

Message Broker (메시지 브로커)

☰ 태그	
👤 작성자	👤 조성찬

메시지 브로커

메시지 브로커가 메시지를 전달해주는 과정

pub/sub

Simple Message Broker의 문제점

외부 메시지 브로커의 도입

메시지 브로커

메시지 브로커란, Publisher(송신자)로부터 전달받은 메시지를 Subscriber(수신자)에게 전달해주는 중간 역할 수행

메시지 브로커가 메시지를 전달해주는 과정

1. 각각 A, B, C 라는 유저가 차례로 5번방에 입장한다.
2. A가 5번방에서 채팅을 전송한다.
3. 5번방 메시지 브로커(중재자)가 메시지를 받는다.
4. 5번방 메시지 브로커가 5번방 구독자들(A, B, C)에게 메시지를 전송한다.

유저들은 채팅방에 입장함과 동시에 채팅방에 대해 구독(Subscribe)하고, 메시지 브로커는 유저의 구독(Subscribe)정보를 메모리에 유지한다.

```
SUBSCRIBE
destination:/subscribe/chat/room/5

// 5번 채팅방에 구독한다.
```

채팅방 안에 있는 어떤 유저가 메시지를 보내면, 유저가 들어 있는 채팅방의 메시지 브로커가 Subscriber들에게 모두 메시지를 보낸다.

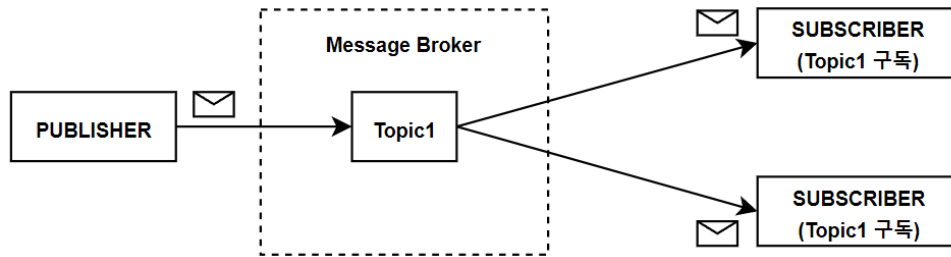
```
SEND
content-type:application/json
destination:/publish/chat

{"chatRoomId":5, "type":"MESSAGE", "writer":"clientB"}

// JSON 형식으로 Body를 작성한다.
```

stomp-specification

STOMP 프로토콜으로 형식을 정의하고 나서, 메시지는 pub/sub 방식으로 Message Broker를 통해 메시지를 보낸다.



pub/sub

메시지를 공급하는 객체(publisher), 소비하는 객체(subscriber)를 분리해서 제공하는 비동기식 메시징 방법

- 채팅, MSA (하나의 서버에서 다른 서버로의 메시지를 API를 통해 전달)에 활용
- publisher나 subscriber가 직접 통신하는 게 아니라 Message Broker를 중간에 거친다. Message Broker는 메시지를 Subscriber에 전달하는 미들웨어 역할을 한다.

Message Broker를 둬서 얻을 수 있는 장점은 여럿 있다.

- 결합을 느슨하게 할 수 있다. 메시지를 공급하는 Publisher는 다른 서비스들에 대해서 알 필요가 없다. Message Broker에게만 메시지를 전달하면 된다.
- 메시지를 버퍼링할 수 있다. 원하는 시점에 Subscriber에게 메시지를 전송할 수 있음

Simple Message Broker의 문제점

Spring에서 제공하는 Stomp와 Simple Message Broker를 이용하면 채팅 서버를 충분히 구현할 수 있지만 스프링 부트 내부 메모리에 채팅 데이터를 저장한다는 한계가 있다. 이 한계는 여러 단점을 불러온다.

- 서버가 꺼지면 Message Broker 안에 있는 데이터는 모두 유실된다.
- 다수의 서버를 만들면 서버 내부 메모리에 채팅 데이터를 저장할 경우, 채팅 데이터를 공유할 수 없다.

⇒ Message Broker를 외부에 두어서 여러 서버가 이 Broker에 접근할 수 있게 한다!

⇒ Redis 또는 RabbitMQ와 같은 외부 전용 메시지 브로커를 사용한다.

외부 메시지 브로커의 도입

위와 같은 문제를 해결하기 위해서 외부 메시지 브로커를 도입한다. Redis는 Stomp 프로토콜을 지원하지 않지만 Pub/Sub 기능을 통해 메시지 브로커로 활용할 수 있다. 물론 RabbitMQ와 같은 전용 메시지 브로커를 사용하면 더 고도화된 기능을 사용할 수 있다.

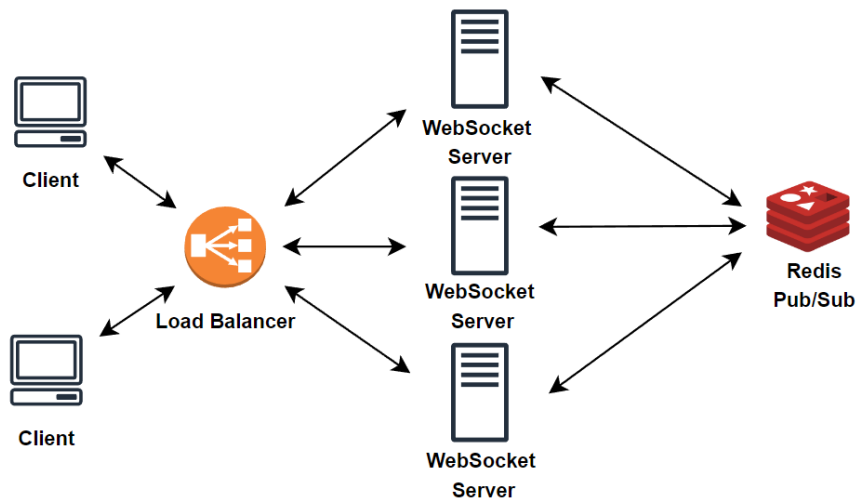
- RabbitMQ와 같은 전용 메시지 브로커를 사용했을 때 얻을 수 있는 장점
 - 메시지 전달 보장
 - 어떻게?

- SSL 지원

▪ 어떻게?

채팅 정보에 대한 세션을 관리할 key-value 데이터베이스가 필요하다.


- 왜? 채팅 정보에 대한 세션은 무엇을 말하는가?
 - HTTP에서 말하는 세션은 쿠키를 통해 전달되는 Session 값을 key값으로 하여 서버에 저장되는 로그인 정보를 뜻한다. 로그인 정보를 알아내어 방문자가 누구인지 알 수 있다. (인증)
 - 세션 서버를 따로 둔다. 사용자가 채팅방을 만들 시 세션 서버에서 채팅방을 생성하고 세션 값을 사용자에게 전달한다. 다른 사용자가 채팅방에 접근 시 해당 채팅 방의 세션 값을 사용자에게 전달한다. 해당 세션 값을 통해 방을 구분하고, 메시지를 전달한다.
 - 세션과 사용자 정보를 통해서 사용자를 인증한다. 침투해 들어온 가짜 사용자를 막는다. 참고 채팅 웹소켓 세션 관리
- WebSocket 연결이 되면 Session이 생성되고, 이 객체를 리스트에 관리하고, 필요하면 꺼내서 메시지를 보낼 수 있다. HttpSession과는 다르다. HttpSession은 브라우저가 Http Request를 할 때마다 쿠키의 세션 키로 구분하여 유저의 정보(로그인 정보)를 가져온다. WebSocket Session은 세션 키로 구분하여 유저의 정보를 저장하거나 가져올 수는 없는 것으로 보인다. 링크



Redis를 활용한 백엔드 시스템 구성을 하기 위해서는 Redis를 설치하거나 Redis를 docker container를 통해 설치해야 한다.

[WebSocket] Spring Boot + STOMP + Redis Pub/Sub 이용한 채팅 서버 구현

WebSocket? WebSocket(웹 소켓)이란 HTTP 환경을 기반으로 하여 TCP/IP 연결을 통해 전이중 통신(양방향 송수신) 채널을 제공하는 컴퓨터 통신 프로토콜이다. WebSocket의 접속 과정은 TCP/IP 접속, 웹 소켓을 열기 위한 HandShake 과정으로 나뉠 수 있다.

 <https://vlog.io/@ohjinseo/WebSocket-Spring-Boot-stomp-Redis-PubSub-이용한-채팅-구현>

