, 0);

    List<ChattingRoom> list = chatRoomRepository.findByRoomId(message.getRoomId());

    // roomId에 해당하는 채팅방이 존재하지 않을 시
    if (list.isEmpty()) {
        // 메시지 보낸 사람으로 db저장
        chatRoomRepository.save(chattingRoomMe);
        // 메시지 보낸 사람으로 db저장
        chatRoomRepository.save(chattingRoomYou);
    }

    // roomId에 해당하는 채팅방에 존재할 시
    else {
        update(list, time, message.getMessage(), message.getWriter());
    }
    System.out.println("Received message: 도대체 뭐가 출력되는 거냐?" + chattingRoomYou);
}

```

04

메세지 확인 단계

메시지 확인

- 메시지를 정해진 주소에서 구독자가 확인 할 수 있도록 전송

```
public class MessageListener {  
  
    @Autowired  
    private SimpMessagingTemplate simpMessagingTemplate;  
  
    public void listen(ChatMessageDTO message) {  
        String roomId = message.getRoomId();  
        int user1 = 0;  
        int user2 = 0;  
  
        // & 이전의 내용을 없애기  
        int index = roomId.indexOf("&");  
        if (index != -1) {  
            user1 = Integer.parseInt(roomId.substring(0, index));  
            user2 = Integer.parseInt(roomId.substring(index + 1));  
        }  
  
        // kafka로 보낸 메시지를 지정주소로 뿌려줌(main채팅)  
        simpMessagingTemplate.convertAndSend("/sub/chat/room/" + message.getRoomId(), message);  
        // side채팅  
        simpMessagingTemplate.convertAndSend("/sub/chat/side/" + user1, message);  
        simpMessagingTemplate.convertAndSend("/sub/chat/side/" + user2, message);  
    }  
}
```

```
// 좌측 대화내용 추가  
stomp.subscribe("/sub/chat/side/" + meNum, (chat) => {  
  
    let content = JSON.parse(chat.body);
```

```
//4. subscribe(path, callback)으로 메세지를 받을 수 있음  
stomp.subscribe("/sub/chat/room/" + roomId, (chat) => {  
  
    let content = JSON.parse(chat.body);
```




Thank you

Contacts

dyd975@naver.com

010 9013 6753

<https://choi-won-ik.github.io/>

