기만콘

websocket with java

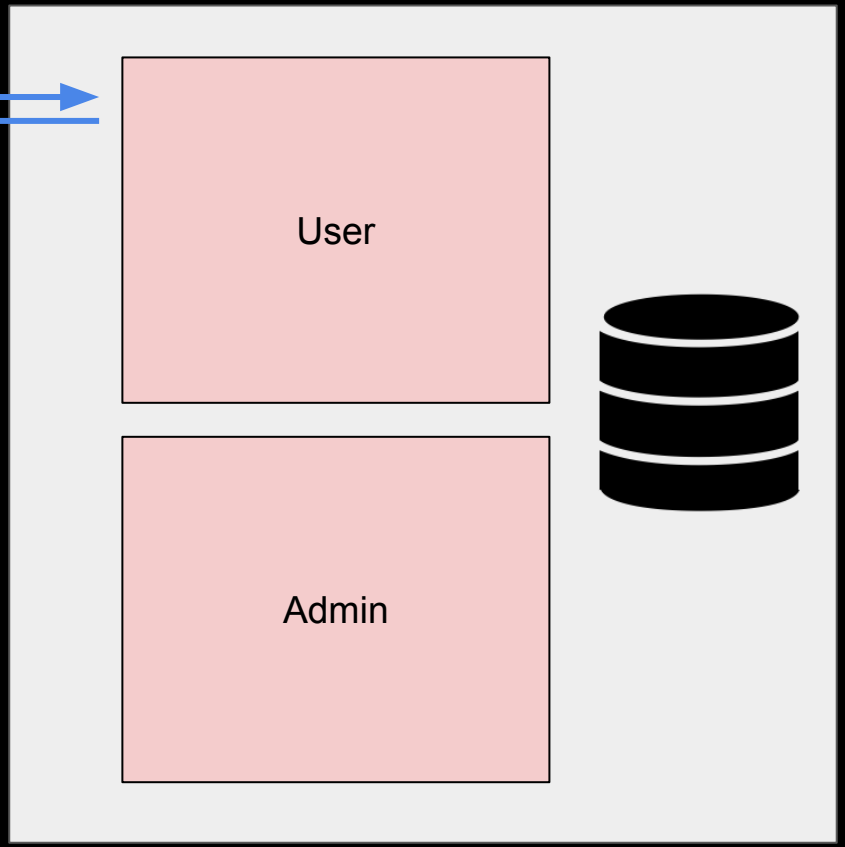
(java라고 쓰고 spring이라고 읽...)

부종민

발표 시작전에 간단한 웹소켓 예제 확인겸
간단한 설문을 진행하겠습니다.

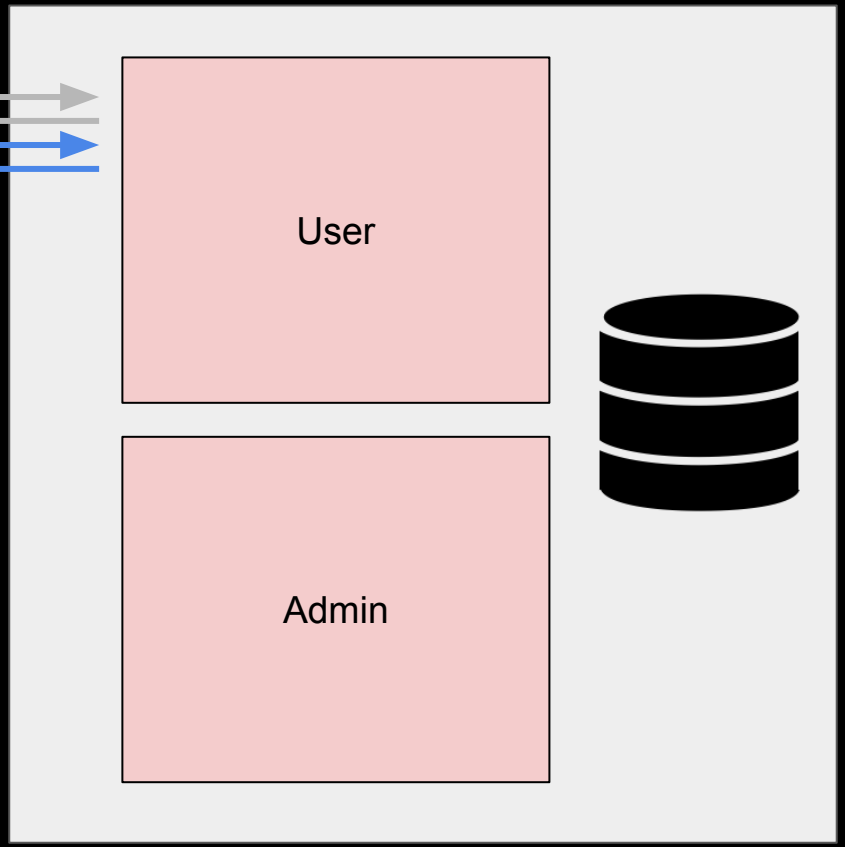

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question


답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
next


User
Admin

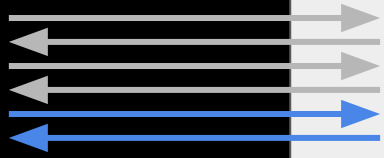

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question

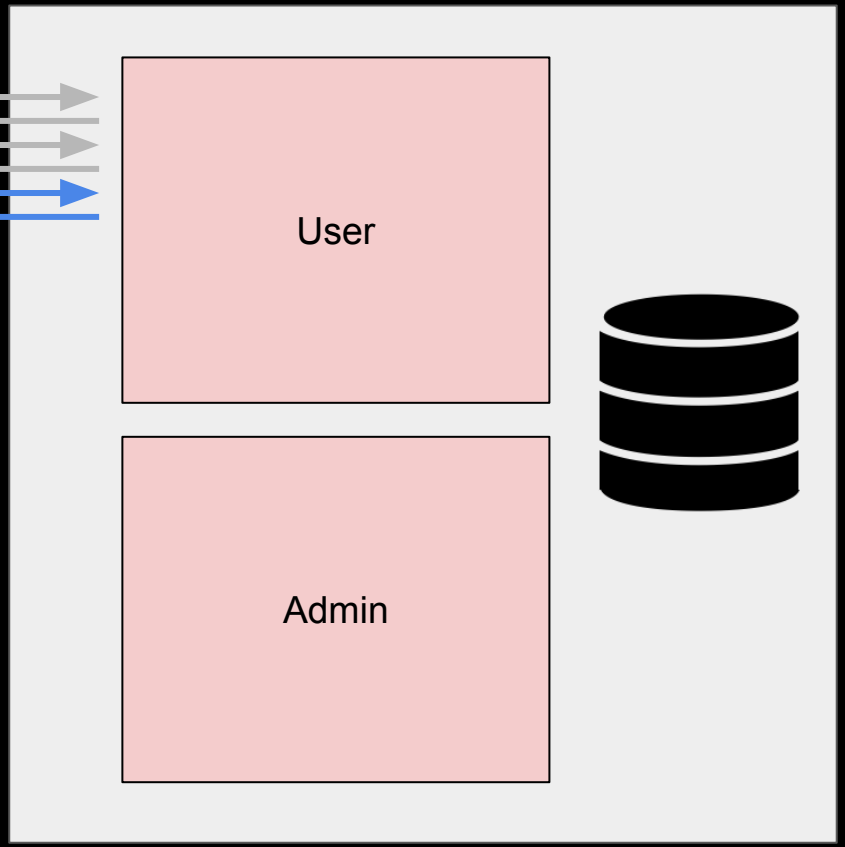

답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
sent

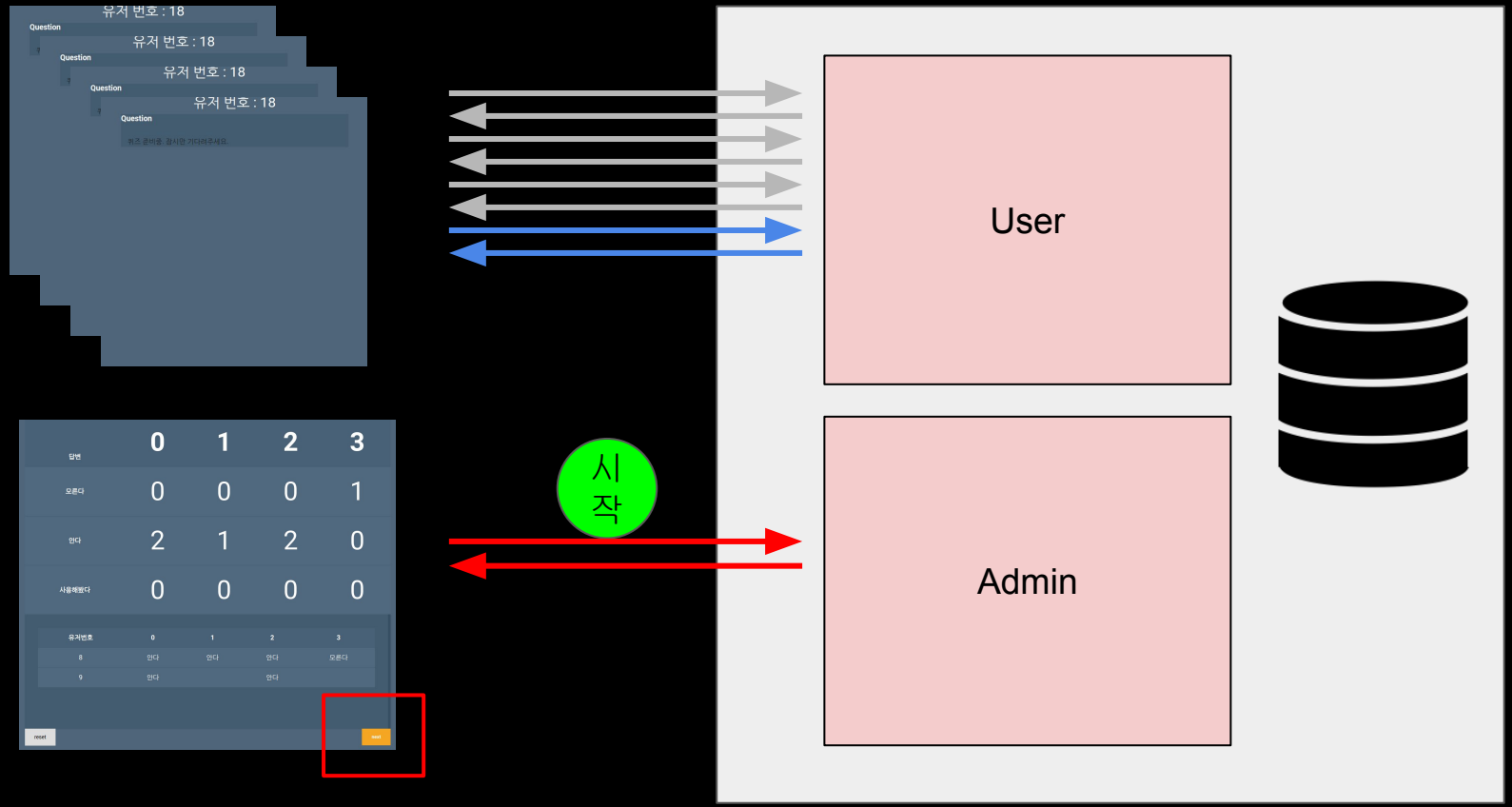

User
Admin

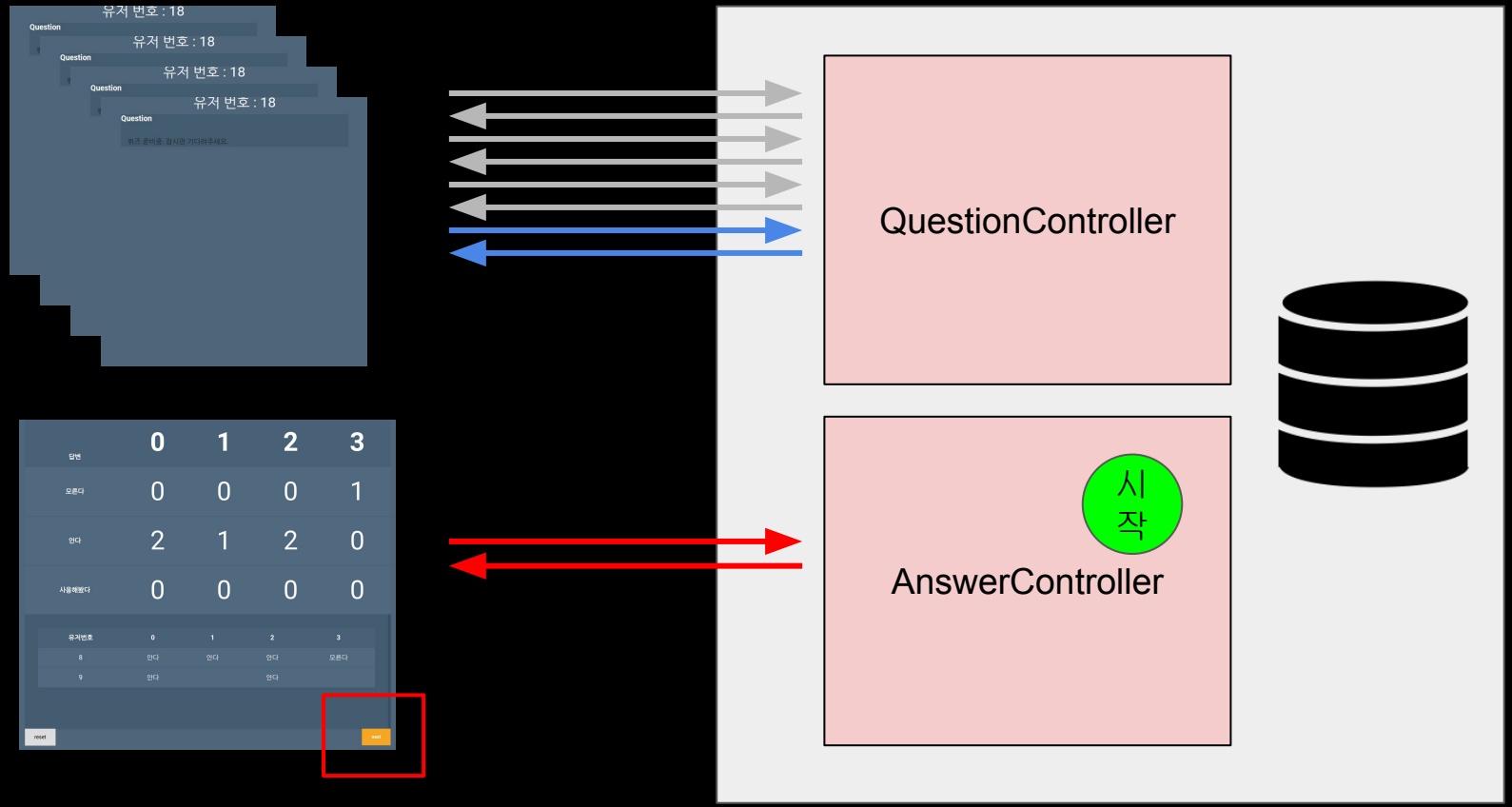

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question


답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
sent

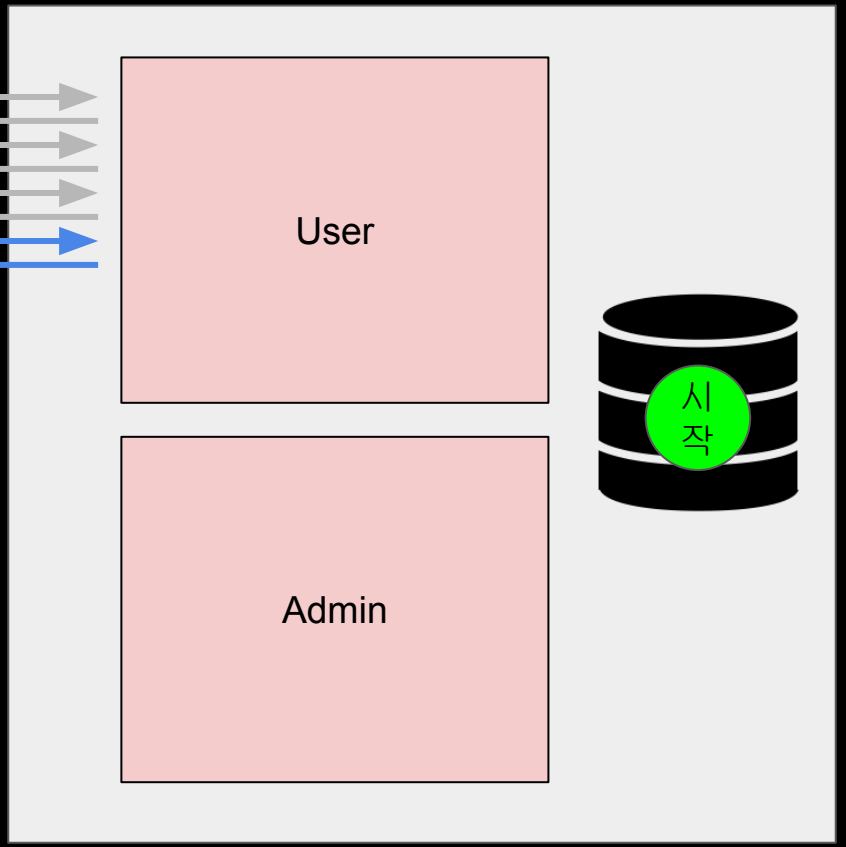



User
Admin


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
퀴즈 준비중 잠시만 기다려주세요.
User
Admin
시
작
0 1 2 3
모른다 0 0 0 1
안다 2 1 2 0
사용해봤다 0 0 0 0
유저번호 0 1 2 3
8 안다 안다 안다 모른다
9 안다 안다
reset
set


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
퀴즈 준비중 잠시만 기다려주세요.
0 1 2 3
답변
모른다 0 0 0 1
안다 2 1 2 0
사용해봤다 0 0 0 0
유저번호 0 1 2 3
8 안다 안다 안다 모른다
9 안다 안다
reset
set
QuestionController
AnswerController
시
작

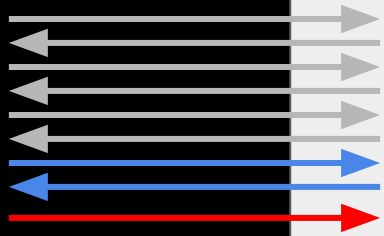

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question

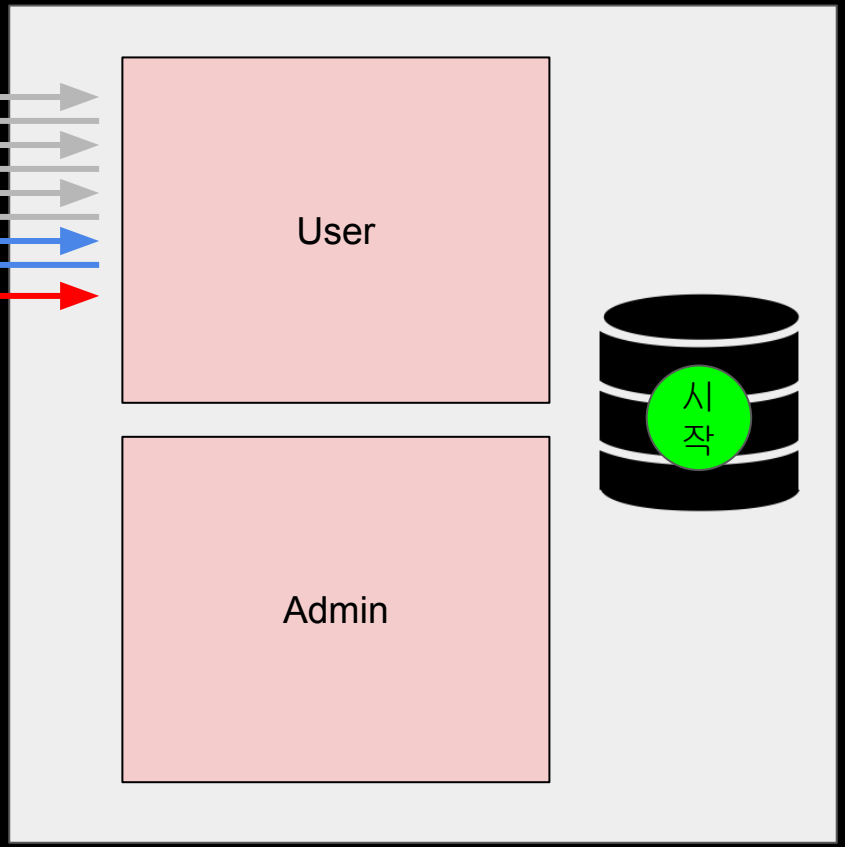

User
Admin
시
작


답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
sest


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question

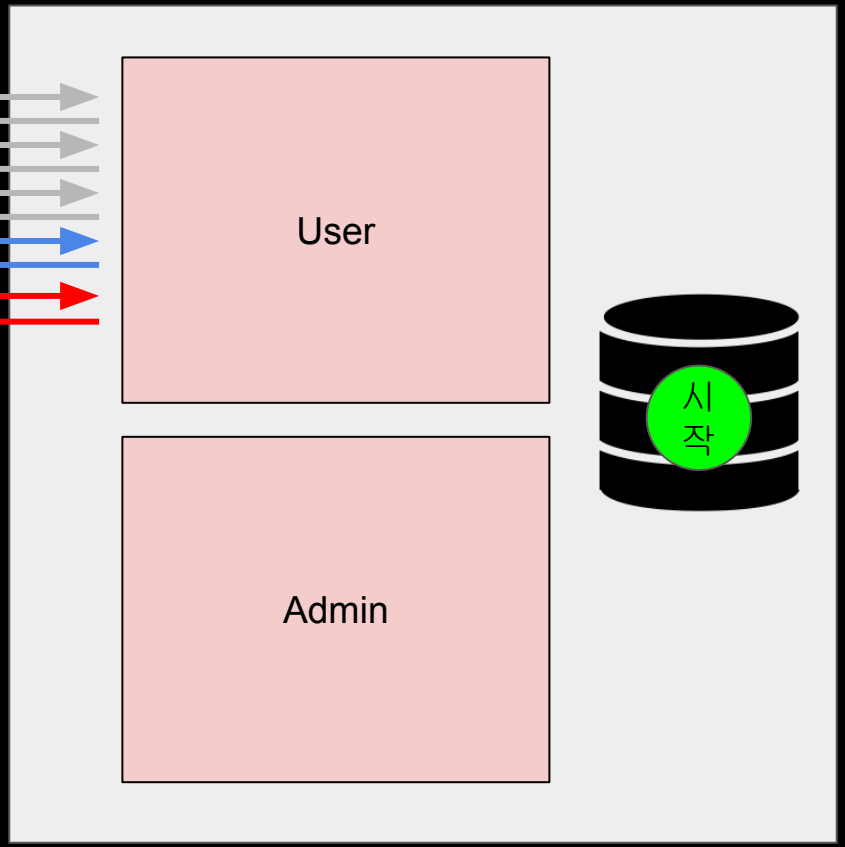

답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
sent




User
Admin
시
작


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
배경지식 설문
HTTP/REST
모른다
안다
사용해봤다
websocket
모른다
안다
사용해봤다

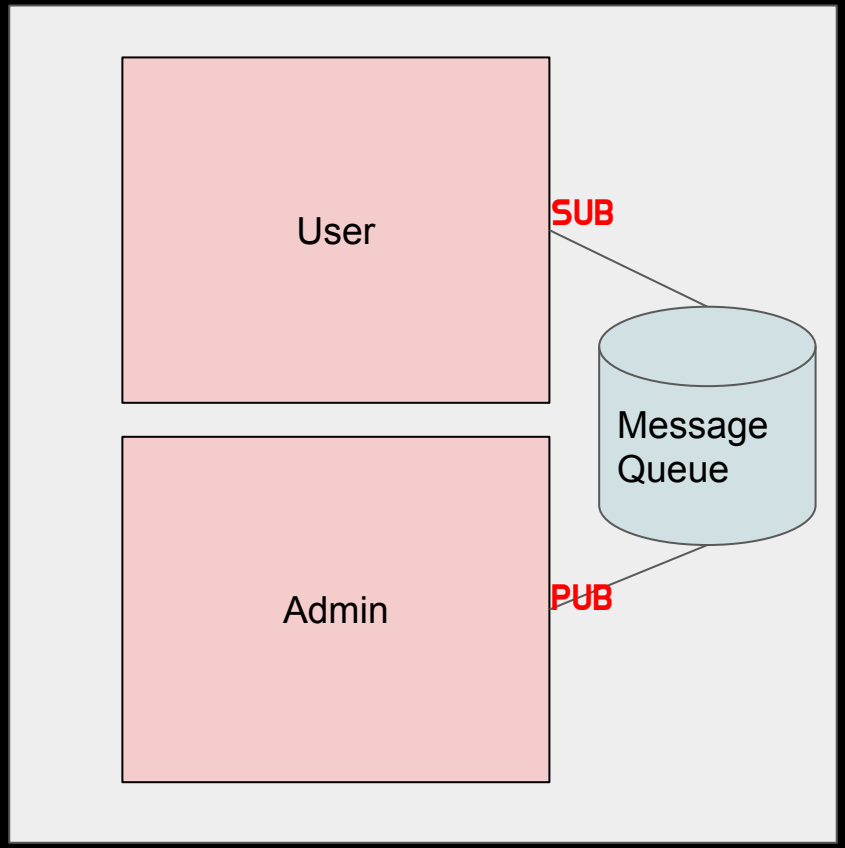

답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
sent

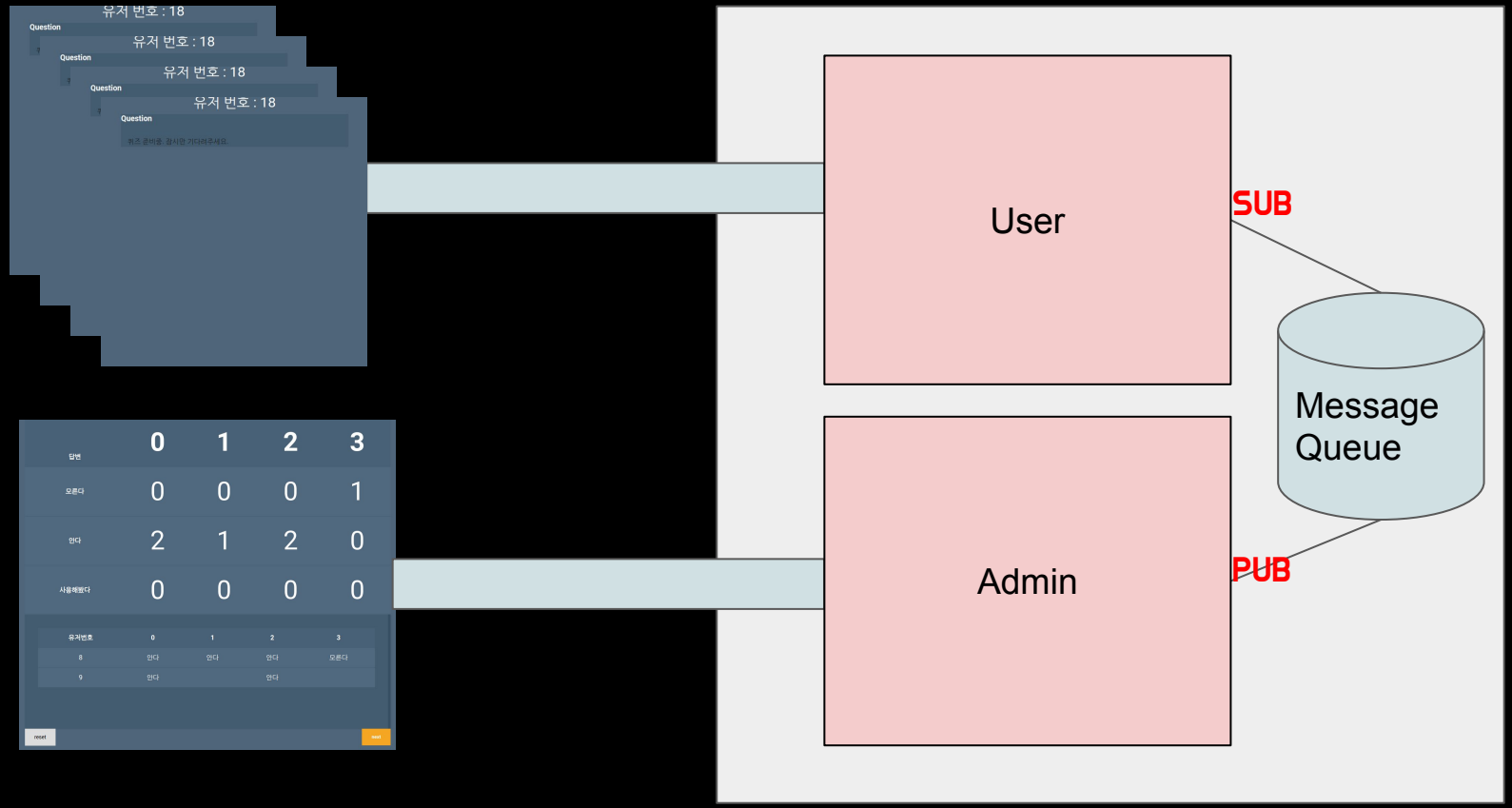

User
Admin
시
작

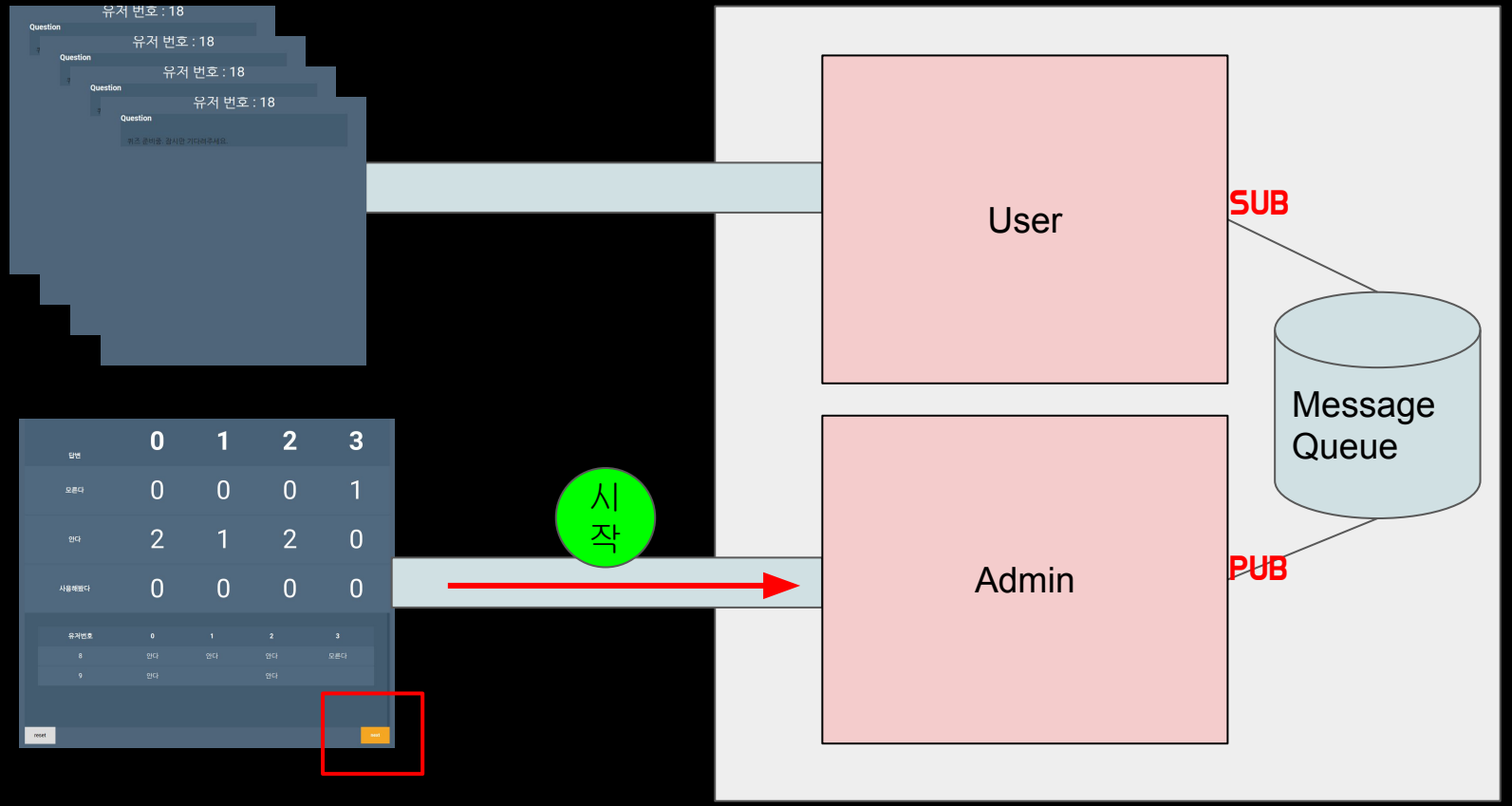
웹소켓으로 한다면?

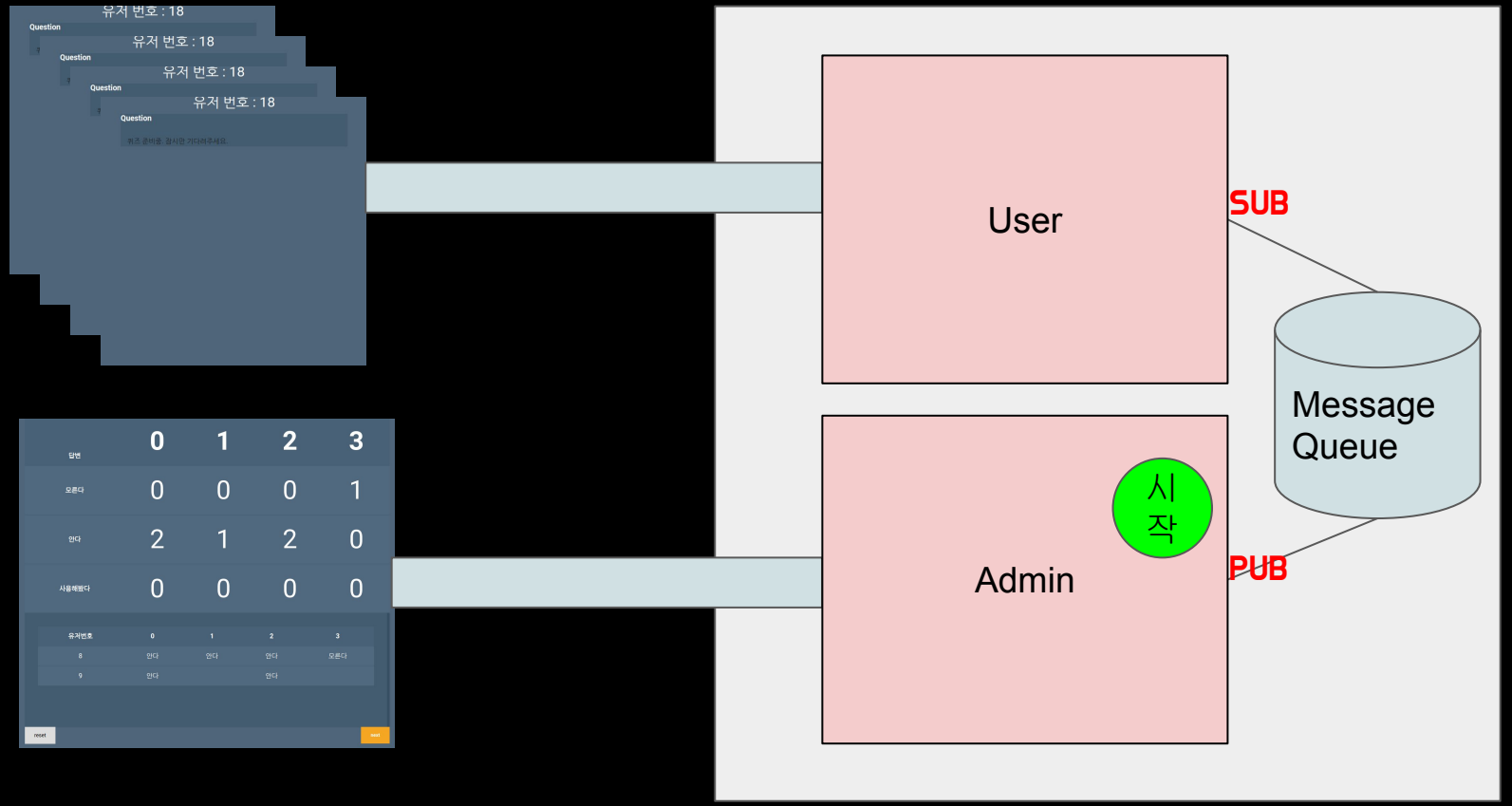

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question

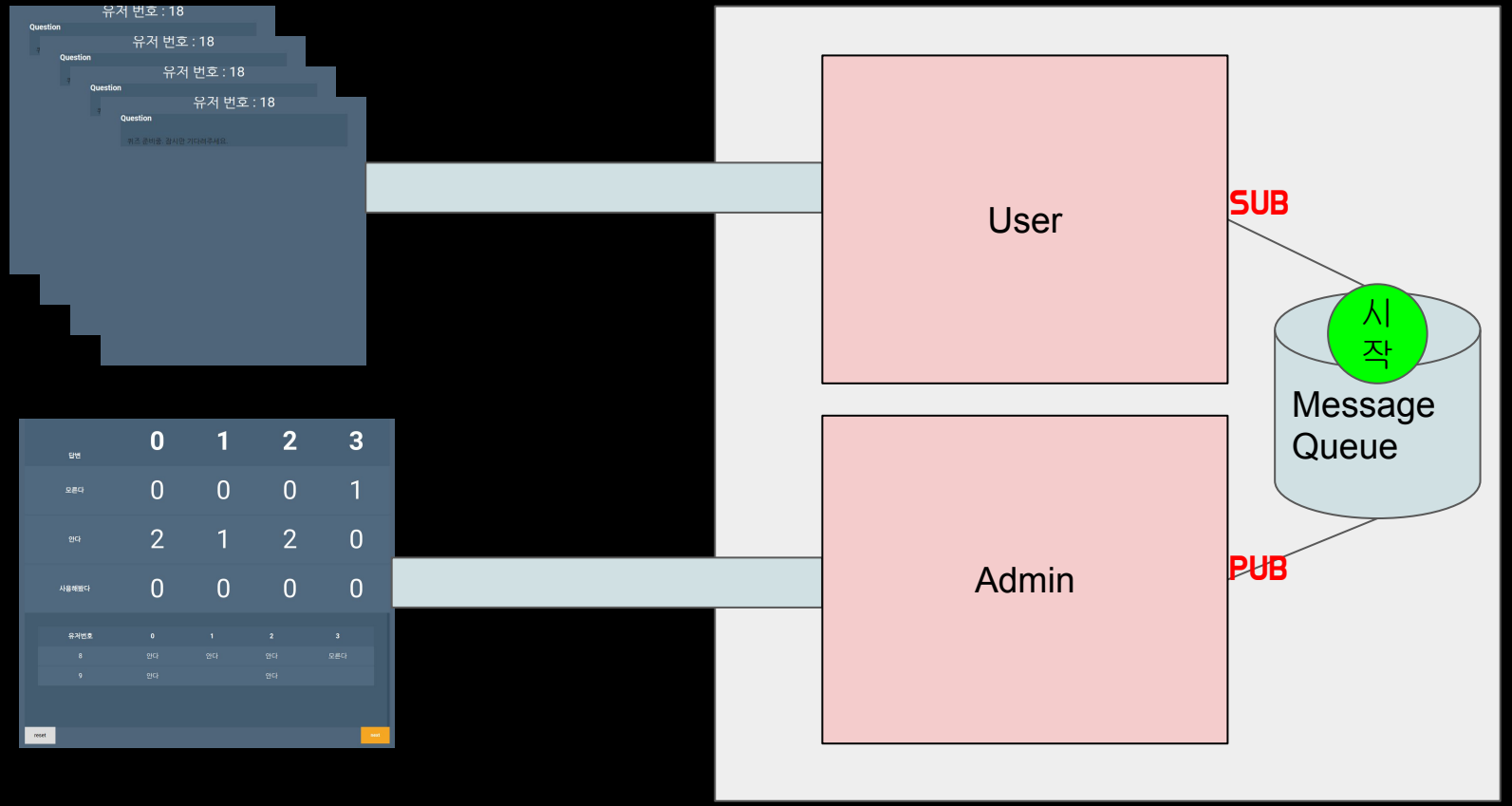

답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
next

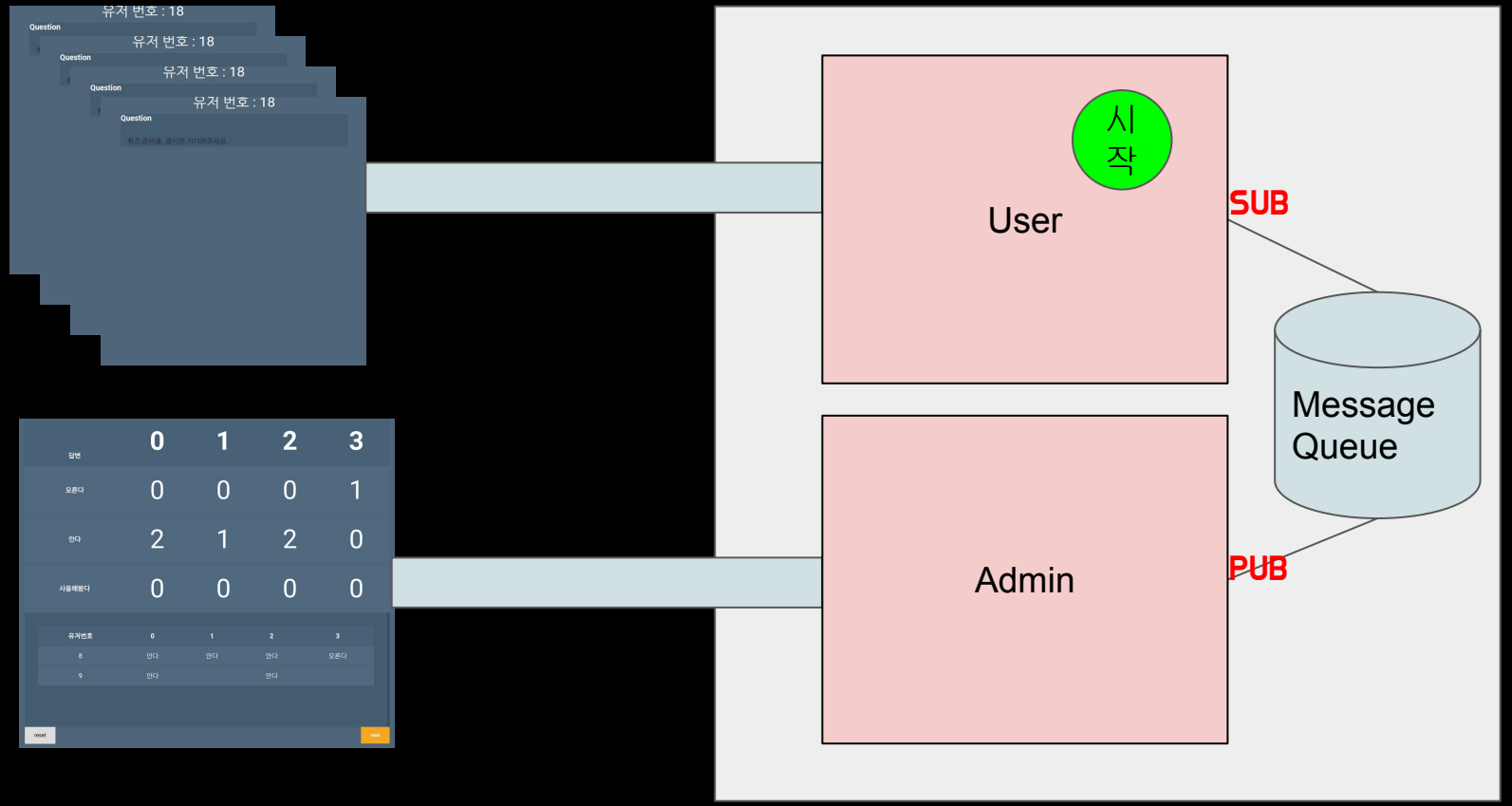

User
SUB
Message
Queue
Admin
PUB

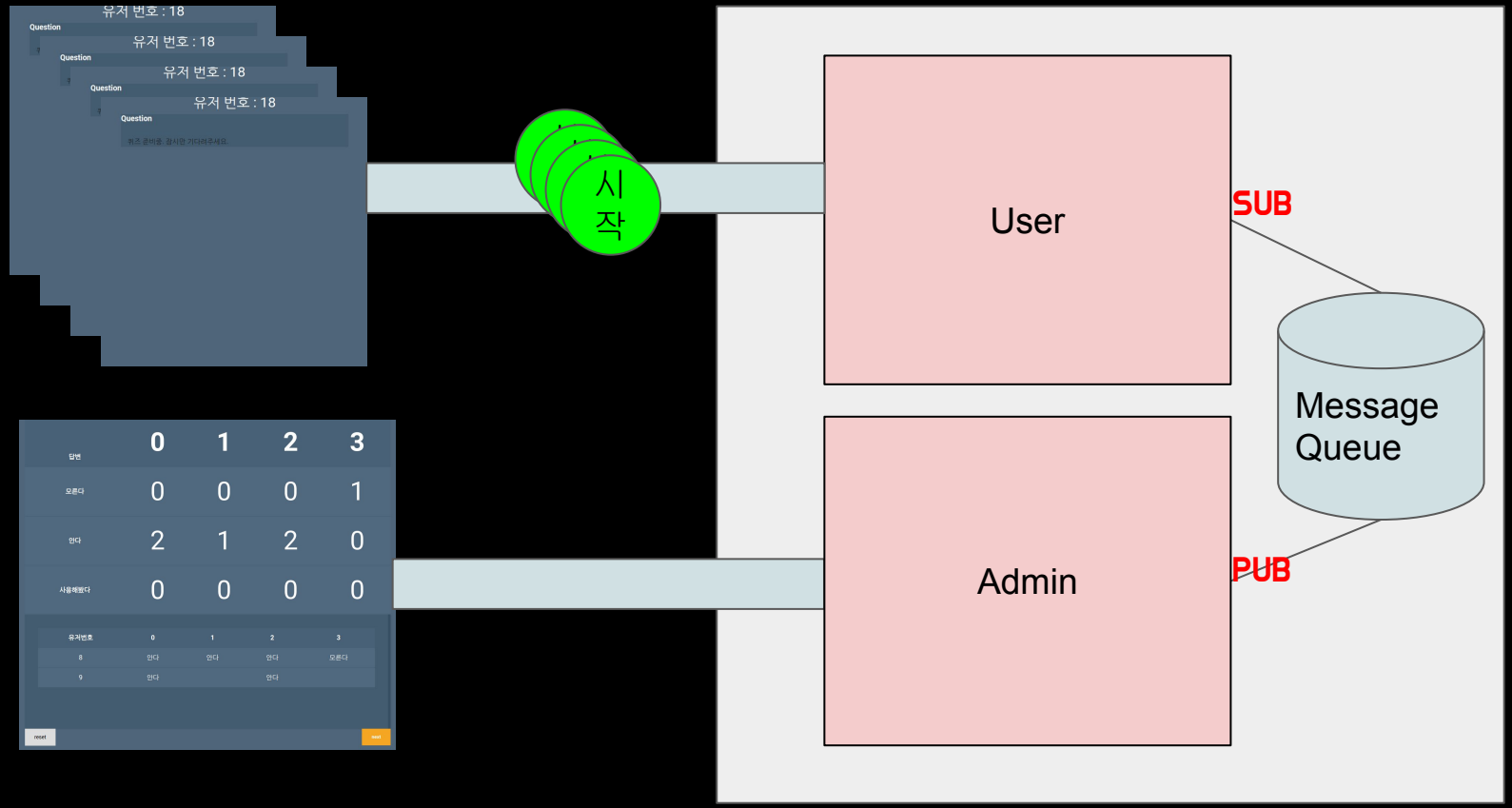

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
퀴즈 준비중 잠시만 기다려주세요.
User
SUB
Message
Queue
Admin
PUB
0 1 2 3
답변
모른다 0 0 0 1
안다 2 1 2 0
사용해봤다 0 0 0 0
유저번호 0 1 2 3
8 안다 안다 안다 모른다
9 안다 안다
reset
sent

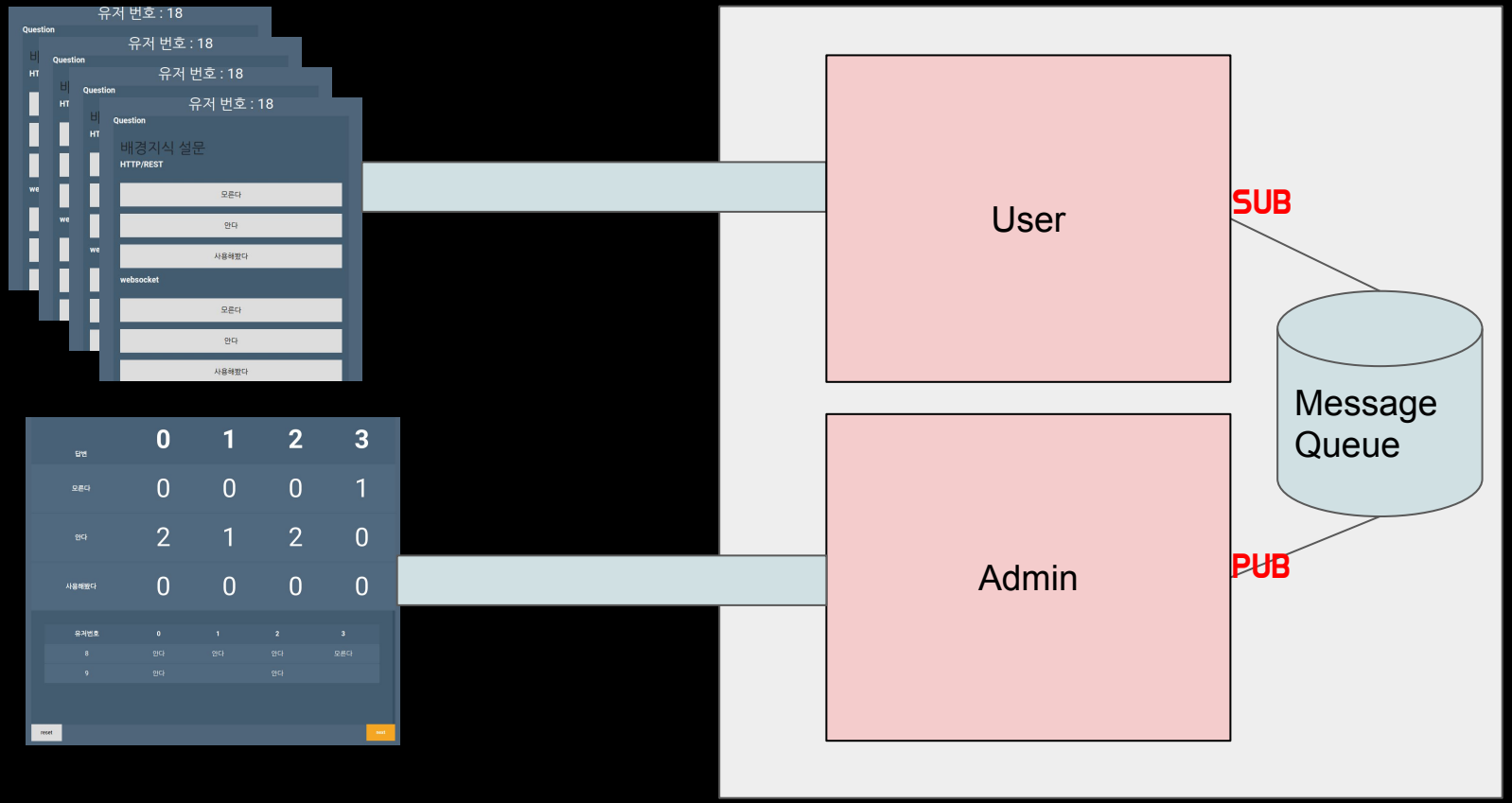

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
퀴즈 준비중 잠시만 기다려주세요.
User
SUB
Message
Queue
0 1 2 3
모른다 0 0 0 1
안다 2 1 2 0
사용해봤다 0 0 0 0
유저번호 0 1 2 3
8 안다 안다 안다 모른다
9 안다 안다
reset
set
시
작
Admin
PUB


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
퀴즈 준비중 잠시만 기다려주세요.
0 1 2 3
답변
모른다 0 0 0 1
안다 2 1 2 0
사용해봤다 0 0 0 0
유저번호 0 1 2 3
8 안다 안다 안다 모른다
9 안다 안다
reset
next
User
Admin
시작
SUB
PUB
Message Queue


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
퀴즈 준비중 잠시만 기다려주세요.
User
SUB
시작
Message Queue
PUB
Admin
0 1 2 3
모른다 0 0 0 1
안다 2 1 2 0
사용해봤다 0 0 0 0
유저번호 0 1 2 3
8 안다 안다 안다 모른다
9 안다 안다
reset
next

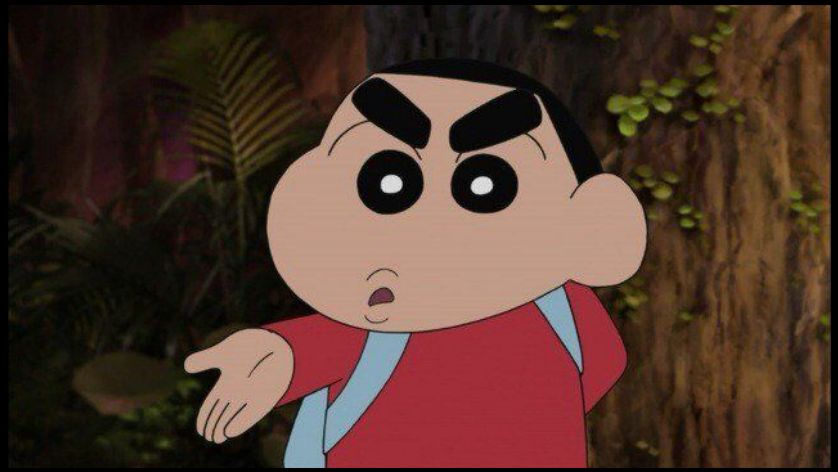

유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
퀴즈 준비중 잠시만 기다려주세요.
0 1 2 3
답변
모른다 0 0 0 1
안다 2 1 2 0
사용해봤다 0 0 0 0
유저번호 0 1 2 3
8 안다 안다 안다 모른다
9 안다 안다
reset
sent
시작
User
SUB
Message Queue
Admin
PUB


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
시
작
User
SUB
Message
Queue
PUB
Admin
0 1 2 3
0 0 0 1
2 1 2 0
0 0 0 0


유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
유저 번호 : 18
Question
배경지식 설문
HTTP/REST
모른다
안다
사용해봤다
websocket
모른다
안다
사용해봤다
User
SUB
Message
Queue
PUB
Admin
답변
0
1
2
3
모른다
0
0
0
1
안다
2
1
2
0
사용해봤다
0
0
0
0
유저번호
0
1
2
3
8
안다
안다
안다
모른다
9
안다
안다
reset
sent

polling과 push의 차이입니다.

polling



push

그럼 웹소켓은?

'web' + 'socket'

web

그냥 웹!!

port	80, 443
프로토콜(http)	ws
SSL(https)	wss
client	javscript socket api를 이용

socket

그냥 소켓!! client

```
public class SocketExample {
    private Socket clientSocket;
    private PrintWriter out;
    private BufferedReader in;

    void start(int port) throws IOException {
        ServerSocket serverSocket = new ServerSocket(port);
        clientSocket = serverSocket.accept();
        out = new PrintWriter(clientSocket.getOutputStream(), autoFlush: true);
        in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

        String inputLine;
        while ((inputLine = in.readLine()) != null) {
            if (".".equals(inputLine)) {
                out.println("good bye");
                break;
            }
            out.println("> " + inputLine);
        }
    }

    public static void main(String[] args) throws IOException {
        new SocketExample().start( port: 8080);
    }
}
```

웹소켓이란?

http는 tcp/ip 기반 비연결지향 프로토콜과 대비

websocket 프로토콜(RFC6455) - **연결지향**(연결을 끊지 않음), **양방향통신**(클라이언트 <-> 서버)

거래소(주식, 암호화폐), 게임등 realtime이 보장되어야하는 상황에서 커넥션비용을 줄이고 불필요한 네트워크 비용을 감소시켜줌.

웹소켓 참조 링크

websocket을 이해하기위해
잠시 프론트 이야기를 하겠습니다.

```
let websocket = new WebSocket("wss://echo.websocket.org");

websocket.onopen = function(event) {
    websocket.send('hello websocket!!')
}

websocket.onmessage = function(event) {
    console.log('recieved message >' + event.data)
}

```

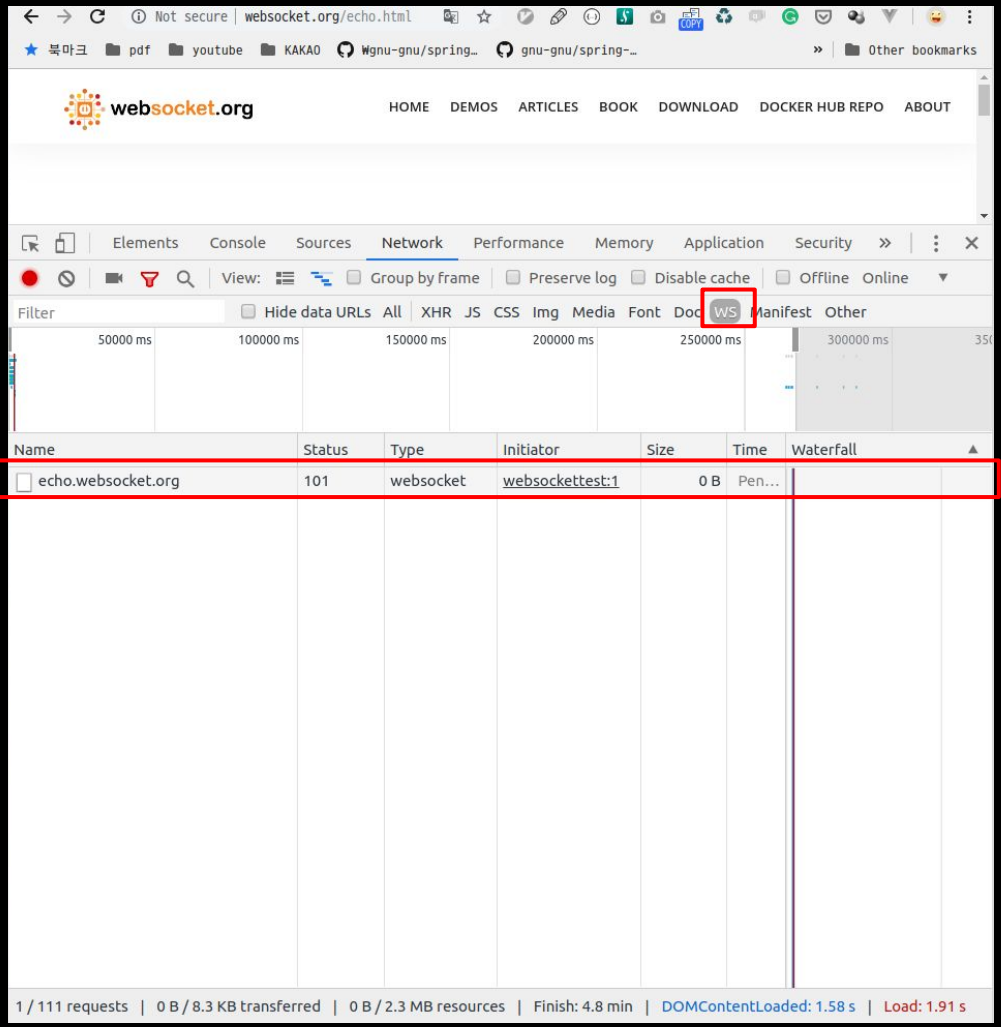
handshake - http -> websocket 서버와 합의

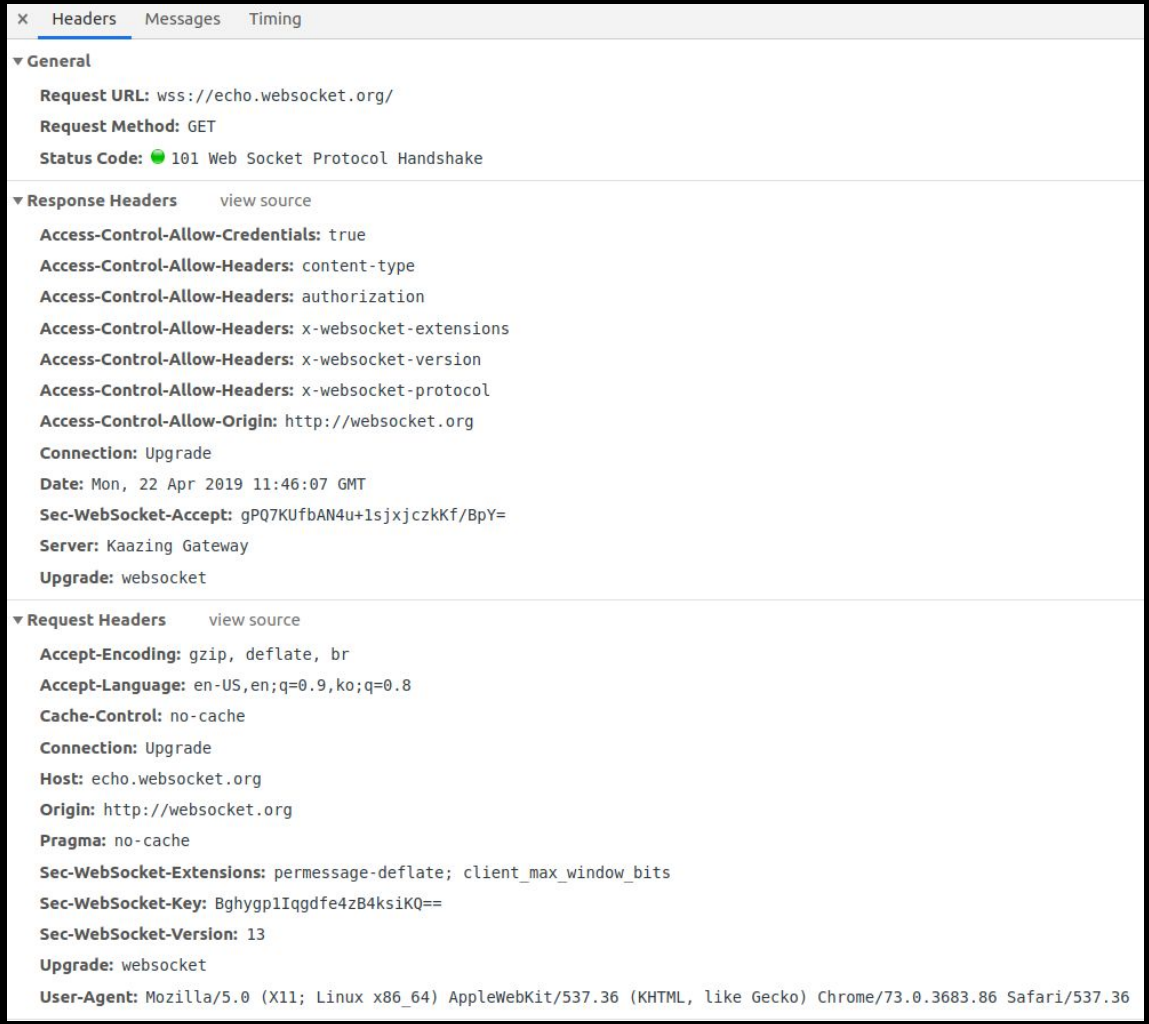
client

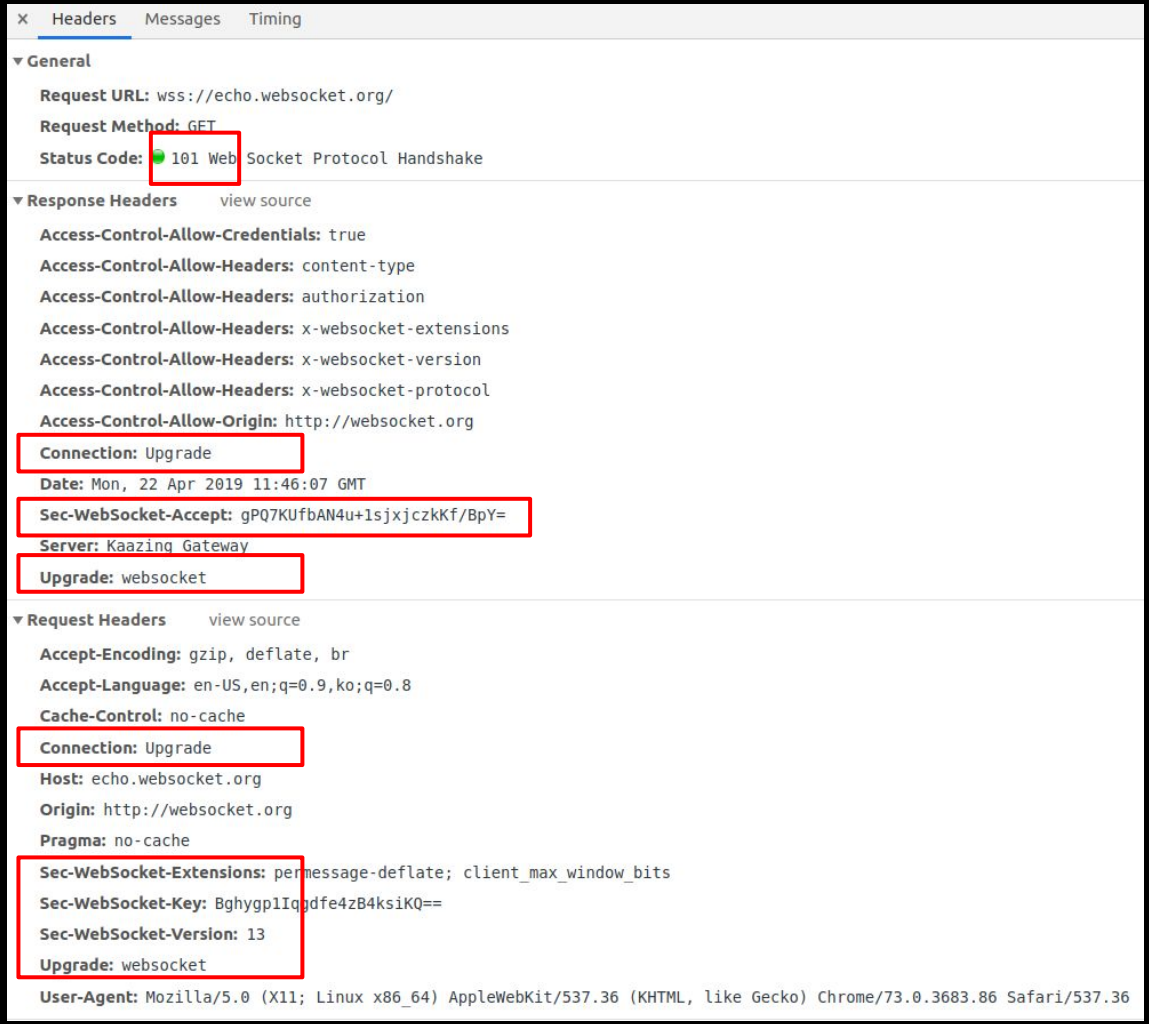
```
GET /chat HTTP/1.1
Host: example.com:8000
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Version: 13
```

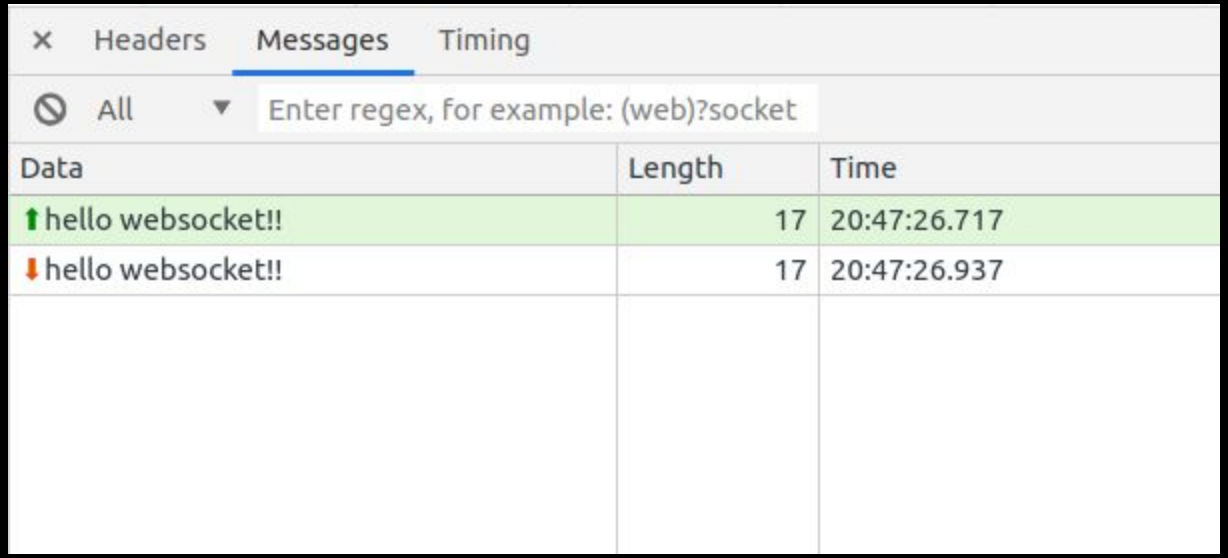
server

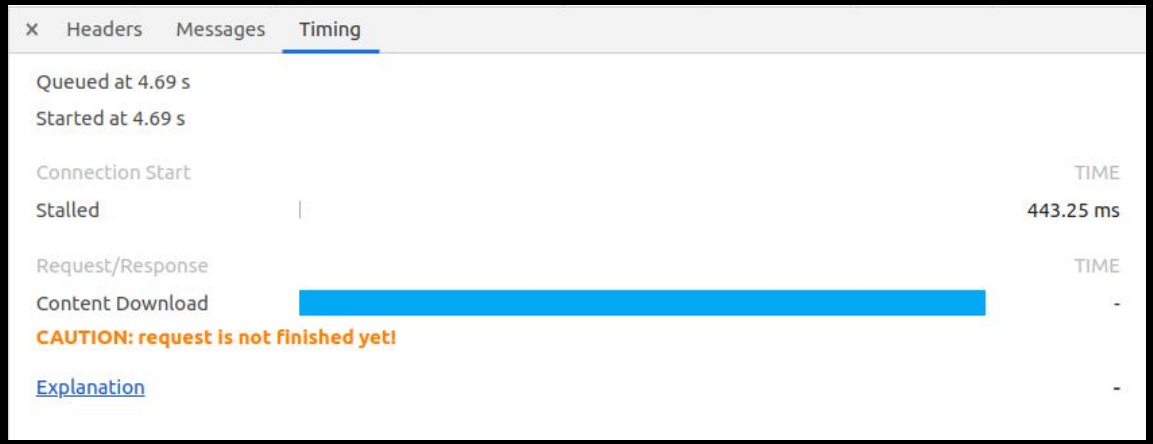
```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```


Not secure | websocket.org/echo.html
북마크 pdf youtube KAKAO Wgnu-gnu/spring… gnu-gnu/spring-… Other bookmarks
websocket.org
HOME DEMOS ARTICLES BOOK DOWNLOAD DOCKER HUB REPO ABOUT
Elements Console Sources Network Performance Memory Application Security
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
50000 ms 100000 ms 150000 ms 200000 ms 250000 ms 300000 ms
Name Status Type Initiator Size Time Waterfall
echo.websocket.org 101 websocket websocketttest:1 0 B Pen…
1 / 111 requests | 0 B / 8.3 KB transferred | 0 B / 2.3 MB resources | Finish: 4.8 min | DOMContentLoaded: 1.58 s | Load: 1.91 s

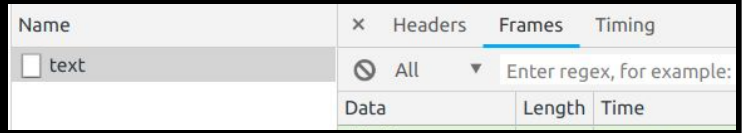

Headers
Messages
Timing
General
Request URL: wss://echo.websocket.org/
Request Method: GET
Status Code: 101 Web Socket Protocol Handshake
Response Headers
view source
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: content-type
Access-Control-Allow-Headers: authorization
Access-Control-Allow-Headers: x-websocket-extensions
Access-Control-Allow-Headers: x-websocket-version
Access-Control-Allow-Headers: x-websocket-protocol
Access-Control-Allow-Origin: http://websocket.org
Connection: Upgrade
Date: Mon, 22 Apr 2019 11:46:07 GMT
Sec-WebSocket-Accept: gPQ7KUfbAN4u+1sjxjczkKf/BpY=
Server: Kaazing Gateway
Upgrade: websocket
Request Headers
view source
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cache-Control: no-cache
Connection: Upgrade
Host: echo.websocket.org
Origin: http://websocket.org
Pragma: no-cache
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
Sec-WebSocket-Key: Bghygp1Iqgdfe4zB4ksiKQ==
Sec-WebSocket-Version: 13
Upgrade: websocket
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36


Headers
Messages
Timing
General
Request URL: wss://echo.websocket.org/
Request Method: GET
Status Code: 101 Web Socket Protocol Handshake
Response Headers
view source
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: content-type
Access-Control-Allow-Headers: authorization
Access-Control-Allow-Headers: x-websocket-extensions
Access-Control-Allow-Headers: x-websocket-version
Access-Control-Allow-Headers: x-websocket-protocol
Access-Control-Allow-Origin: http://websocket.org
Connection: Upgrade
Date: Mon, 22 Apr 2019 11:46:07 GMT
Sec-WebSocket-Accept: gPQ7KUfbAN4u+1sjxjczkKf/BpY=
Server: Kaazing Gateway
Upgrade: websocket
Request Headers
view source
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cache-Control: no-cache
Connection: Upgrade
Host: echo.websocket.org
Origin: http://websocket.org
Pragma: no-cache
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
Sec-WebSocket-Key: Bghygp1Iqgdfe4zB4ksiKQ==
Sec-WebSocket-Version: 13
Upgrade: websocket
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36

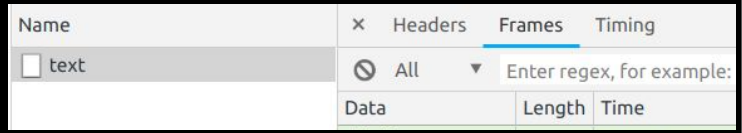

Headers
Messages
Timing
All
Enter regex, for example: (web)?socket
Data
Length
Time
hello websocket!!
17
20:47:26.717
hello websocket!!
17
20:47:26.937


Headers
Messages
Timing
Queued at 4.69 s
Started at 4.69 s
Connection Start
TIME
Stalled
443.25 ms
Request/Response
TIME
Content Download
-
CAUTION: request is not finished yet!
Explanation
-

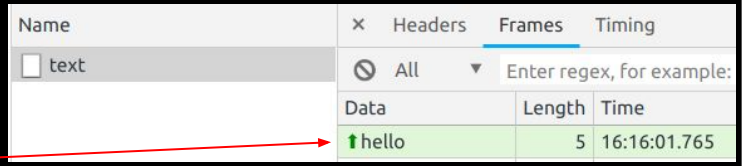
```
<script>
var ws = new WebSocket('ws://localhost:8080/text');

ws.onopen = function() {
    ws.send('hello');
}

ws.onmessage = function(event) {
    console.log(event);
}

document.getElementById('btn').onclick = function() {
    ws.send('asdfasfd');
}
```



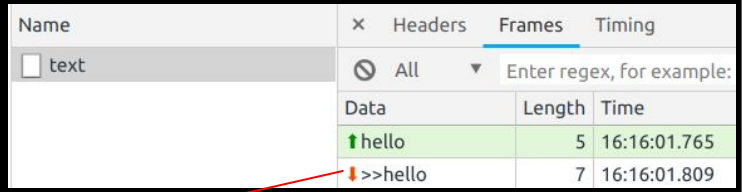
```
<script>
var ws = new WebSocket('ws://localhost:8080/text');

ws.onopen = function() {
    ws.send('hello');
}

ws.onmessage = function(event) {
    console.log(event);
}

document.getElementById('btn').onclick = function() {
    ws.send('asdfasfd');
}
```



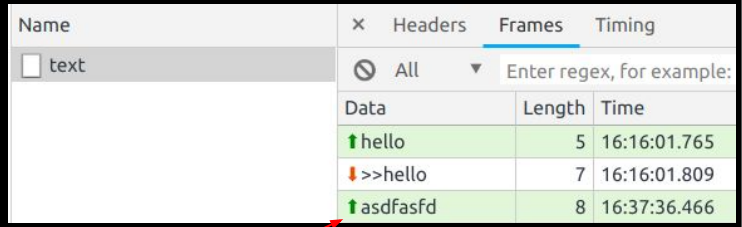
```
<script>
var ws = new WebSocket('ws://localhost:8080/text');

ws.onopen = function() {
    ws.send('hello');
}

ws.onmessage = function(event) {
    console.log(event);
}

document.getElementById('btn').onclick = function() {
    ws.send('asdfasfd');
}
```



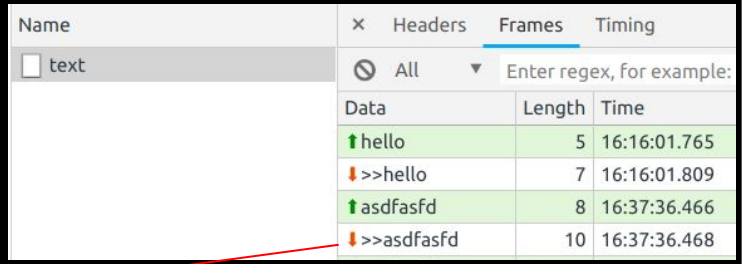
```
<script>
var ws = new WebSocket('ws://localhost:8080/text');

ws.onopen = function() {
    ws.send('hello');
}

ws.onmessage = function(event) {
    console.log(event);
}

document.getElementById('btn').onclick = function() {
    ws.send('asdfasfd');
}
```



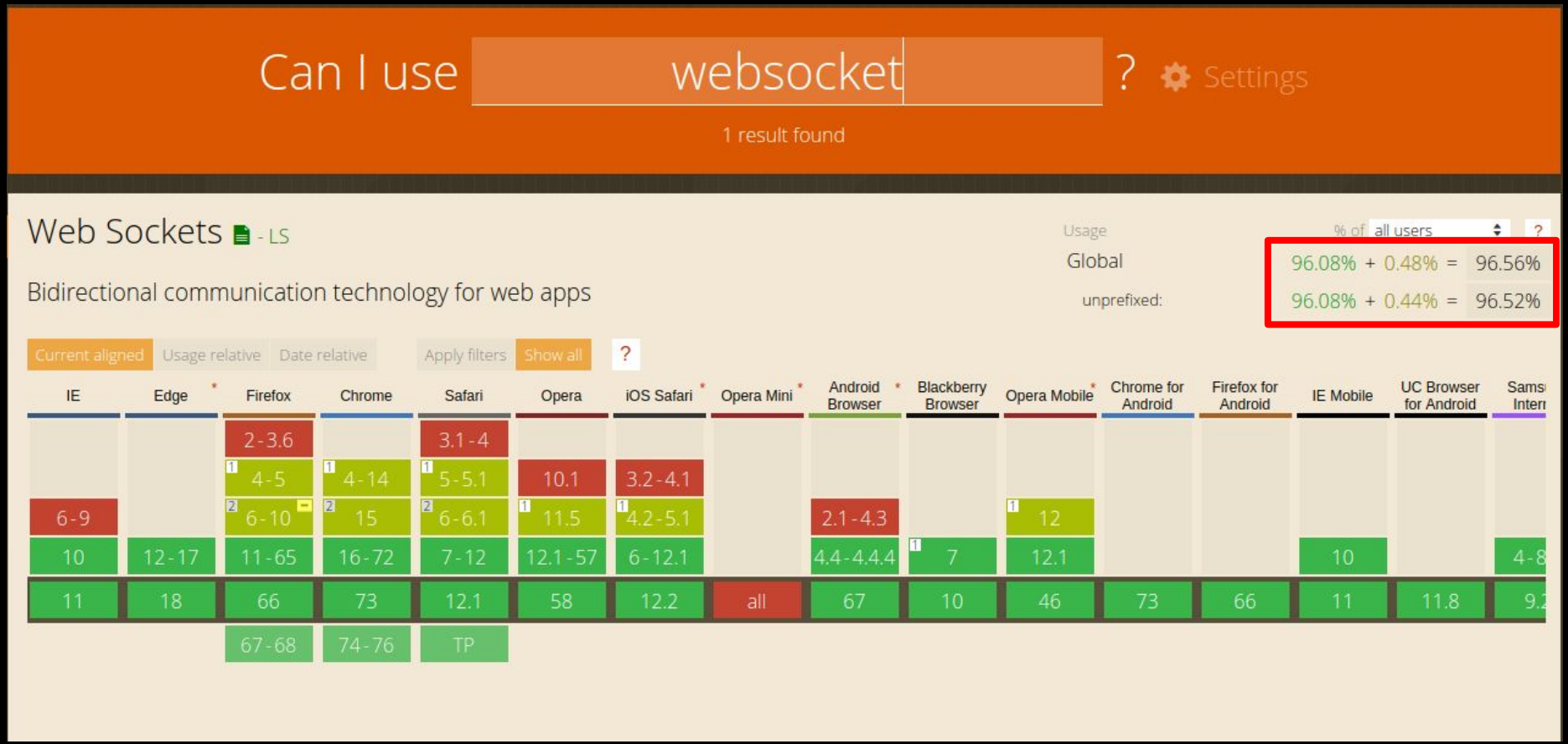
```
<script>
var ws = new WebSocket('ws://localhost:8080/text');

ws.onopen = function() {
    ws.send('hello');
}

ws.onmessage = function(event) {
    console.log(event);
}

document.getElementById('btn').onclick = function() {
    ws.send('asdfasfd');
}
```



```
<script>
var ws = new WebSocket('ws://localhost:8080/text');

ws.onopen = function() {
    ws.send('hello');
}

ws.onmessage = function(event) {
    console.log(event);
}

document.getElementById('btn').onclick = function() {
    ws.send('asdfasfd');
}
```



브라우저 지원현황


Can I use
websocket
?
Settings
1 result found
Web Sockets - LS
Bidirectional communication technology for web apps
Usage
% of all users
Global
96.08% + 0.48% = 96.56%
unprefixed:
96.08% + 0.44% = 96.52%
Current aligned
Usage relative
Date relative
Apply filters
Show all
?
IE
Edge
Firefox
Chrome
Safari
Opera
iOS Safari
Opera Mini
Android Browser
Blackberry Browser
Opera Mobile
Chrome for Android
Firefox for Android
IE Mobile
UC Browser for Android
Samsung Internet
2-3.6
3.1-4
4-5
4-14
5-5.1
10.1
3.2-4.1
6-9
6-10
15
6-6.1
11.5
4.2-5.1
2.1-4.3
12
10
12-17
11-65
16-72
7-12
12.1-57
6-12.1
4.4-4.4.4
7
12.1
10
4-8
11
18
66
73
12.1
58
12.2
all
67
10
46
73
66
11
11.8
9.2
67-68
74-76
TP

sockjs

websocket emulation

websocket과 같이 동작하기 위한
cross-browser javascript library

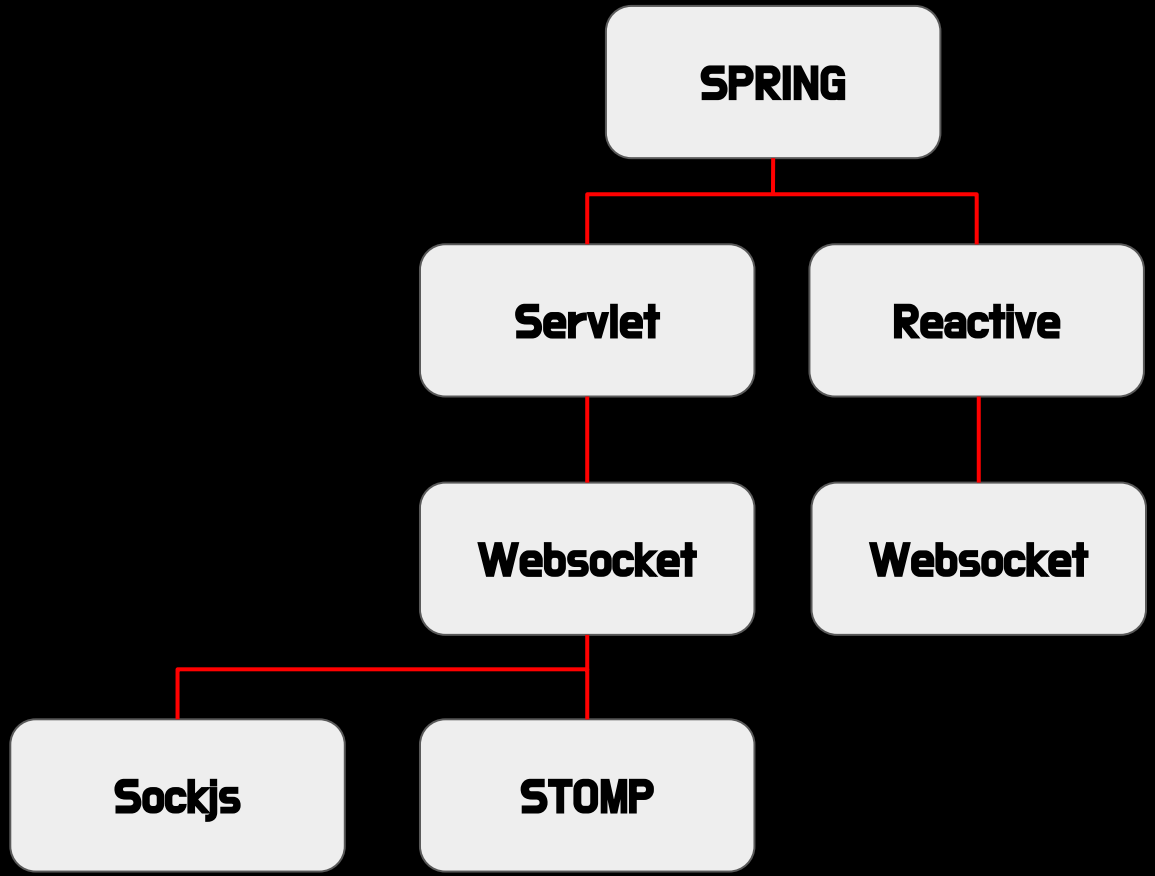
protocol

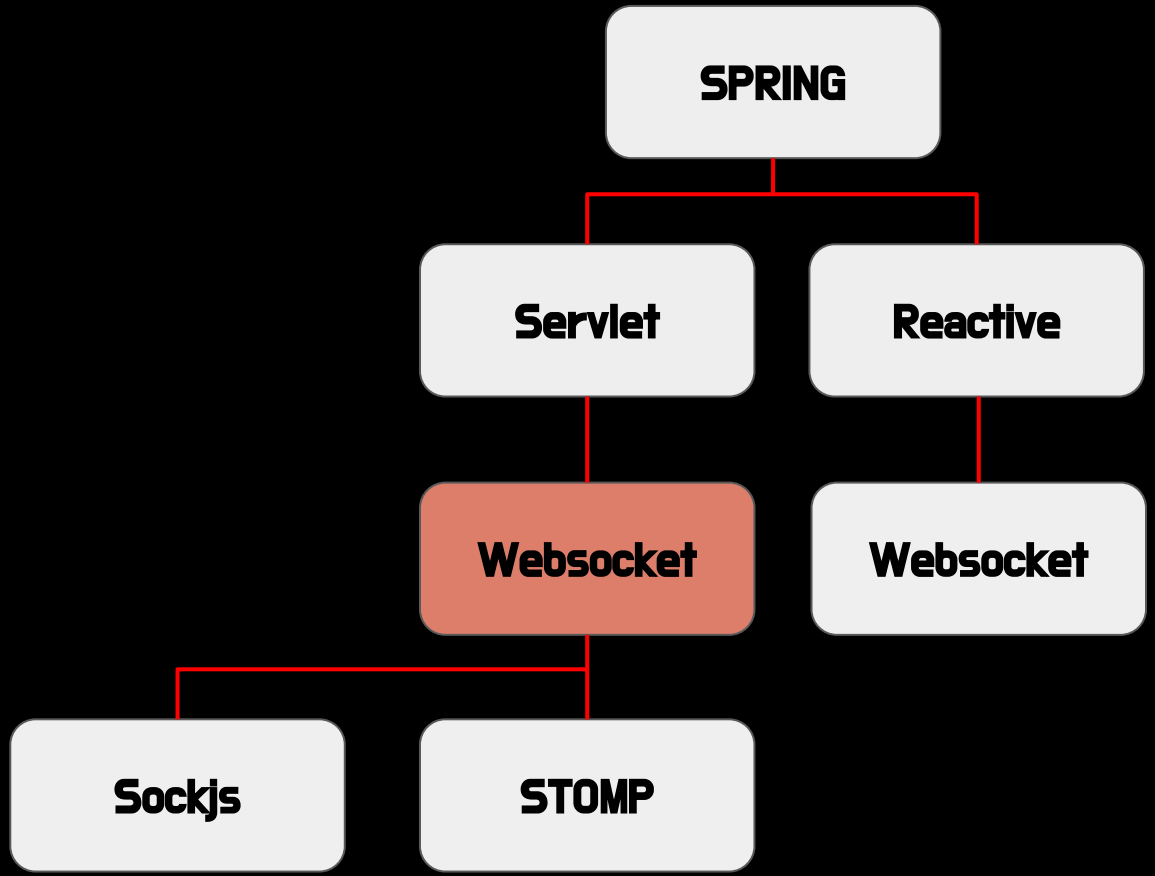
Browser	Websockets	Streaming	Polling
IE 6, 7	no	no	jsonp-polling
IE 8, 9 (cookies=no)	no	xdr-streaming †	xdr-polling †
IE 8, 9 (cookies=yes)	no	iframe-htmlfile	iframe-xhr-polling
IE 10	rfc6455	xhr-streaming	xhr-polling
Chrome 6-13	hixie-76	xhr-streaming	xhr-polling
Chrome 14+	hybi-10 / rfc6455	xhr-streaming	xhr-polling
Firefox <10	no ‡	xhr-streaming	xhr-polling
Firefox 10+	hybi-10 / rfc6455	xhr-streaming	xhr-polling
Safari 5.x	hixie-76	xhr-streaming	xhr-polling
Safari 6+	rfc6455	xhr-streaming	xhr-polling
Opera 10.70+	no ‡	iframe-eventsource	iframe-xhr-polling
Opera 12.10+	rfc6455	xhr-streaming	xhr-polling
Konqueror	no	no	jsonp-polling

```html
<html>
<head>
  <script src="https://cdn.jsdelivr.net/npm/sockjs-client@1/dist/sockjs.min.js"></script>
</head>
<body>
<script>
  var sock = new SockJS('https://localhost:8080/websocket');
  sock.onopen = function() {
    console.log('open');
    sock.send('test');
  };

  sock.onmessage = function(e) {
    console.log('message', e.data);
    sock.close();
  };

  sock.onclose = function() {
    console.log('close');
  };
</script>
</body>
</html>
```

이제 백엔드 이야기를 해볼까요?

spring framework으로 websocket을 사용하는 방법.


SPRING
Servlet
Reactive
Websocket
Websocket
Sockjs
STOMP


SPRING
Servlet
Reactive
Websocket
Websocket
Sockjs
STOMP

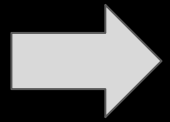
```java
@Configuration
@EnableWebSocket
class WebSocketConfig implements WebSocketConfigurer {
    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(websocketHandler(), ...paths: "/websocket");
    }

    @Bean
    public WebSocketHandler websocketHandler() { return new WebsocketHandler(); }
}
```

```java
@Configuration
@EnableWebSocket
class WebSocketConfig implements WebSocketConfigurer {
    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(websocketHandler(), ...paths: "/websocket");
    }

    @Bean
    public WebSocketHandler websocketHandler() { return new WebsocketHandler(); }
}
```

entry point를 하나 만들고
client side에서 정해진 규칙으로
데이터를 던지고
로직으로 분기해주면 편함.

client

```
{
  "type": "BOARD.CREATE",
  "data": {
    "title": "hello world",
    "content": "websocket"
  }
}
```

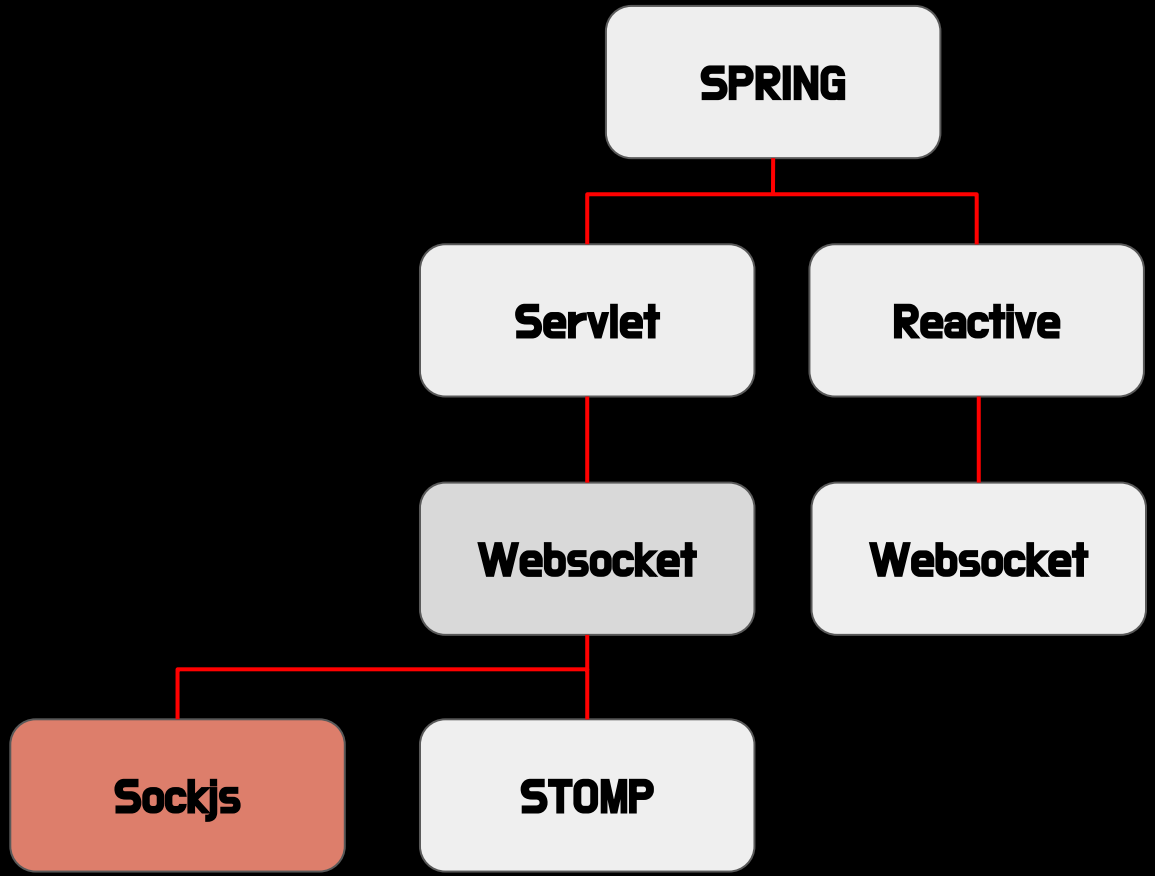


server

```
public class Event<T> {
    private EventType type;
    private String username;
    private T data;
}
```

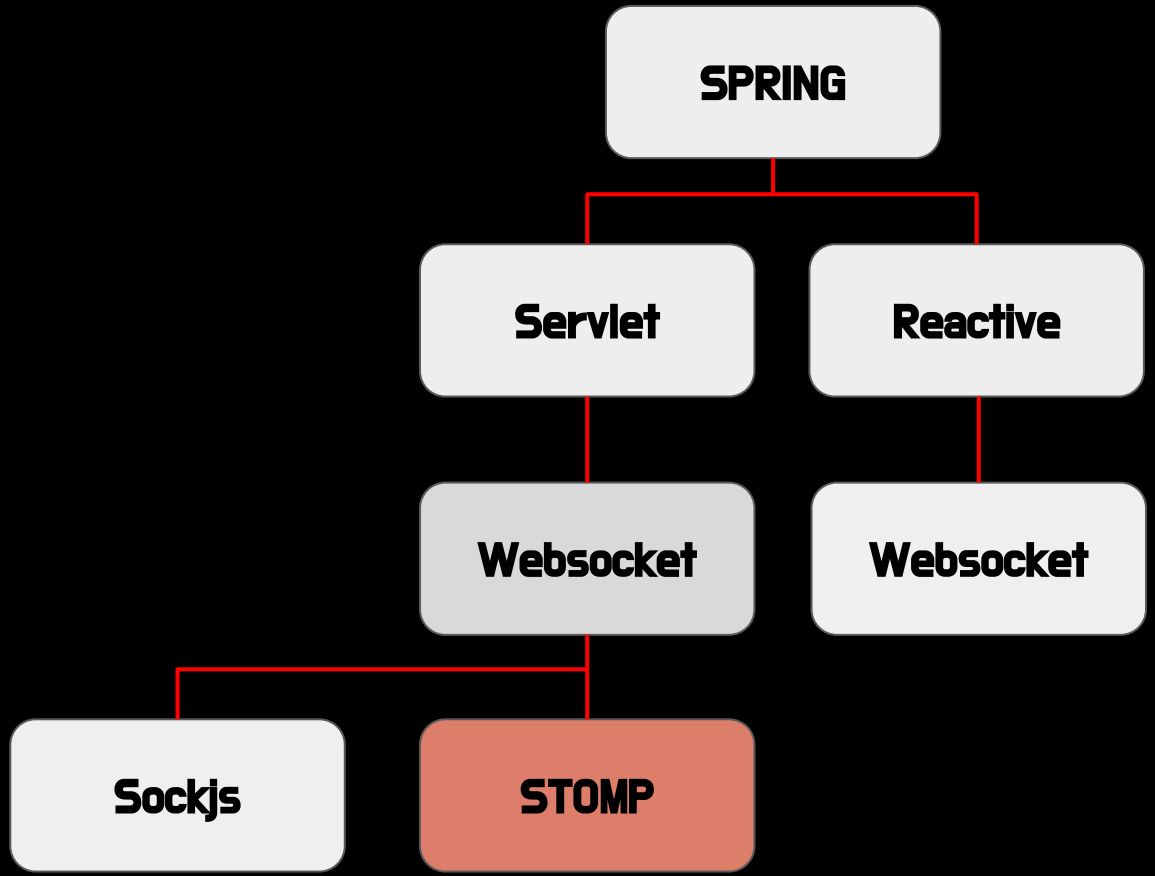
```java
class WebsocketHandler extends TextWebSocketHandler {
    private Set<WebSocketSession> sessions = new ConcurrentHashMap().newKeySet();

    @Override
    public void afterConnectionEstablished(WebSocketSession session) throws Exception {
        sessions.add(session);
    }

    @Override
    public void handleTextMessage(WebSocketSession session, TextMessage message) {
        try {
            session.sendMessage(new TextMessage( payload: "hello client: " + (Math.random() * 10)));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws Exception {
        sessions.remove(session);
    }
}
```

```
class WebsocketHandler extends TextWebSocketHandler {
    private Set<WebSocketSession> sessions = new ConcurrentHashMap().newKeySet();

    @Override
    public void afterConnectionEstablished(WebSocketSession session) throws Exception {
        sessions.add(session);
    }

    @Override
    public void handleTextMessage(WebSocketSession session, TextMessage message) {
        try {
            session.sendMessage(new TextMessage( payload: "hello client: " + (Math.random() * 10)));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void afterConnectionClosed(WebSocketSession session, CloseStatus status) throws Exception {
        sessions.remove(session);
    }
}
```


SPRING
Servlet
Reactive
Websocket
Websocket
Sockjs
STOMP

```java
@Configuration
@EnableWebSocket
class WebSocketConfig implements WebSocketConfigurer {
    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(websocketHandler(), ...paths: "/websocket")
            .withSockJS();
    }

    @Bean
    public WebSocketHandler websocketHandler() {
        return new WebsocketHandler();
    }
}
```

```
@Configuration
@EnableWebSocket
class WebSocketConfig implements WebSocketConfigurer {
    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(websocketHandler(), ...paths: "/websocket")
            .withSockJS();
    }

    @Bean
    public WebSocketH                                    ler() {
        return new We
    }
}
```


SPRING
Servlet
Reactive
Websocket
Websocket
Sockjs
STOMP

STOMP

웹소켓을 쓰면 일단은 프로토콜을 다 만들어주고

파싱하셔야 하는게

일단 브라우저당 한 도메인에는 하나만 연결되거나 그럴꺼에요.

stomp가 얼마만틈 해주는지는 잘 모르겠네요

저도 웹소켓은 사용한적이 없어서 ㅎㅎ

프로토콜 정의만 잘해두면...

사실 큰 문제는 없을듯 한데...

일단 컨넥션이 맺고 있는 상태고

다만 기본적으로는 text base니...

binary 형태보다는 수신량이 많을 수 있다.

spring이 처리를 해주면...

다른 언어지원은?

간단한게 모니터링하거나 테스트 하는것도

다 쉬운가?

stomp 에 문제가 있으면...

누가 처리할 것인가 정도겠네요.

웹소켓을 쓰면 일단은 프로토콜을 다 만들어주고

파싱하셔야 하는게

그저 빛...

하나만 연결되거나 그럴꺼에요.

stomp가 얼마만틈 해주는지는 잘 모르겠네요

저도 웹소켓은 사용한적이 없어서 ㅎㅎ

프로토콜 정의만 잘해두면...

사실 큰 문제는 없을듯 한데...

일단 컨넥션이 맺고 있는 상태고

다만 기본적으로는 text base니...

binary 형태보다는 수신량이 많을 수 있다.

spring이 처리를 해주면...

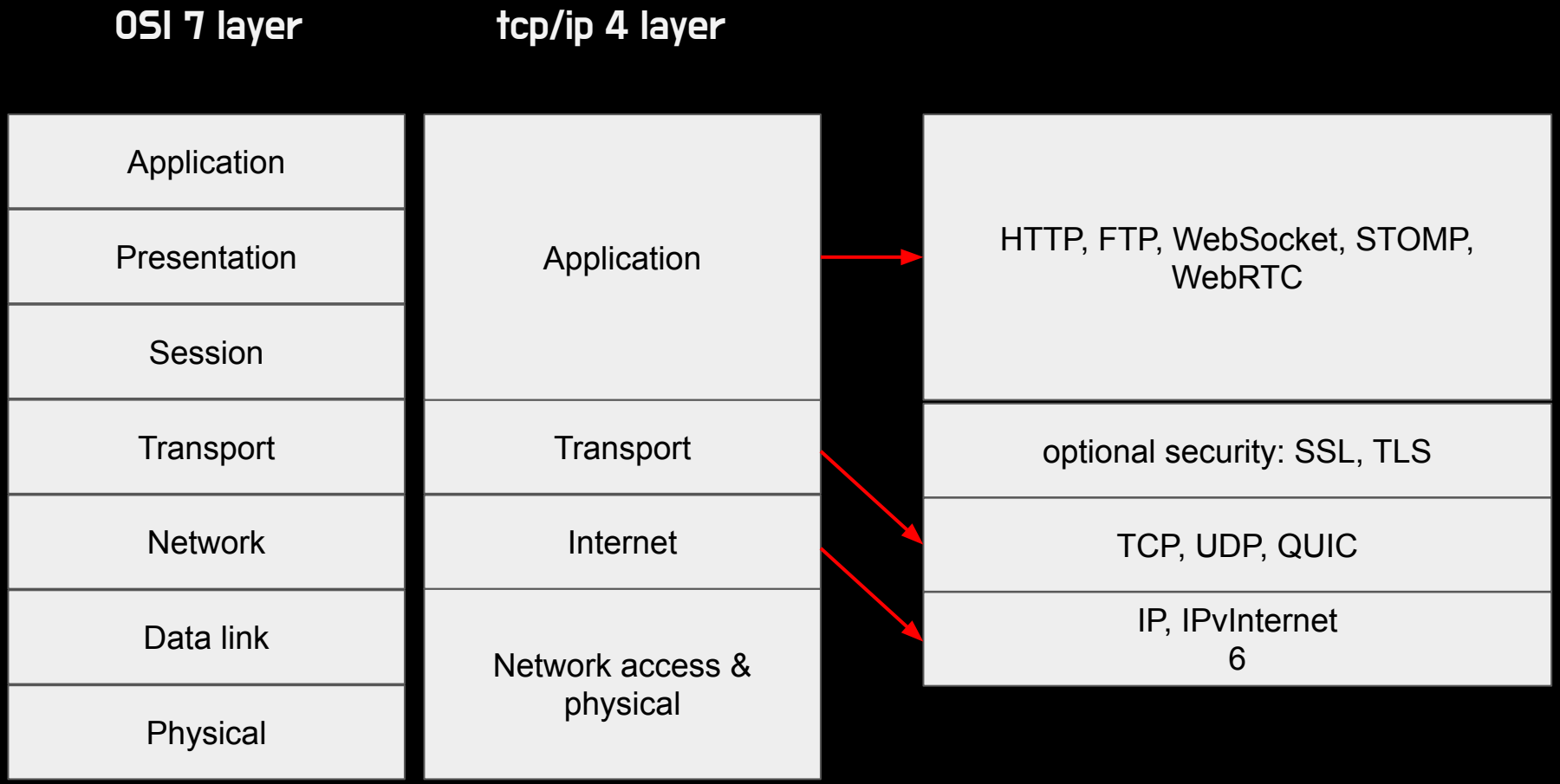
다른 언어지원은?

간단한게 모니터링하거나 테스트 하는것도

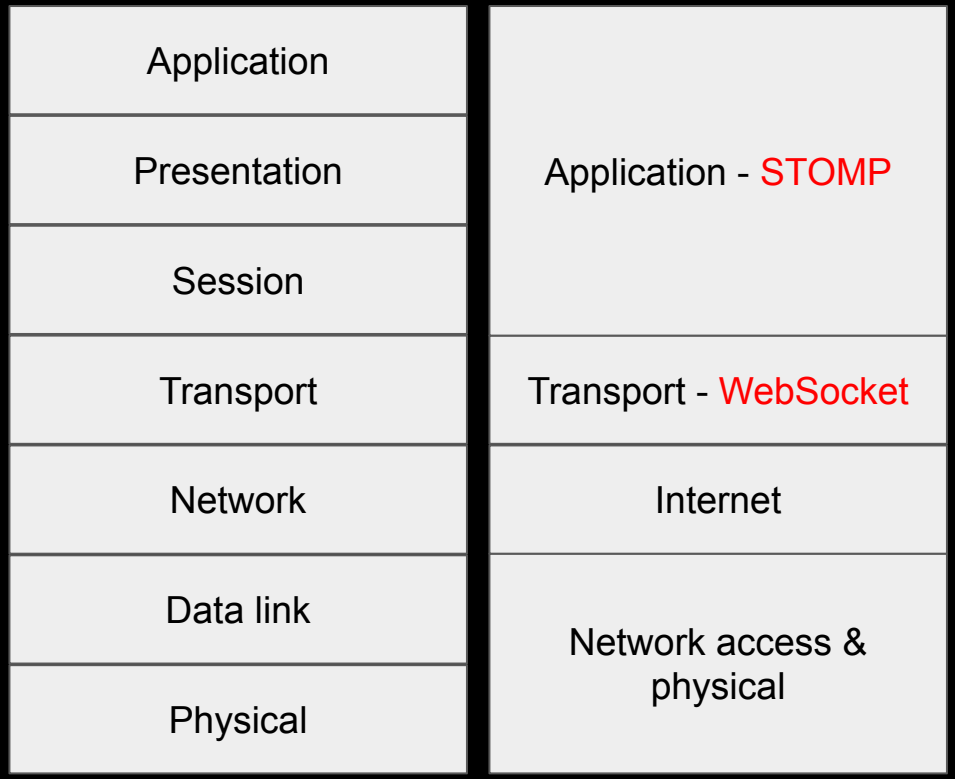
다 쉬운가?

stomp 에 문제가 있으면...

누가 처리할 것인가 정도겠네요.


OSI 7 layer
tcp/ip 4 layer
Application
Presentation
Session
Transport
Network
Data link
Physical
Application
Transport
Internet
Network access & physical
HTTP, FTP, WebSocket, STOMP, WebRTC
optional security: SSL, TLS
TCP, UDP, QUIC
IP, IPvInternet 6

이런식으로 생각하면
좀 편함



websocket 위에 STOMP 프로토콜을 이용한다면?

websocket

websocket

프로토콜 설계
클라/서버 구현

security

websocket

프로토콜 설계
클라/서버 구현

security

websocket

프로토콜 설계
클라/서버 구현

라우팅

security

websocket

프로토콜 설계
클라/서버 구현

아키텍처/스케일아웃

라우팅

security

websocket

프로토콜 설계
클라/서버 구현

아키텍처/스케일아웃

라우팅

API 설계

security

websocket

프로토콜 설계
클라/서버 구현

아키텍처/스케일아웃

라우팅

API 설계

에러처리

security

websocket

프로토콜 설계
클라/서버 구현

아키텍처/스케일아웃

라우팅

API 설계

에러처리

STOMP란? 링크

Simple	간단한
Text	텍스트
Oriented	기반
Message	메세지
Protocol	프로토콜

간단하고 오래된(마지막 버전인 1.2가 2012년 10월) 프로토콜.

프레임(Command + Header + Body) 단위로 구성

메세지큐의 동작을 프로토콜로 정의했다고 생각하면 편리함.

rabbitmq의 경우 plugin을 설치하면 사용 가능 (63613 port)

STOMP Servers

Here are the known STOMP compliant message servers:

Name	Description	Compliance
Apache ActiveMQ	the most popular and powerful open source messaging and Integration Patterns server	1.0 1.1
Apache ActiveMQ Artemis	Apache ActiveMQ Artemis has a proven non blocking architecture. It delivers outstanding performance.	1.0 1.1 1.2
Apache Apollo	a redesigned version of ActiveMQ	1.0 1.1 1.2
CoilMQ	a lightweight pure Python STOMP broker inspired by StompServer	1.0
Gozirra	a lightweight Java STOMP broker	1.0
HornetQ	puts the buzz in messaging	1.0
MorbidQ	a STOMP publish/subscribe server with absolutely no potential to cluster	1.0
RabbitMQ	an Erlang-based, multi-protocol broker with full support for STOMP via a plugin	1.0 1.1 1.2
Sprinkle	written in Python and runs on Unix type platforms	1.0
Stampy	a Java implementation of the STOMP 1.2 specification	1.2
StompConnect	provides a bridge to any other JMS provider	1.0
StompServer	a lightweight pure Ruby STOMP server	1.0

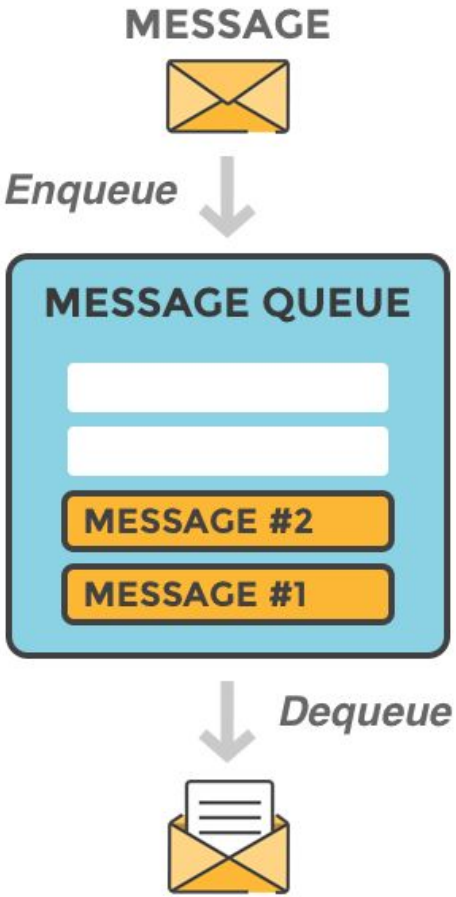
STOMP Clients

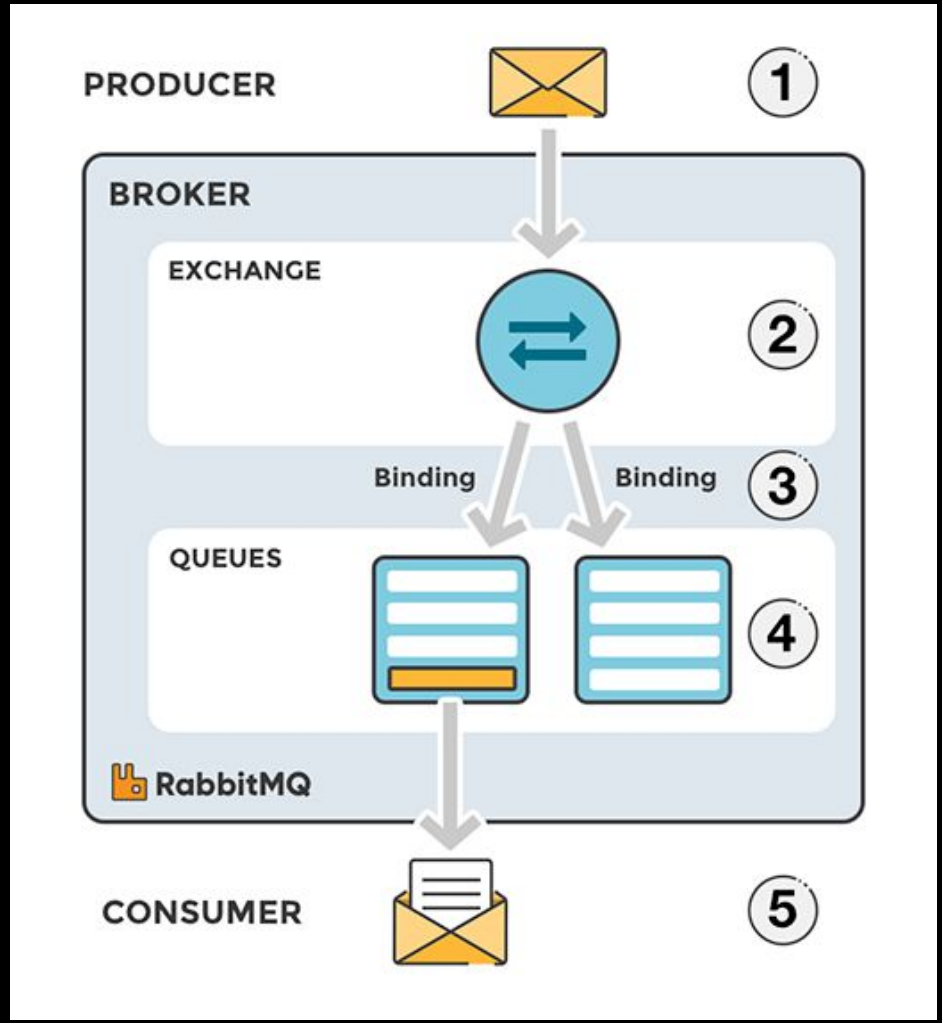
Here are the known STOMP compliant client libraries:

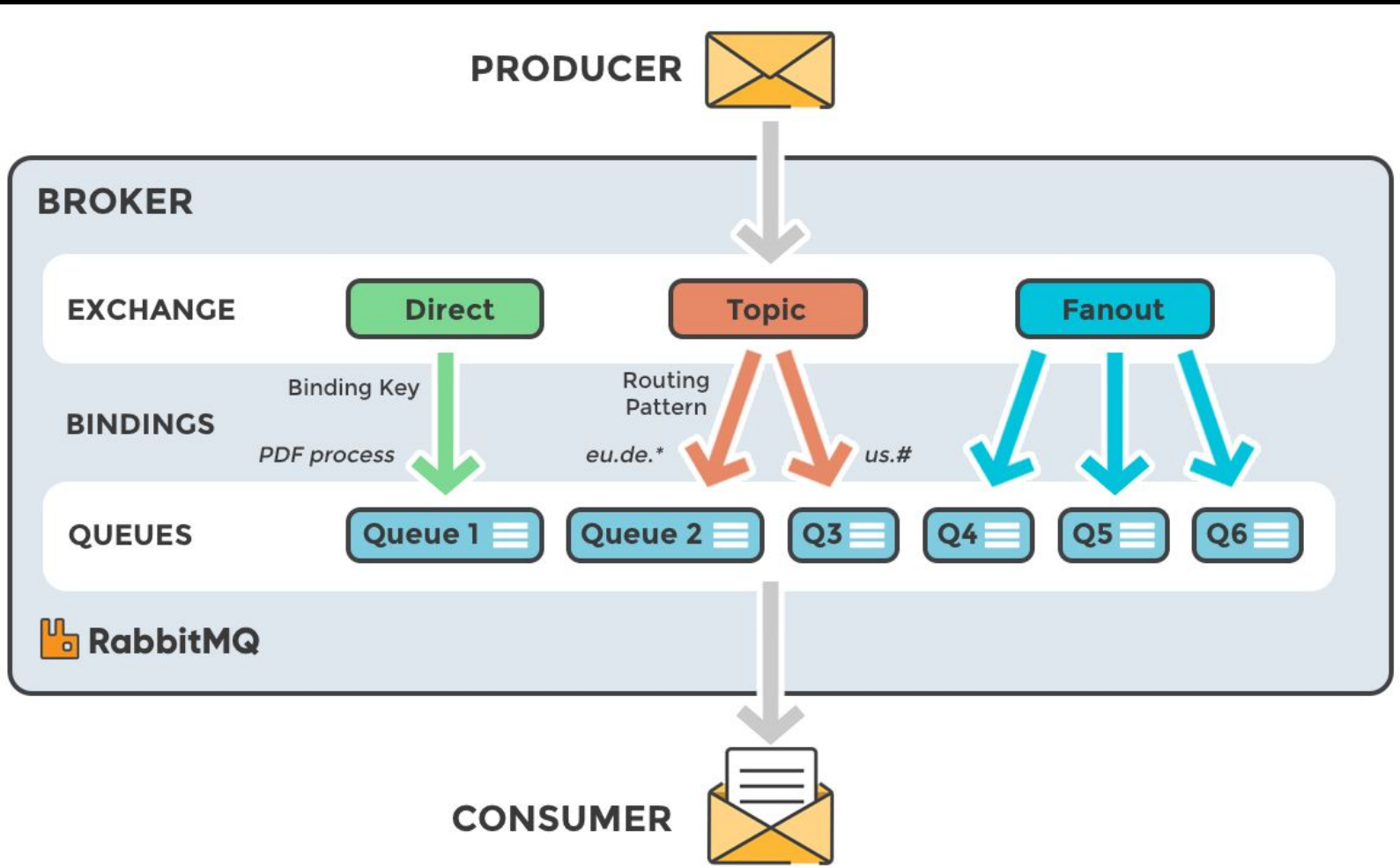
Name	Language	Description	Compliance
activemessaging	Ruby	an attempt to bring the simplicity and elegance of Rails development to the world of messaging	1.0
AnyEvent::STOMP	Perl	a lightweight event-driven STOMP client	1.0
Apache CMS	C++	is a JMS-like API for C++	1.0
Apache NMS	C# and .Net	a JMS-like API for .Net	1.0
as3-stomp	Flash	an actionscript 3 implementation of the STOMP protocol	1.0
delphistompclient	Delphi and FreePascal	a STOMP client for Embarcadero Delphi and FreePascal	1.0
dstomp	Dynamic C	a STOMP client library written in Dynamic C for Rabbit	1.0
Gozirra	Java	a lightweight implementation of the STOMP specification	1.0
hxStomp	Haxe	a TCP socket-based STOMP protocol client library written for the Haxe language	1.0
libstomp	C	an APR based C library	1.0
Net::Stomp	Perl	a Streaming Text Orientated Messaging Protocol client	1.0
Net::STOMP::Client	Perl	STOMP object oriented client module	1.0 1.1 1.2
objc-stomp	Objective-C	a simple STOMP client based on AsynSocket	1.0
POE::Component::Client::Stomp	Perl	a Perl extension for the POE Environment	1.0
onstomp	Ruby	client library for message passing with brokers that support the STOMP protocol	1.0 1.1

링크

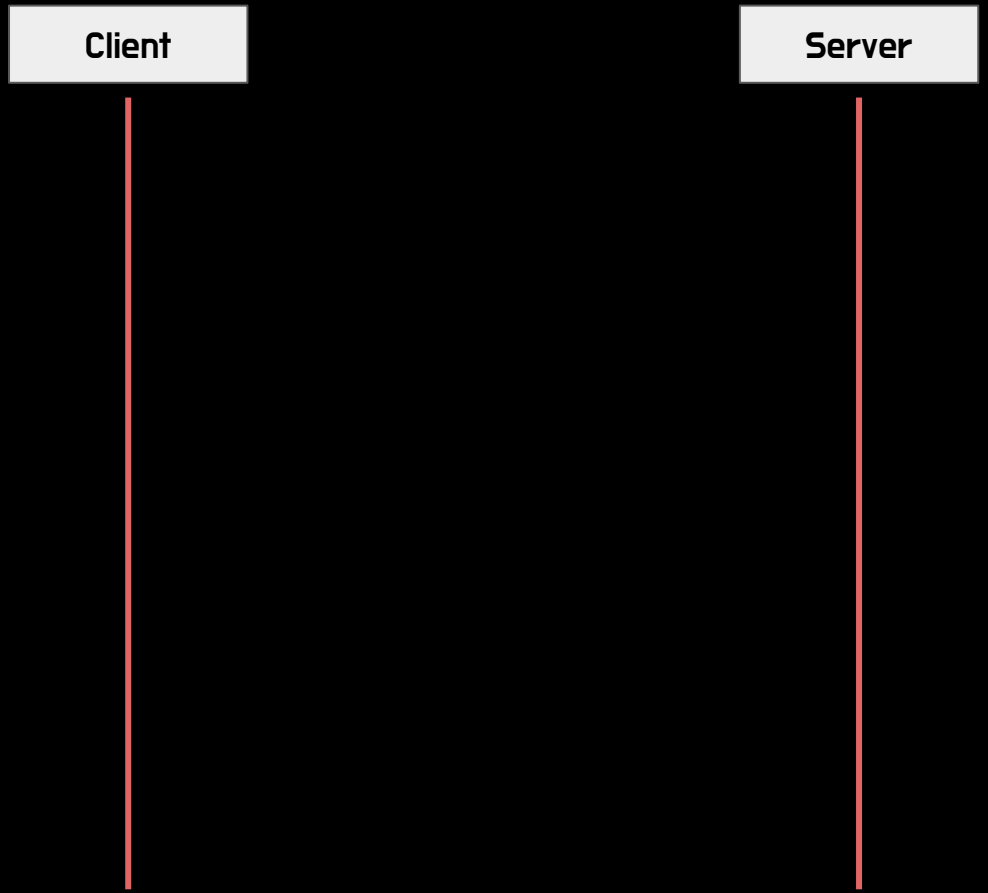
(배경지식) 메세지큐(브로커) 개념


MESSAGE
Enqueue
MESSAGE QUEUE
MESSAGE #2
MESSAGE #1
Dequeue

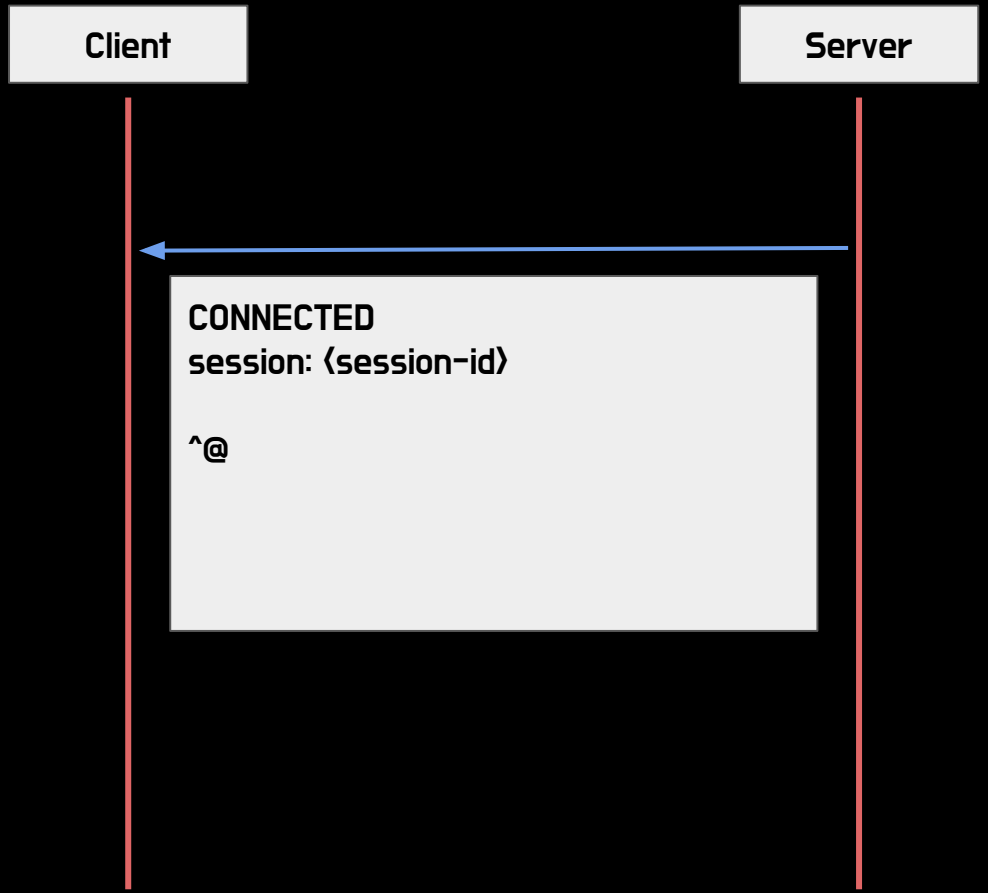

PRODUCER
1
BROKER
EXCHANGE
2
Binding
Binding
3
QUEUES
4
RabbitMQ
CONSUMER
5

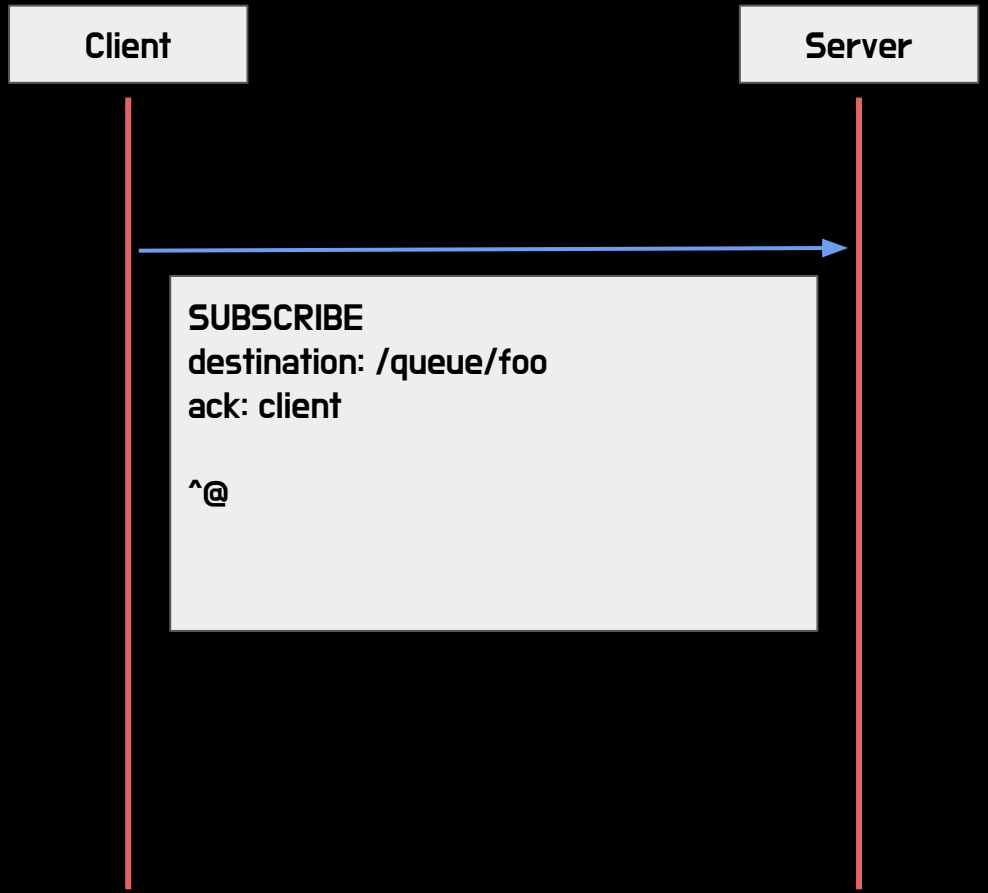

PRODUCER
BROKER
EXCHANGE
Direct
Topic
Fanout
BINDINGS
Binding Key
PDF process
Routing Pattern
eu.de.*
us.#
QUEUES
Queue 1
Queue 2
Q3
Q4
Q5
Q6
RabbitMQ
CONSUMER

STOMP 프로토콜 동작 도식화


Client
Server

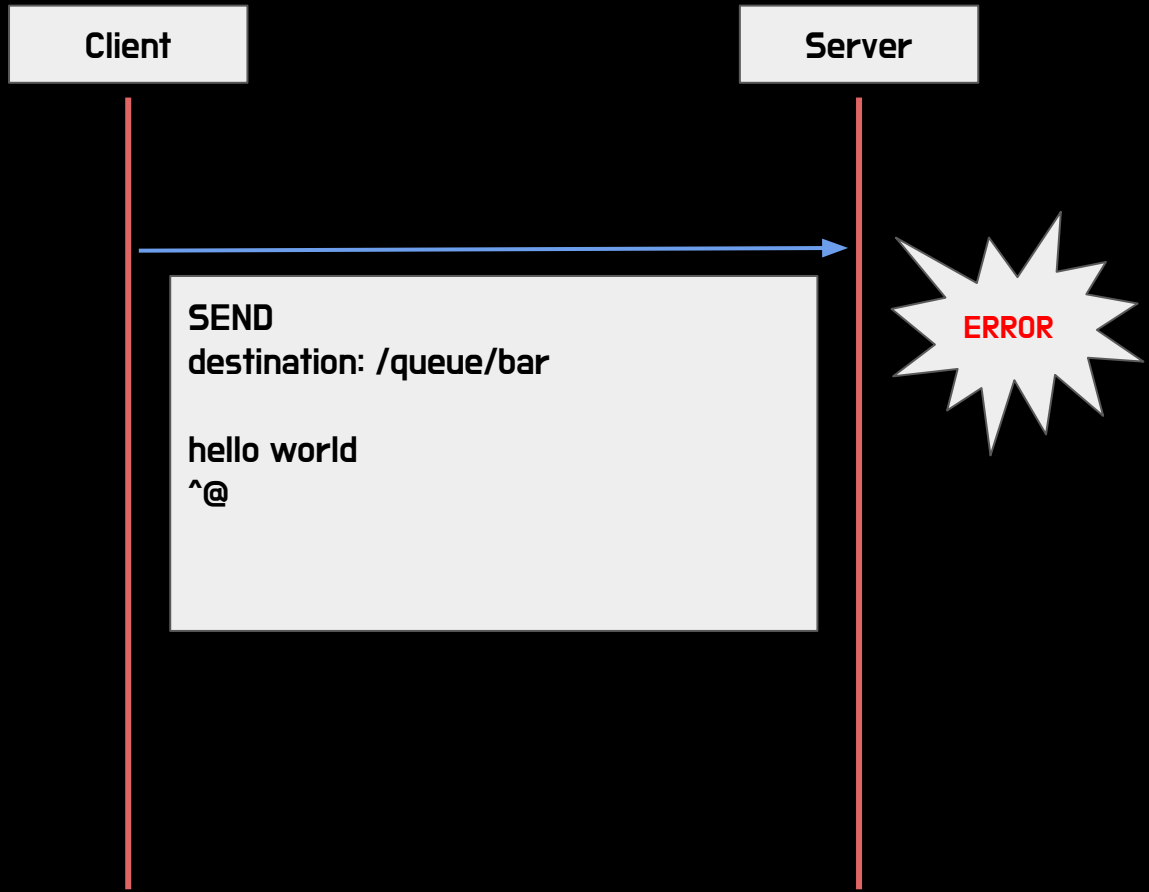

Client
Server
CONNECT
login: ⟨id⟩
password: ⟨password⟩
^@


Client
Server
CONNECTED
session: <session-id>
^@


Client
Server
SUBSCRIBE
destination: /queue/foo
ack: client
^@

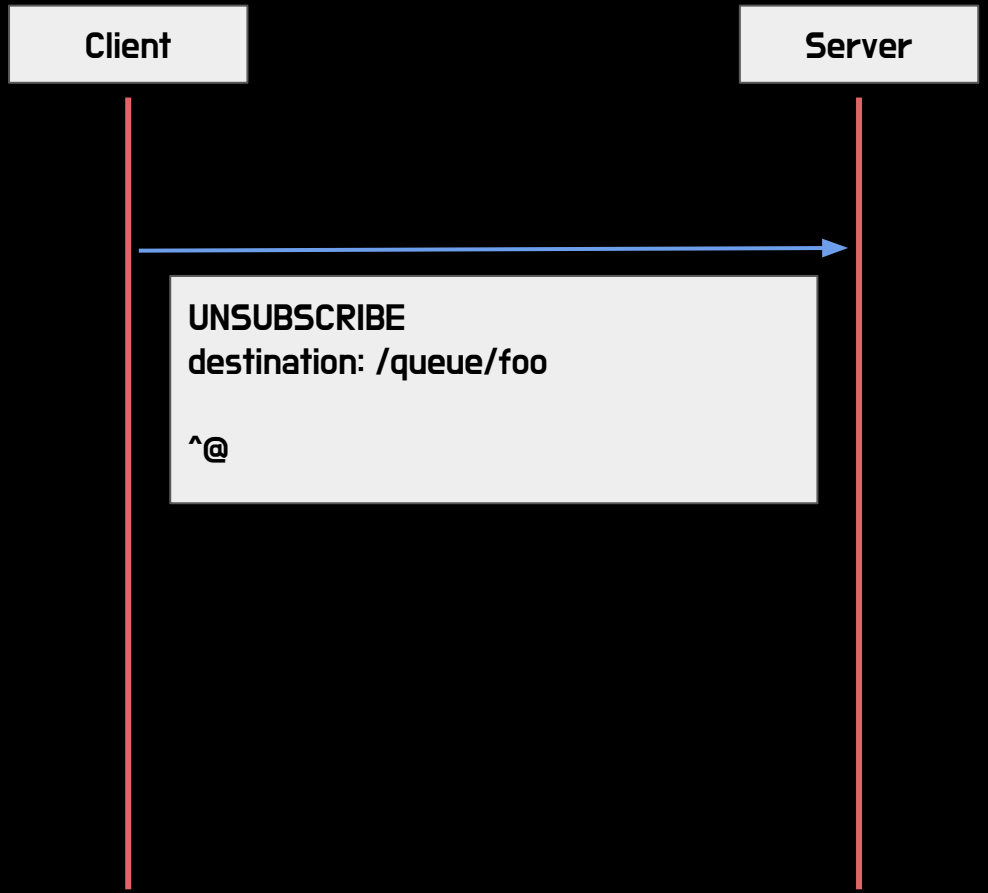

Client
Server
SEND
destination: /queue/bar
hello world
^@

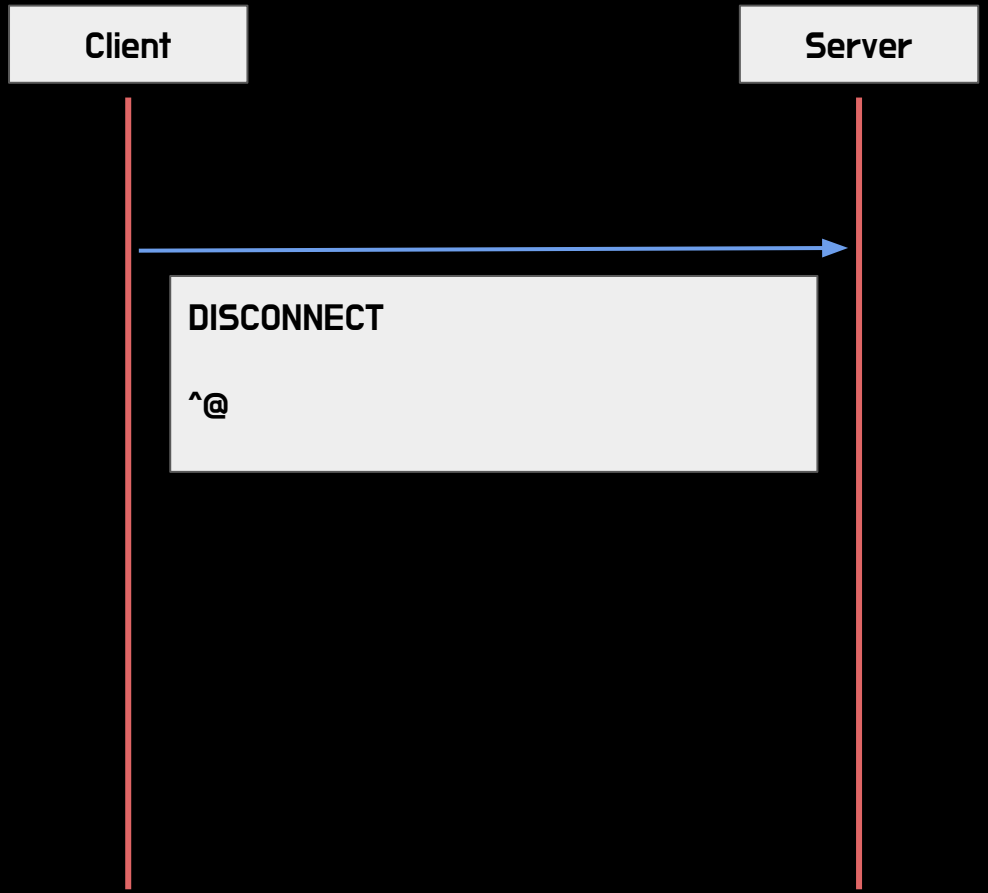

Client
Server
MESSAGE
destination: /queue/foo
message-id: <message-identifier>
hello world
^@


Client
Server
SEND
destination: /queue/bar
hello world
^@
ERROR


Client
Server
ERROR
message: malformed package recieved
The message:

MESSAGE
destined: /queue/foo
Hello world
---
error message
^@


Client
Server
UNSUBSCRIBE
destination: /queue/foo
^@


Client
Server
DISCONNECT
^@

transaction(BEGIN, COMMIT, ACK, ABORT), RECEIPT 있음.

자세한건 stomp 문서 참조.

스프링이 websocket 위에
stomp 프로토콜을 구현하였고
이와함께 message broker와 spring security를
그리고 spring mvc와 비슷한 개발 스타일을
사용할 수 있습니다.

client

stomp

server

stomp

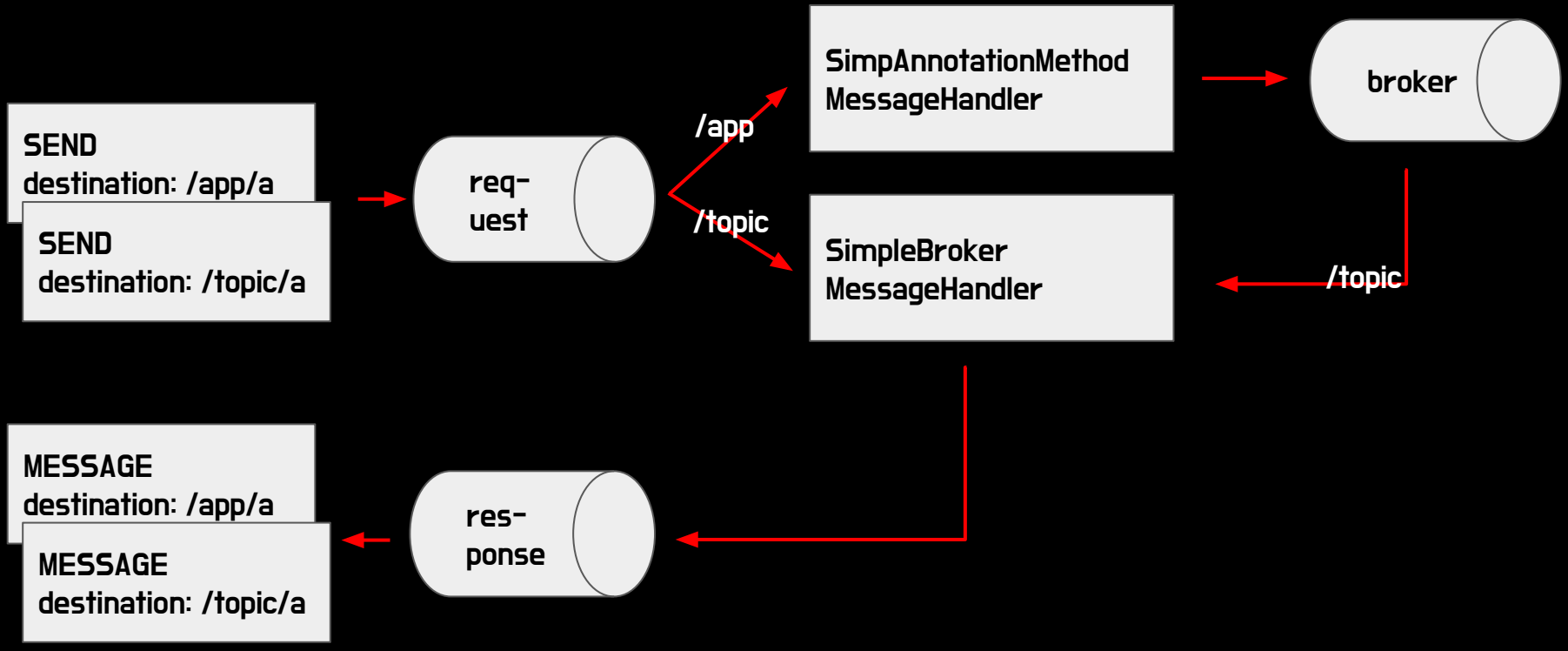
Java

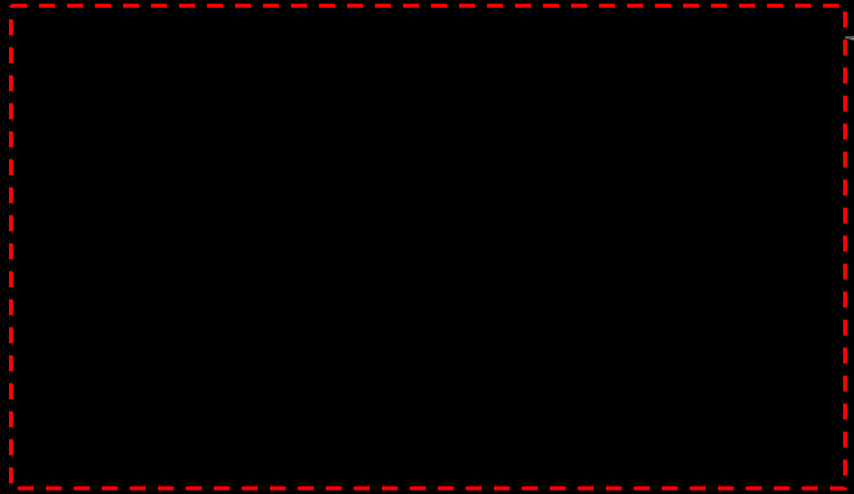
worker

stomp

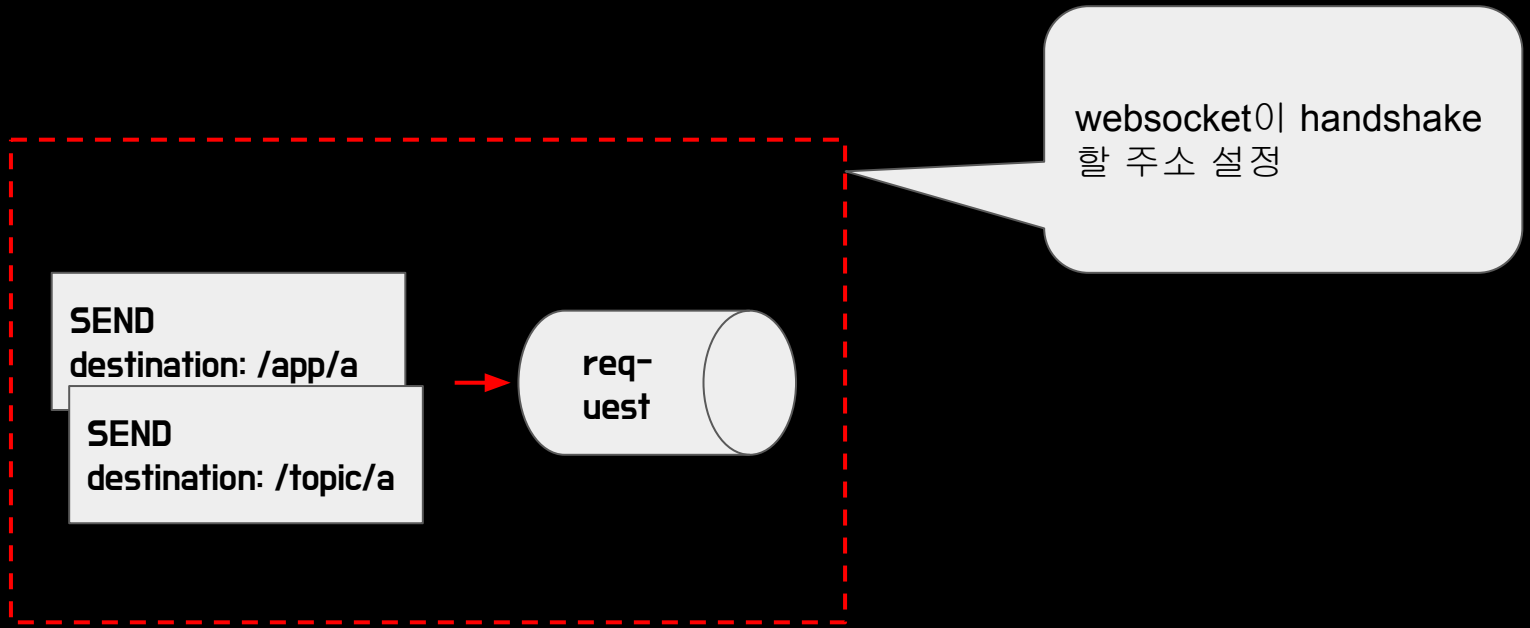
RabbitMQ

메세지 전체 흐름

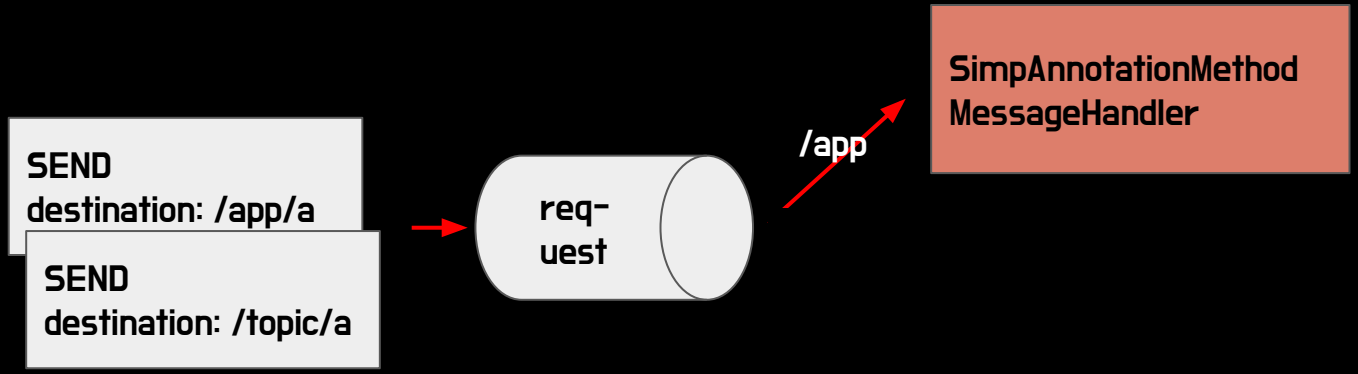

SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
broker
SimpleBroker
MessageHandler
/topic
MESSAGE
destination: /app/a
MESSAGE
destination: /topic/a
res-
ponse



websocket이 handshake 할 주소 설정


websocket이 handshake 할 주소 설정
SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest

```java
@Configuration
@EnableWebSocketMessageBroker
class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/stomp").withSockJS();
    }



}
```

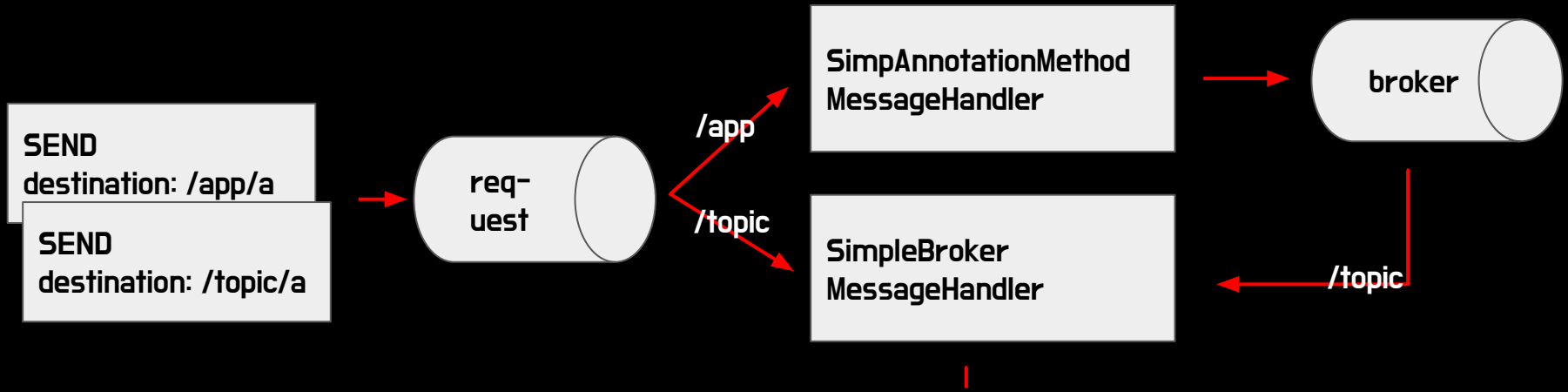

SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
SimpAnnotationMethod
MessageHandler

```java
@Configuration
@EnableWebSocketMessageBroker
class WebSocketConfig implements WebSocketMessageBrokerConfigurer {



    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {

        config.setApplicationDestinationPrefixes("/app");

    }
}
```

client-side
javascript

```
const socket = new SockJs('http://user:password@localhost:8080/stomp');
const stompClient = stompjs.over(socket);


stompClient.send('/app/event', {}, JSON.stringify( value: {type: "test"}));
```

```
@Controller
class StompController {


    @MessageMapping("/event")
    public void stomp(Map map) {
        ...
    }
}
```


SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
SimpleBroker
MessageHandler
broker
/topic

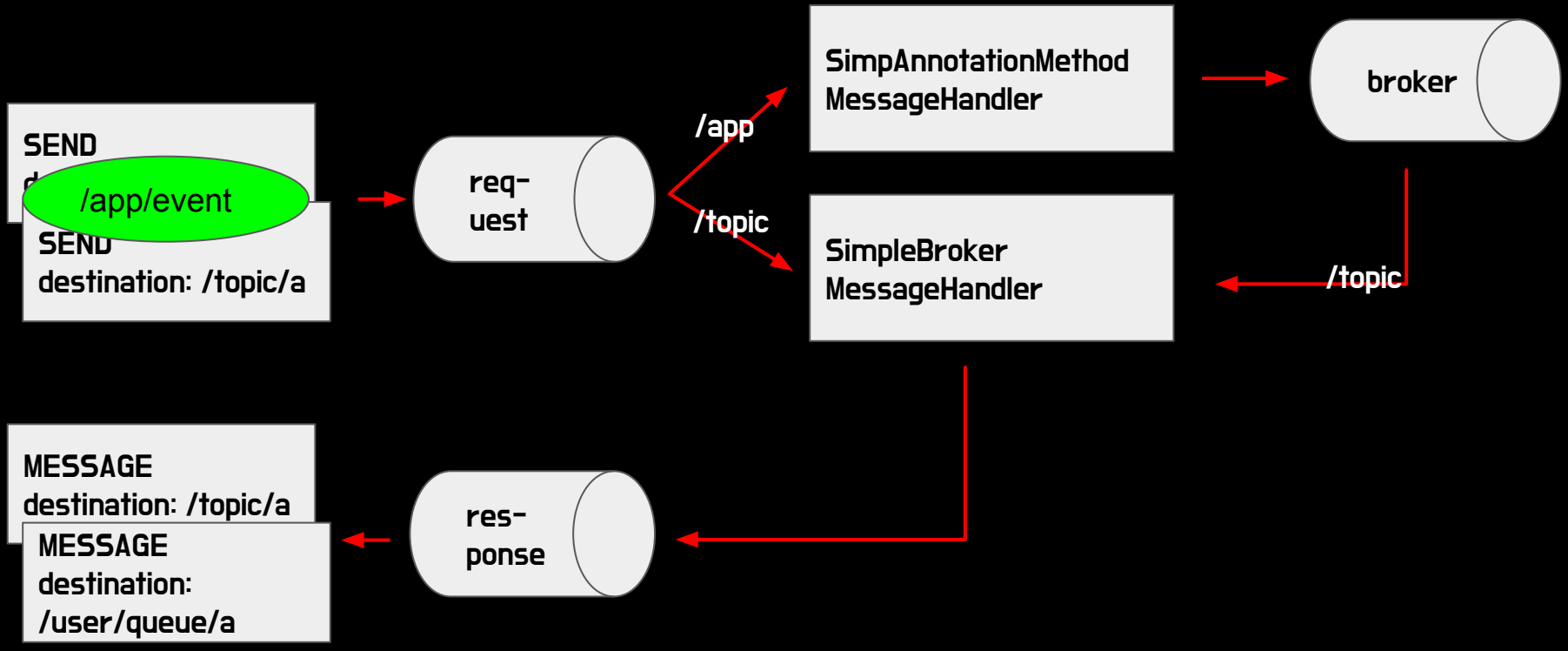
```java
@Configuration
@EnableWebSocketMessageBroker
class WebSocketConfig implements WebSocketMessageBrokerConfigurer {



    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {

        config.enableSimpleBroker( ...destinationPrefixes: "/topic", "/queue");
    }
}
```

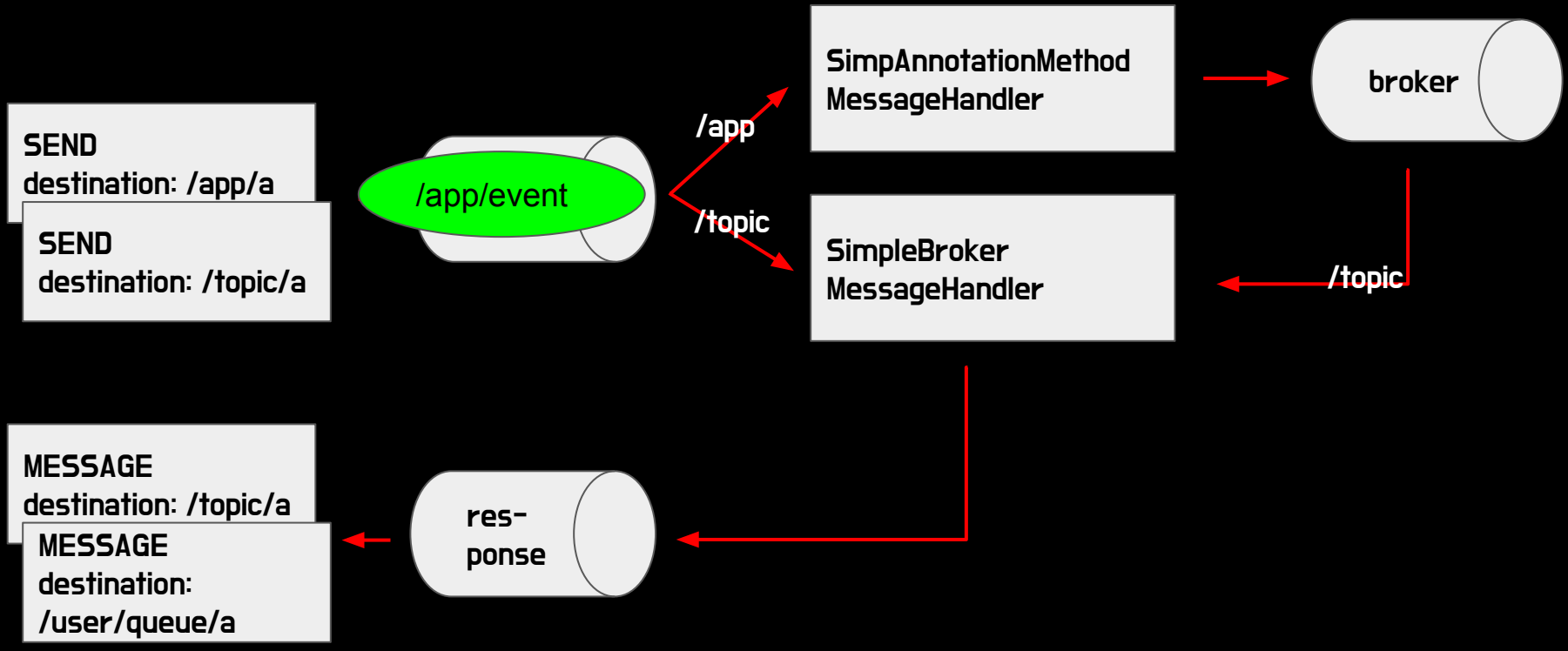
```java
@Configuration
@EnableWebSocketMessageBroker
class WebSocketConfig implements WebSocketMessageBrokerConfigurer {



    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {

        config.enableSimpleBroker( ...destinationPrefixes: "/topic", "/queue");
    }
}
```

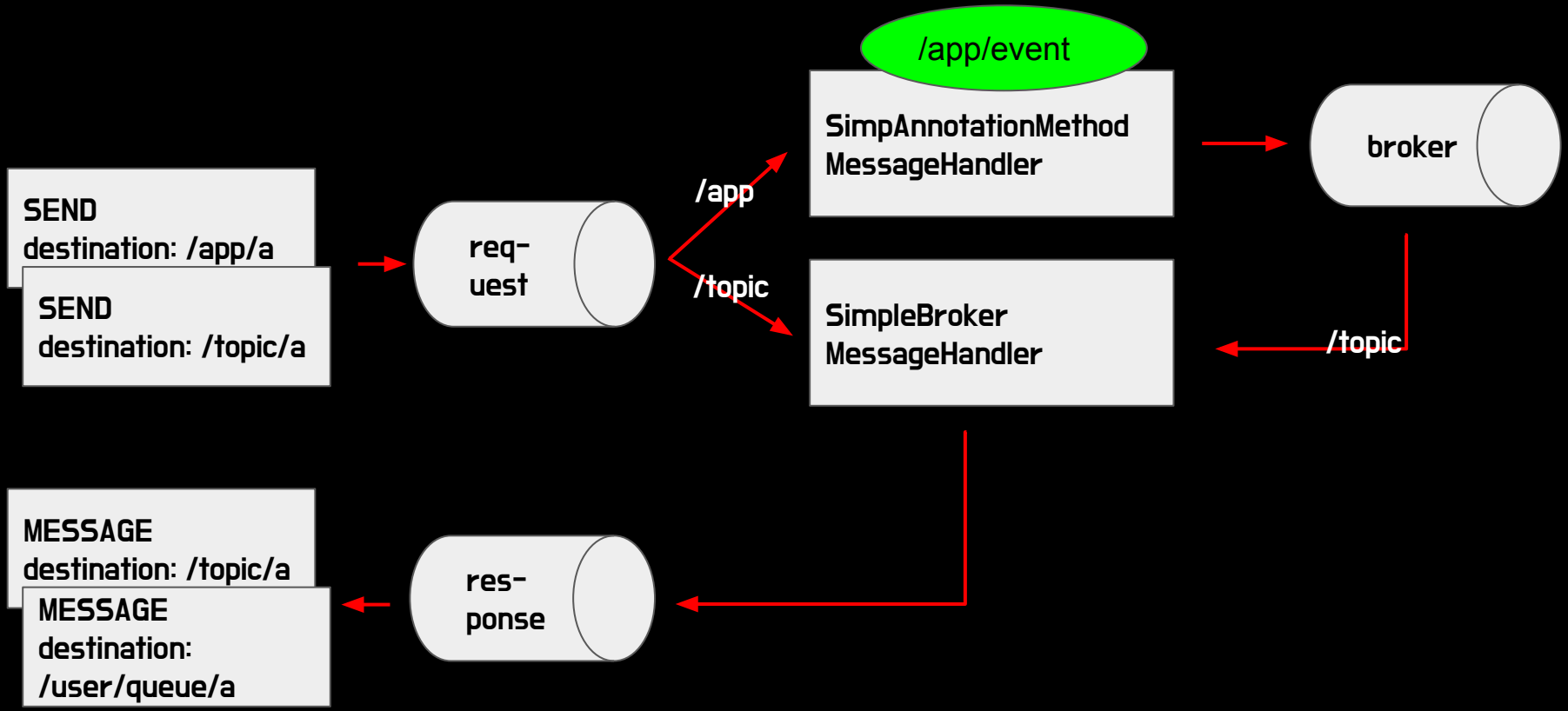
client-side
javascript

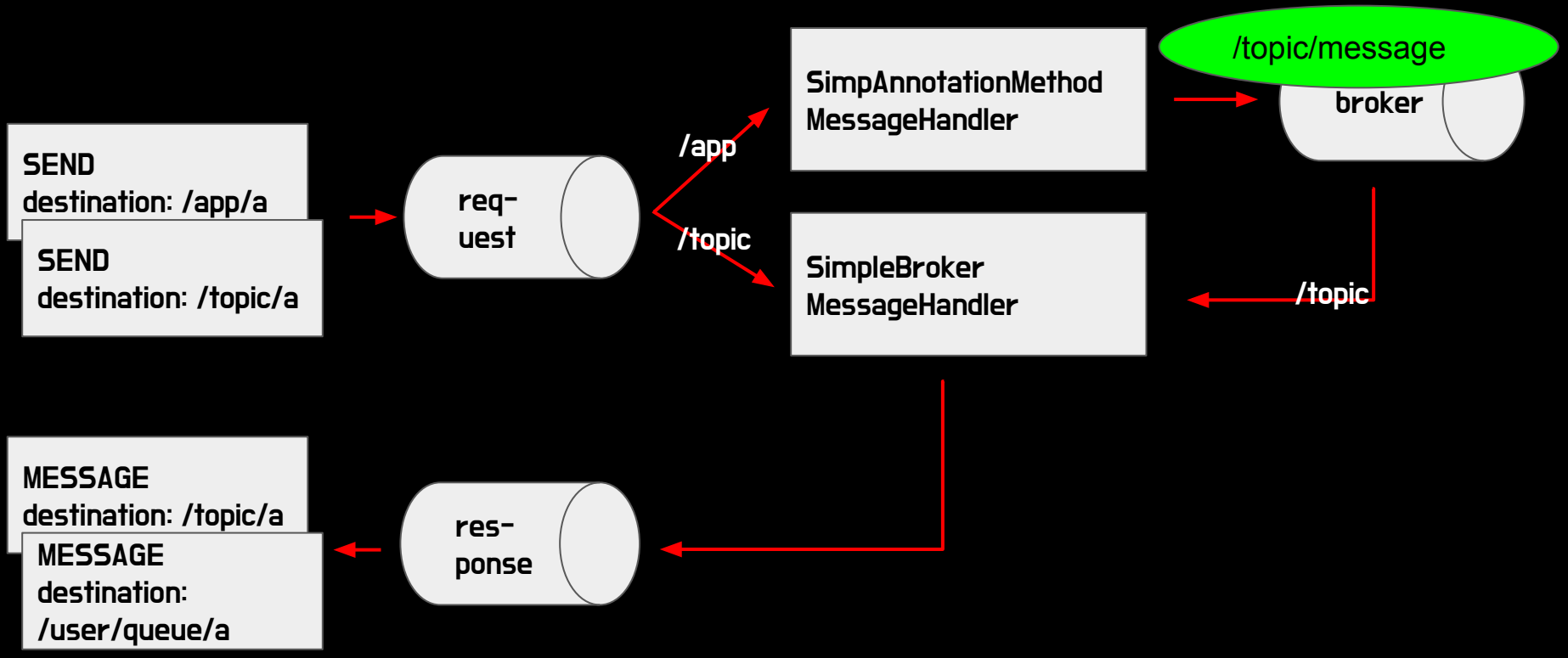
```
stompClient.subscribe('/topic/message', frame => {
  console.log(frame)
});
```

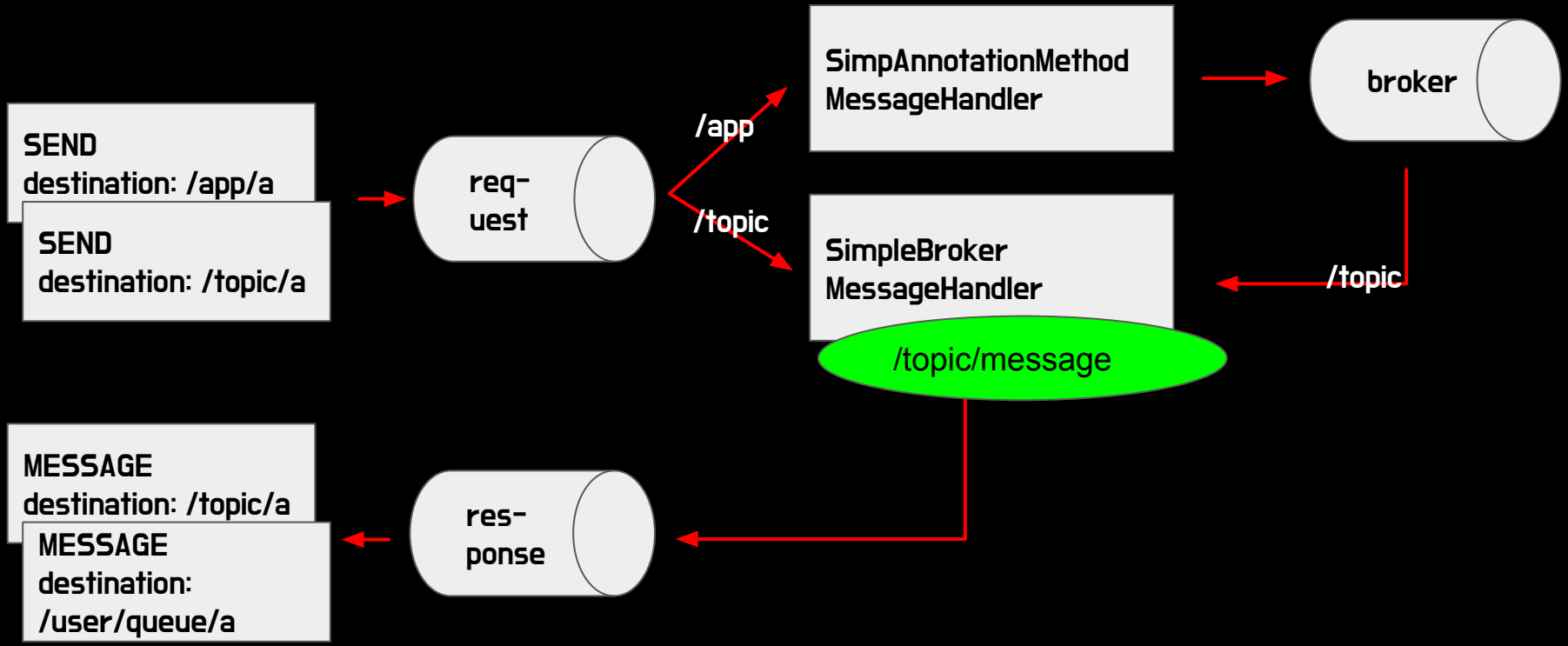
```java
@Controller
class StompController {
    private SimpMessagingTemplate template;

    @MessageMapping("/event")
    public void stomp(Map map) {
        this.template.convertAndSend(destination: "/topic/message", payload: "hello");
    }
}
```

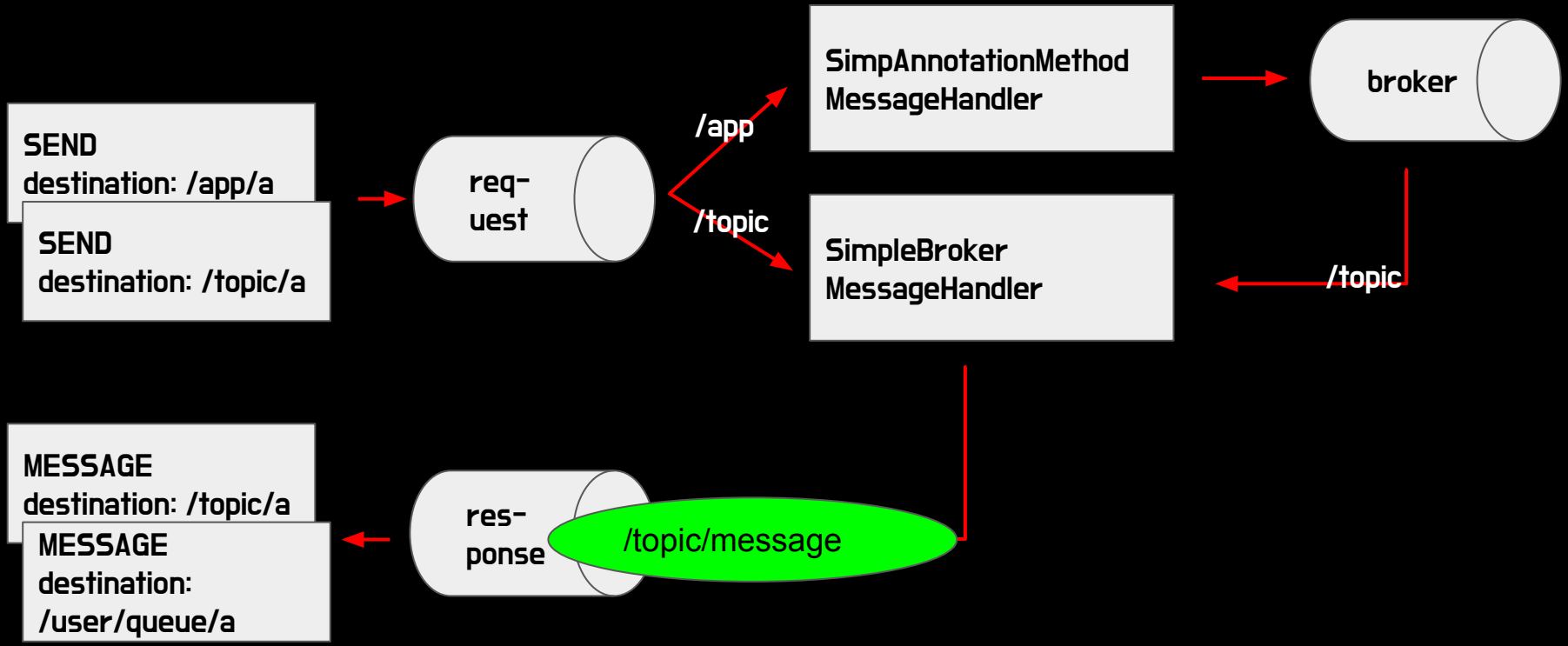

SEND
/app/event
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
SimpleBroker
MessageHandler
broker
/topic
MESSAGE
destination: /topic/a
MESSAGE
destination:
/user/queue/a
res-
ponse

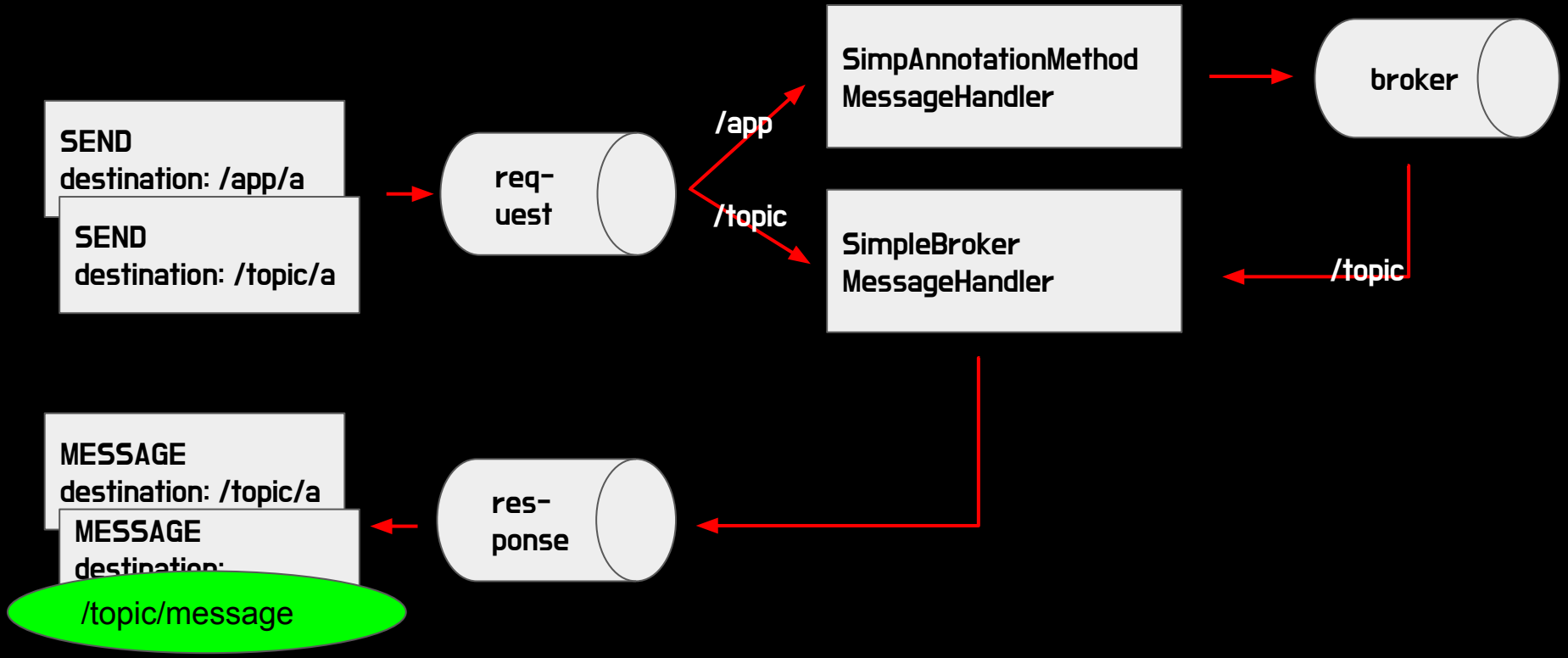

SEND
destination: /app/a
SEND
destination: /topic/a
/app/event
/app
/topic
SimpAnnotationMethod
MessageHandler
broker
SimpleBroker
MessageHandler
/topic
MESSAGE
destination: /topic/a
MESSAGE
destination:
/user/queue/a
res-
ponse


/app/event
SimpAnnotationMethod
MessageHandler
broker
SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpleBroker
MessageHandler
/topic
MESSAGE
destination: /topic/a
MESSAGE
destination:
/user/queue/a
res-
ponse


SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
SimpleBroker
MessageHandler
/topic/message
broker
/topic
MESSAGE
destination: /topic/a
MESSAGE
destination:
/user/queue/a
res-
ponse


SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
broker
SimpleBroker
MessageHandler
/topic
/topic/message
MESSAGE
destination: /topic/a
MESSAGE
destination:
/user/queue/a
res-
ponse


SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
SimpleBroker
MessageHandler
broker
/topic
MESSAGE
destination: /topic/a
MESSAGE
destination:
/user/queue/a
res-
ponse
/topic/message


SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
SimpleBroker
MessageHandler
broker
/topic
res-
ponse
MESSAGE
destination: /topic/a
MESSAGE
destination:
/topic/message

개념적인건 이정도로하고
나머지는 간단히 훑어보겠습니다.

```java
@Controller
@MessageMapping("ksug")
class HelloController {
    @MessageMapping("springcamp.{year}")
    @SendTo("/topic/message")
    public String message1(@DestinationVariable int year) {
        return format("welcome ksug %d springcamp", year);
    }

    @MessageMapping("echo")
    @SendToUser("/queue/message")
    public String message2(String message) {
        return format("echo '%s'", message);
    }

    @MessageExceptionHandler
    @SendToUser("/queue/error")
    public String handleException(Exception exception) {
        return exception.toString();
    }
}
```

```java
@MessageMapping("echo-every")
public void echoToEvery(String echo) {
    this.template.convertAndSend( destination: "/topic/message", echo);
}

@MessageMapping("echo-one")
public void echoToOne(Principal principal, String echo) {
    this.template.convertAndSendToUser(principal.getName(), destination: "/queue/message", echo);
}
```

security

```java
@Controller
class SecurityController {

    @MessageMapping("user")
    @SendToUser("/queue/message")
    public Principal echo(Principal principal) {
        return principal;
    }
}
```

Adding CSRF to Stomp Headers

By default Spring Security requires the CSRF token in any CONNECT message type. This ensures that only a site that has access to the CSRF token can connect. Since only the **Same Origin** can access the CSRF token, external domains are not allowed to make a connection.

Typically we need to include the CSRF token in an HTTP header or an HTTP parameter. However, SockJS does not allow for these options. Instead, we must include the token in the Stomp headers

Applications can obtain a CSRF token by accessing the request attribute named _csrf. For example, the following will allow accessing the `CsrfToken` in a JSP:

```
var headerName = "${_csrf.headerName}";
var token = "${_csrf.token}";
```

If you are using static HTML, you can expose the `CsrfToken` on a REST endpoint. For example, the following would expose the `CsrfToken` on the URL /csrf

```
@RestController
public class CsrfController {

    @RequestMapping("/csrf")
    public CsrfToken csrf(CsrfToken token) {
        return token;
    }
}
```

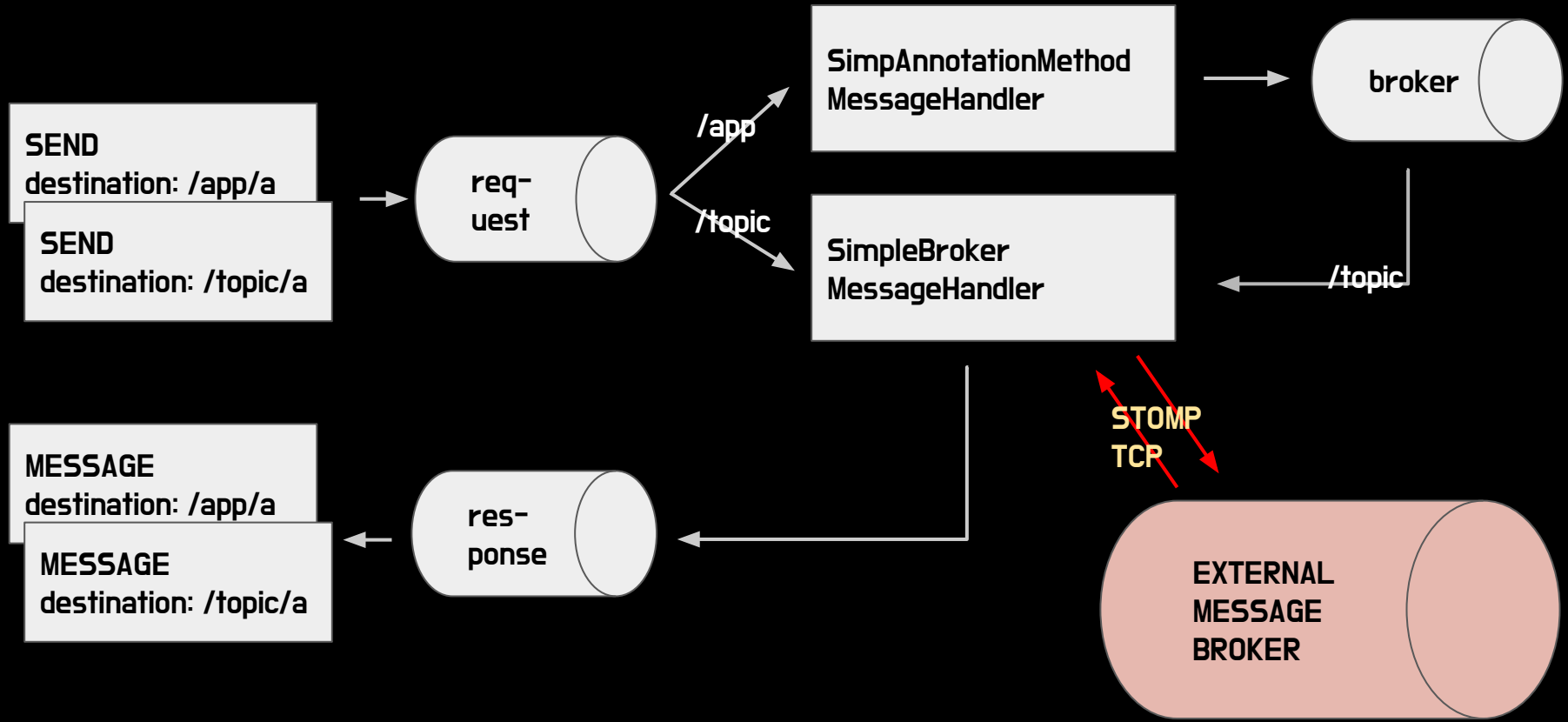
The JavaScript can make a REST call to the endpoint and use the response to populate the headerName and the token.

We can now include the token in our Stomp client. For example:

```
...
var headers = {};
headers[headerName] = token;
stompClient.connect(headers, function(frame) {
  ...

}
```

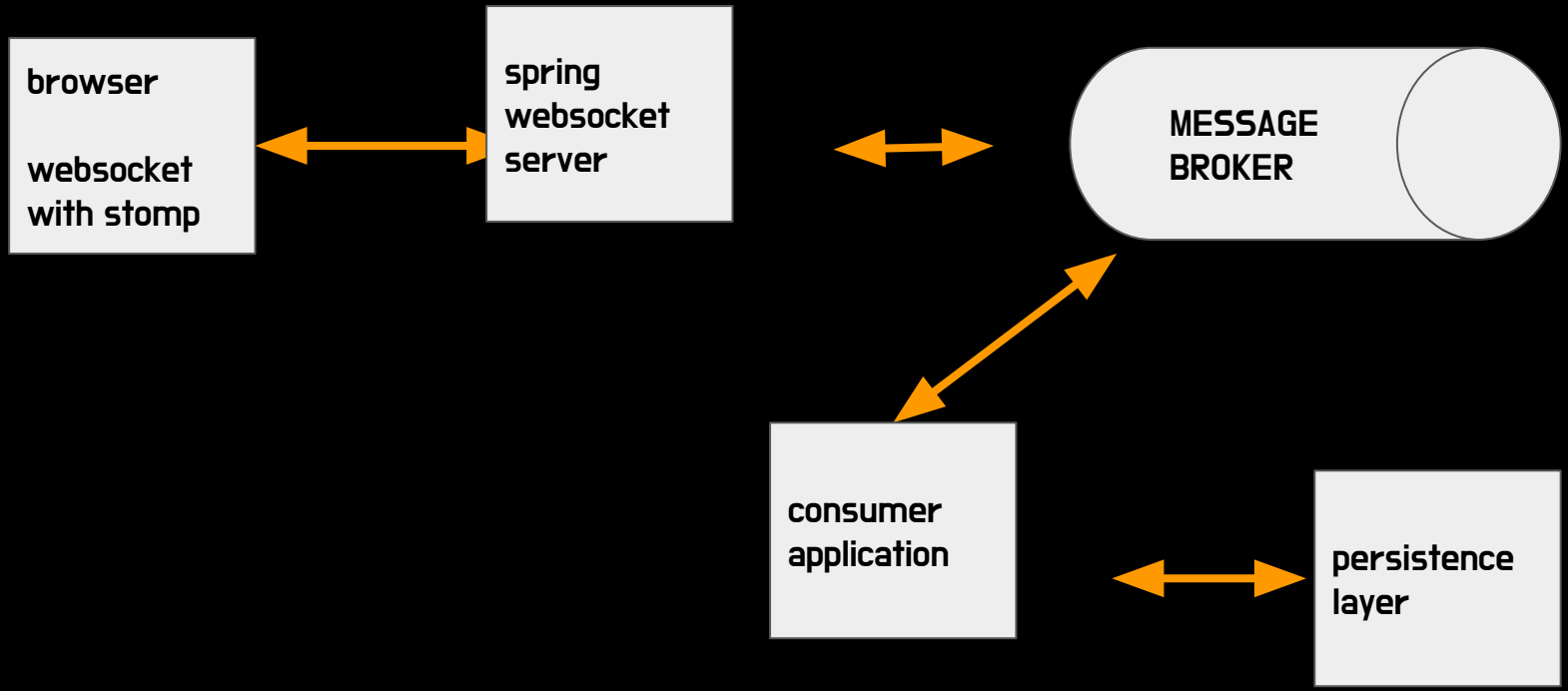
Scaleout

메세지 흐름

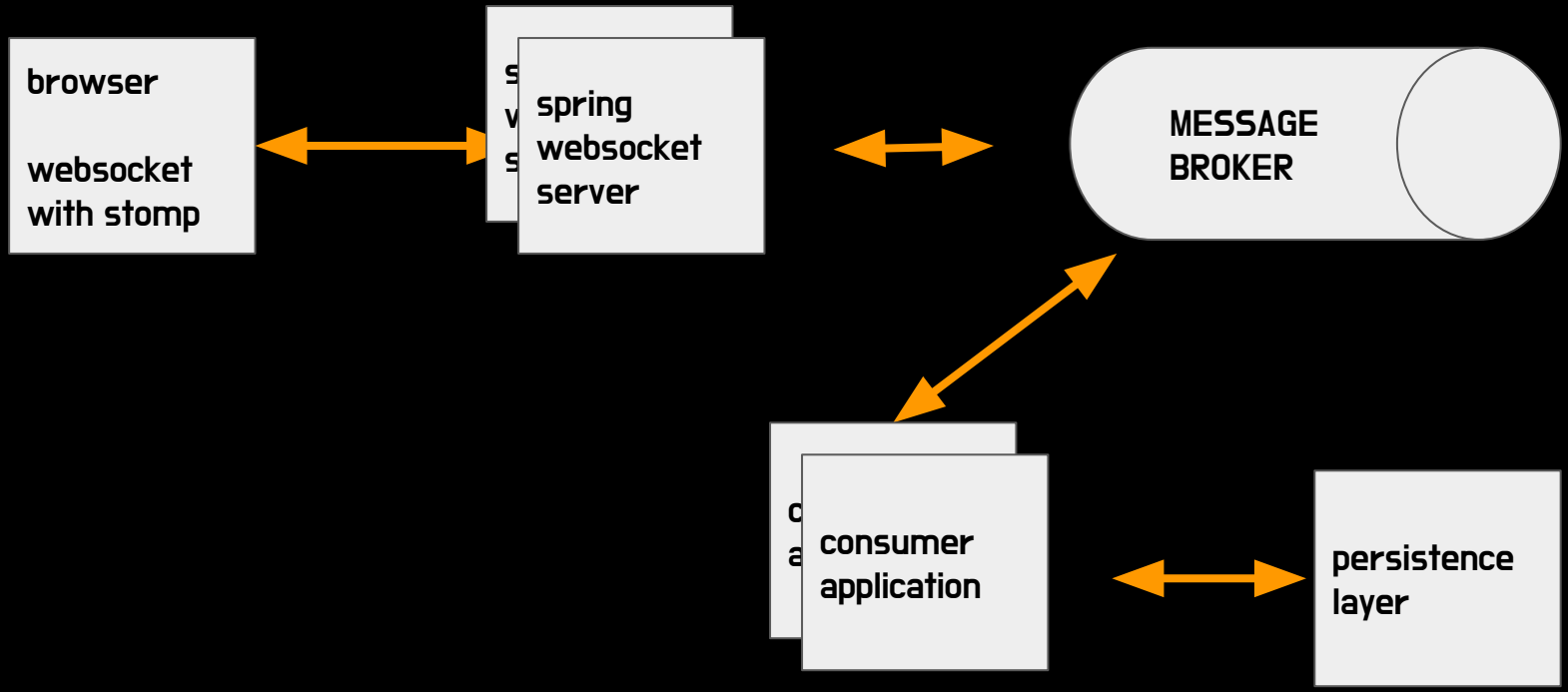

SEND
destination: /app/a
SEND
destination: /topic/a
req-
uest
/app
/topic
SimpAnnotationMethod
MessageHandler
SimpleBroker
MessageHandler
broker
/topic
STOMP
TCP
EXTERNAL
MESSAGE
BROKER
res-
ponse
MESSAGE
destination: /app/a
MESSAGE
destination: /topic/a

```java
@Configuration
@EnableWebSocketMessageBroker
class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint( ...paths: "/stomp").withSockJS();
    }

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.setPathMatcher(new AntPathMatcher( pathSeparator: "."));
        config.setApplicationDestinationPrefixes("/app");
        config.enableStompBrokerRelay( ...destinationPrefixes: "/topic", "/queue")
                .setRelayHost( "192.168.0.3" );
    }
}
```

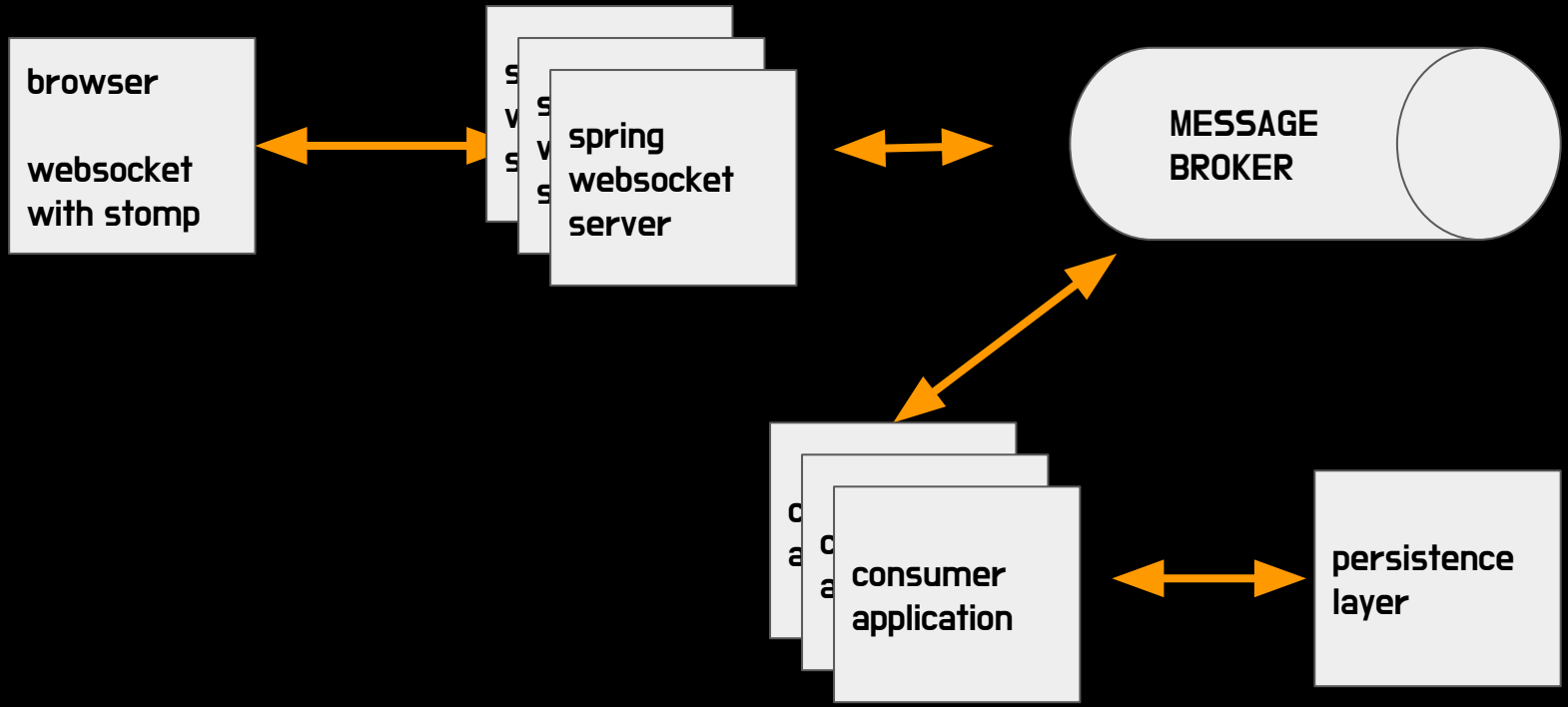
이런식으로 구성 가능


browser
websocket
with stomp
spring
websocket
server
MESSAGE
BROKER
consumer
application
persistence
layer

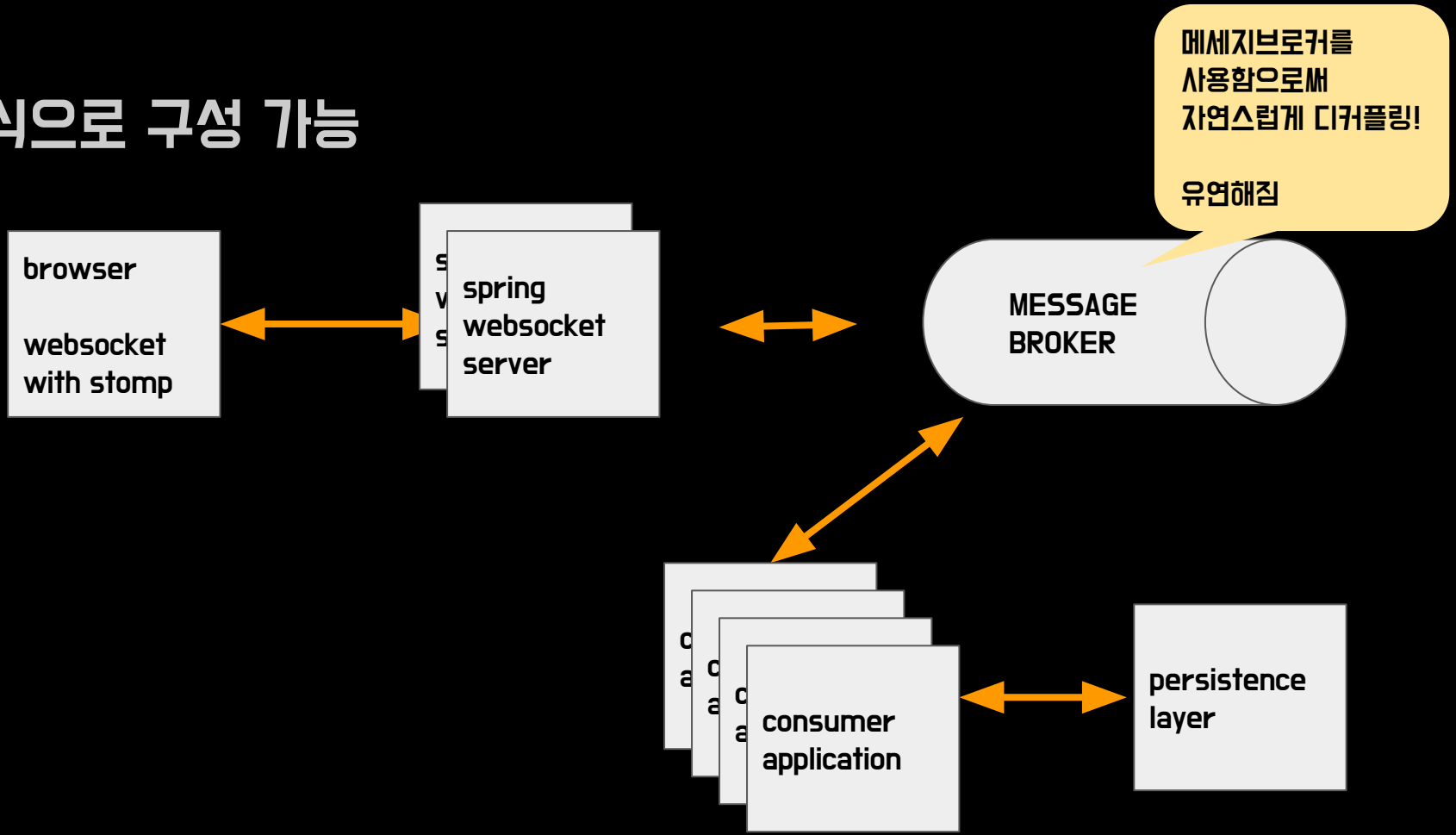
이런식으로 구성 가능

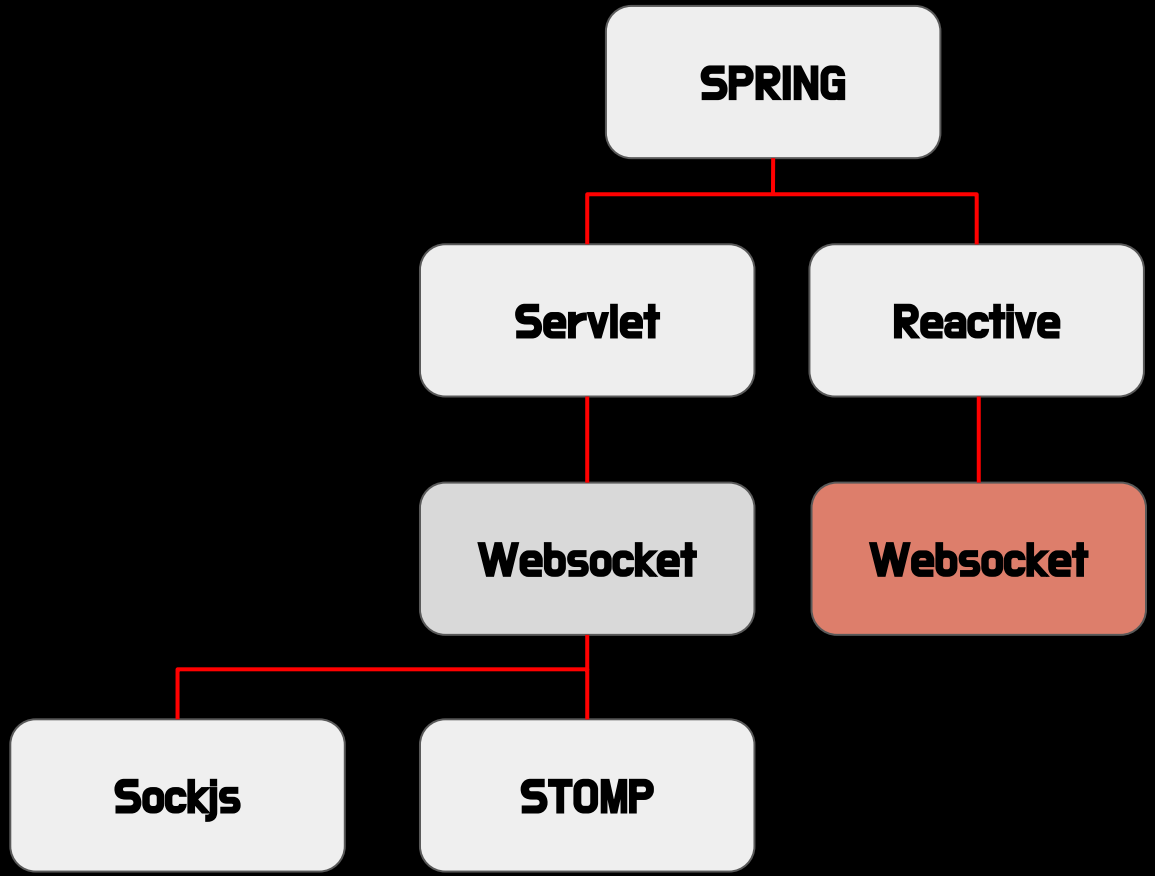

browser
websocket
with stomp
spring
websocket
server
MESSAGE
BROKER
consumer
application
persistence
layer

이런식으로 구성 가능


browser
websocket
with stomp
spring
websocket
server
MESSAGE
BROKER
consumer
application
persistence
layer

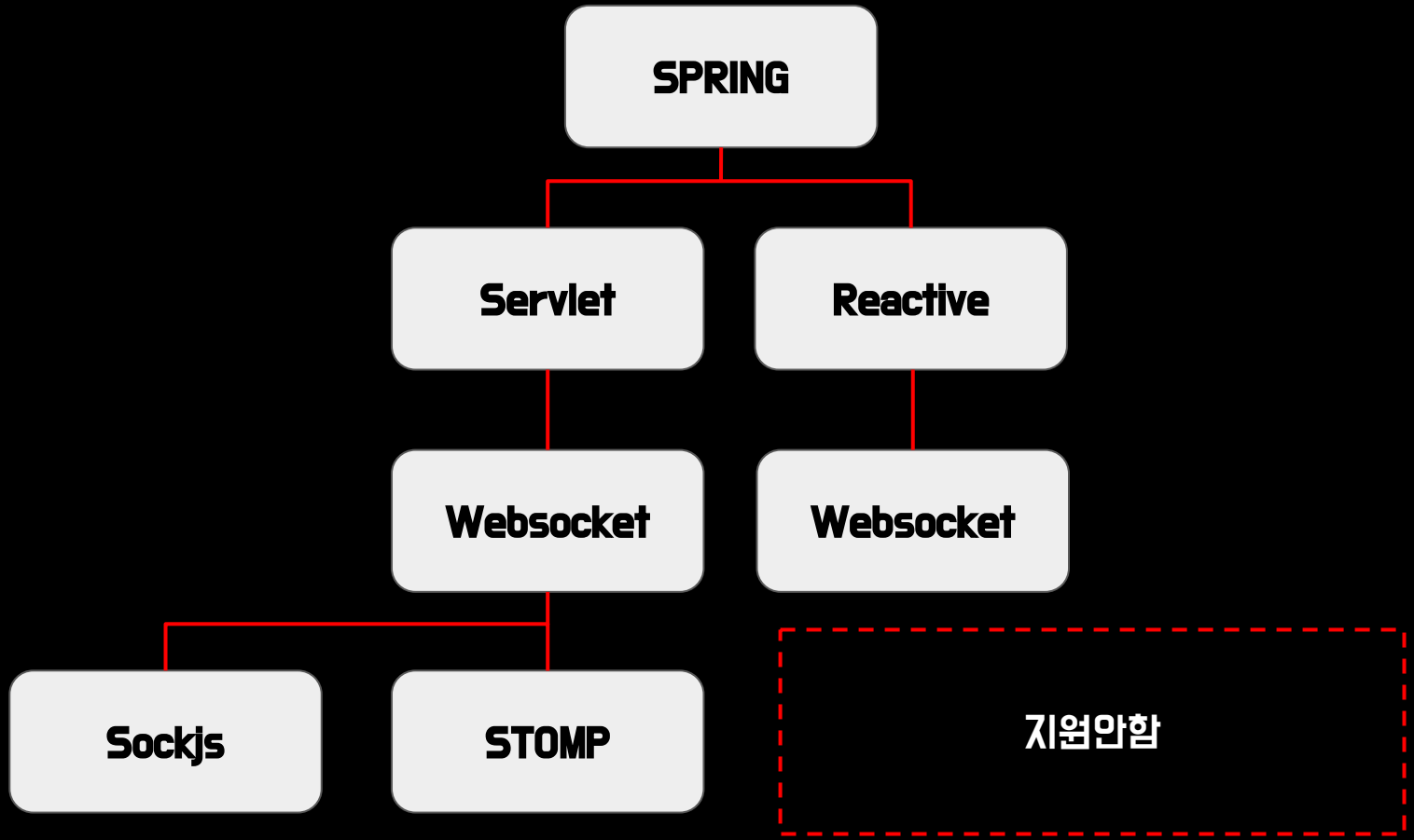
이런식으로 구성 가능


메세지브로커를
사용함으로써
자연스럽게 디커플링!
유연해짐
browser
websocket
with stomp
spring
websocket
server
MESSAGE
BROKER
consumer
application
persistence
layer


SPRING
Servlet
Reactive
Websocket
Websocket
Sockjs
STOMP

```java
@Configuration
class WebConfig {

    @Bean
    public HandlerMapping handlerMapping() {
        Map<String, WebSocketHandler> map = new HashMap<>();
        map.put("/websocket", new SpringWebSocketHandler());

        SimpleUrlHandlerMapping mapping = new SimpleUrlHandlerMapping();
        mapping.setUrlMap(map);
        mapping.setOrder(Ordered.HIGHEST_PRECEDENCE);
        return mapping;
    }

    @Bean
    public WebSocketHandlerAdapter handlerAdapter() { return new WebSocketHandlerAdapter(); }
}
```

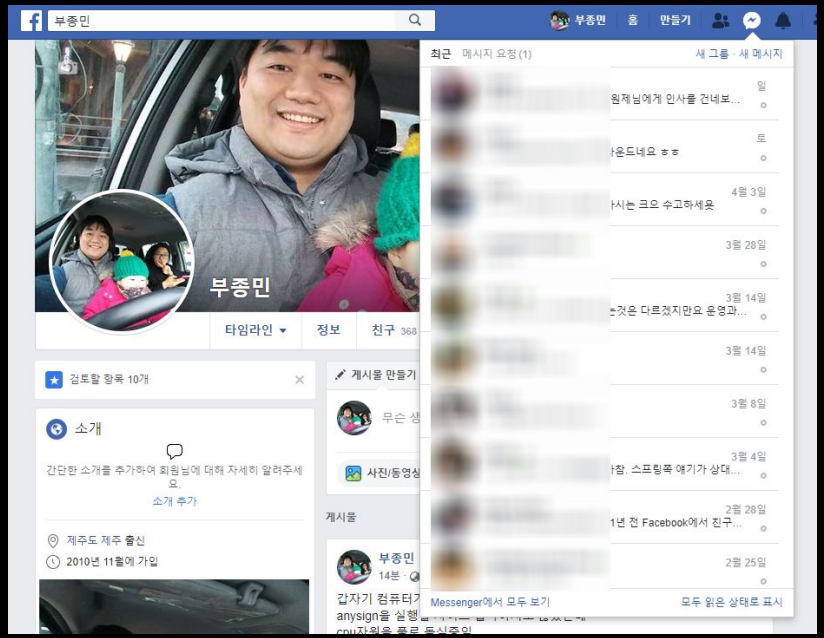
```
class SpringWebSocketHandler implements WebSocketHandler {
    private Set<WebSocketSession> sessions = new ConcurrentHashMap().newKeySet();

    @Override
    public Mono<Void> handle(WebSocketSession session) {
        Flux<WebSocketMessage> out = session.receive()
                .doOnSubscribe(x -> {
                    sessions.add(session);
                    System.out.println("new session: " + sessions.size());
                })
                .map(message -> session.textMessage( payload: "hello client: " + (Math.random() * 10)))
                .doOnCancel(() -> System.out.println("canceled"))
                .doOnError(e -> e.printStackTrace())
                .doOnComplete(() -> {
                    sessions.remove(session);
                    System.out.println("completed: " +
                            "" + sessions.size());
                });
        return session.send(out);
    }
}
```

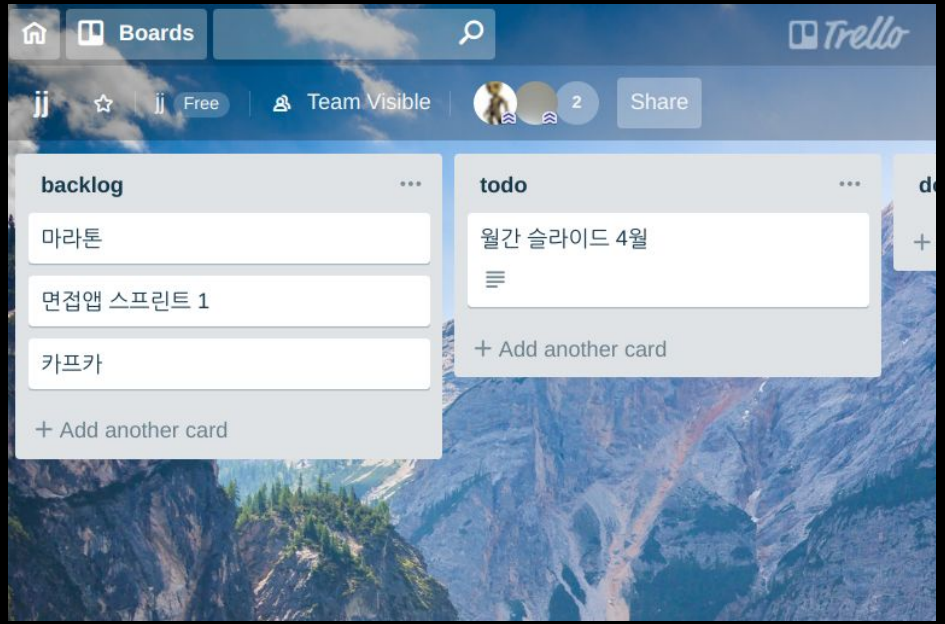

SPRING
Servlet
Reactive
Websocket
Websocket
Sockjs
STOMP
지원안함


websocket

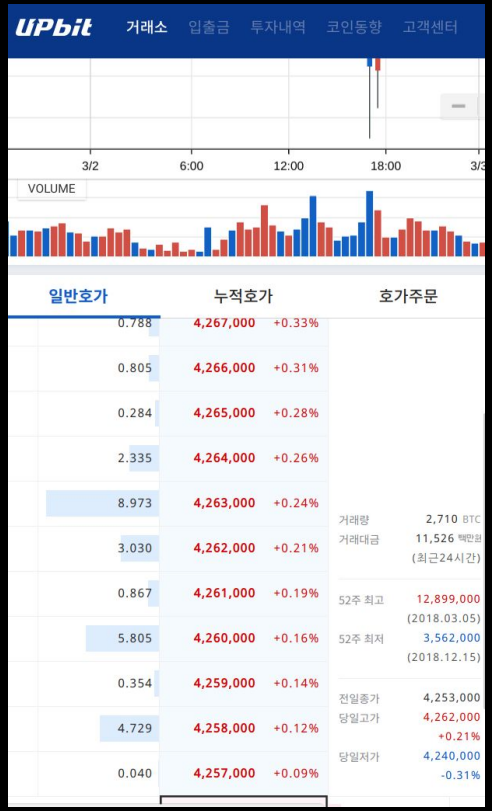
웹소켓을 사용하는 서비스

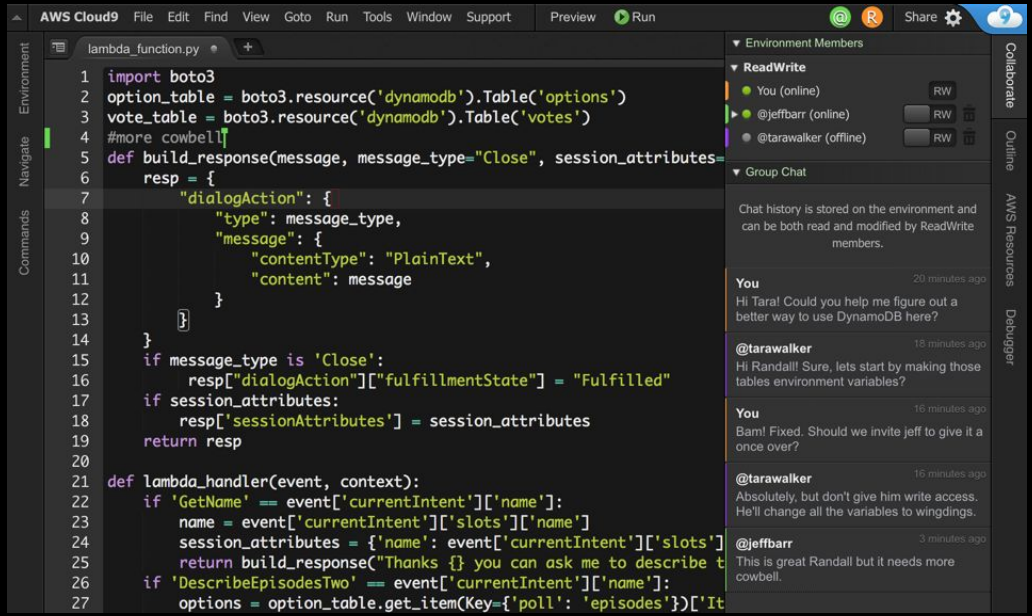
웹소켓을 사용하는 서비스



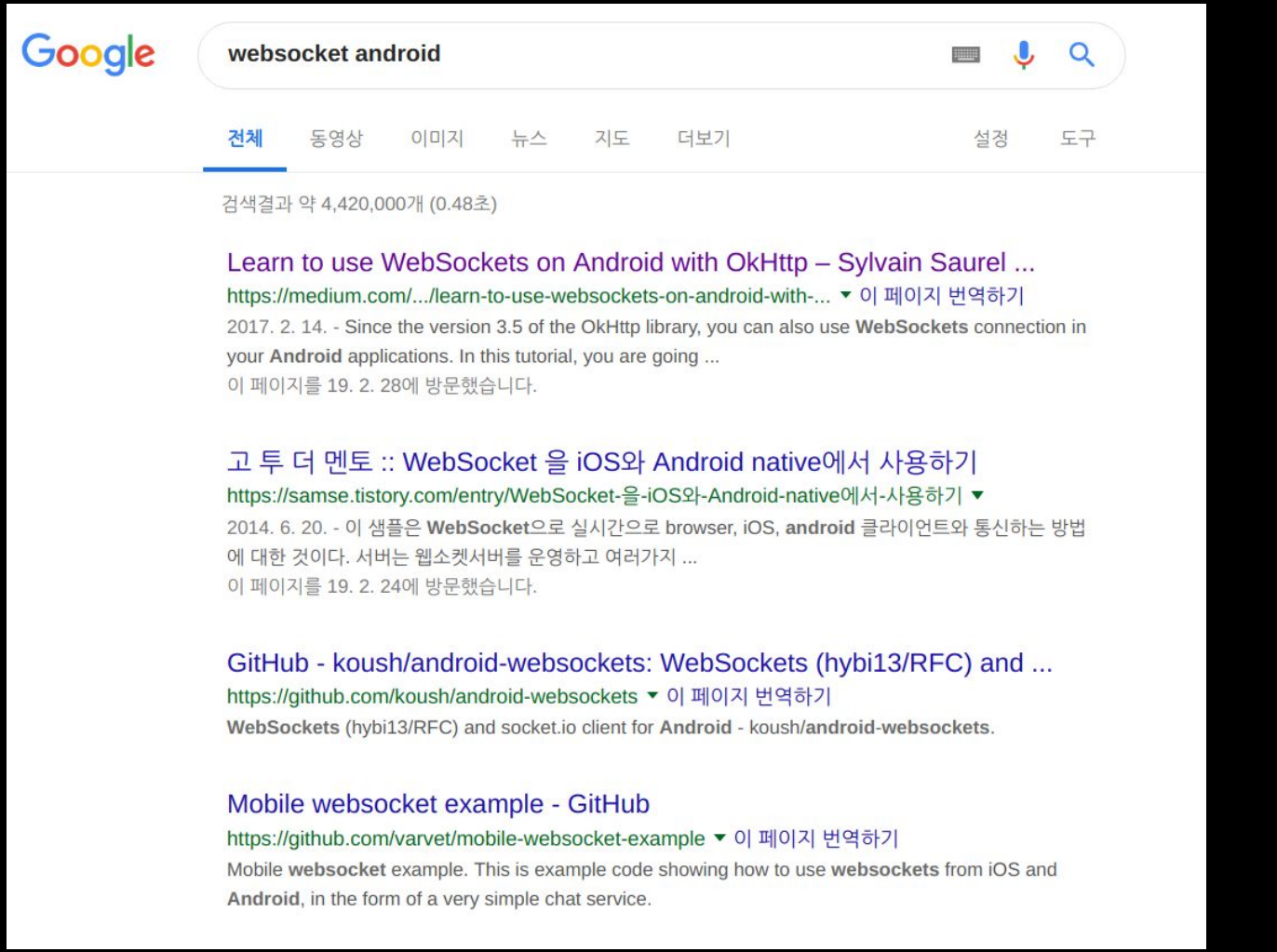


웹소켓을 사용하는 서비스





웹소켓은 웹브라우저 전용 기술?

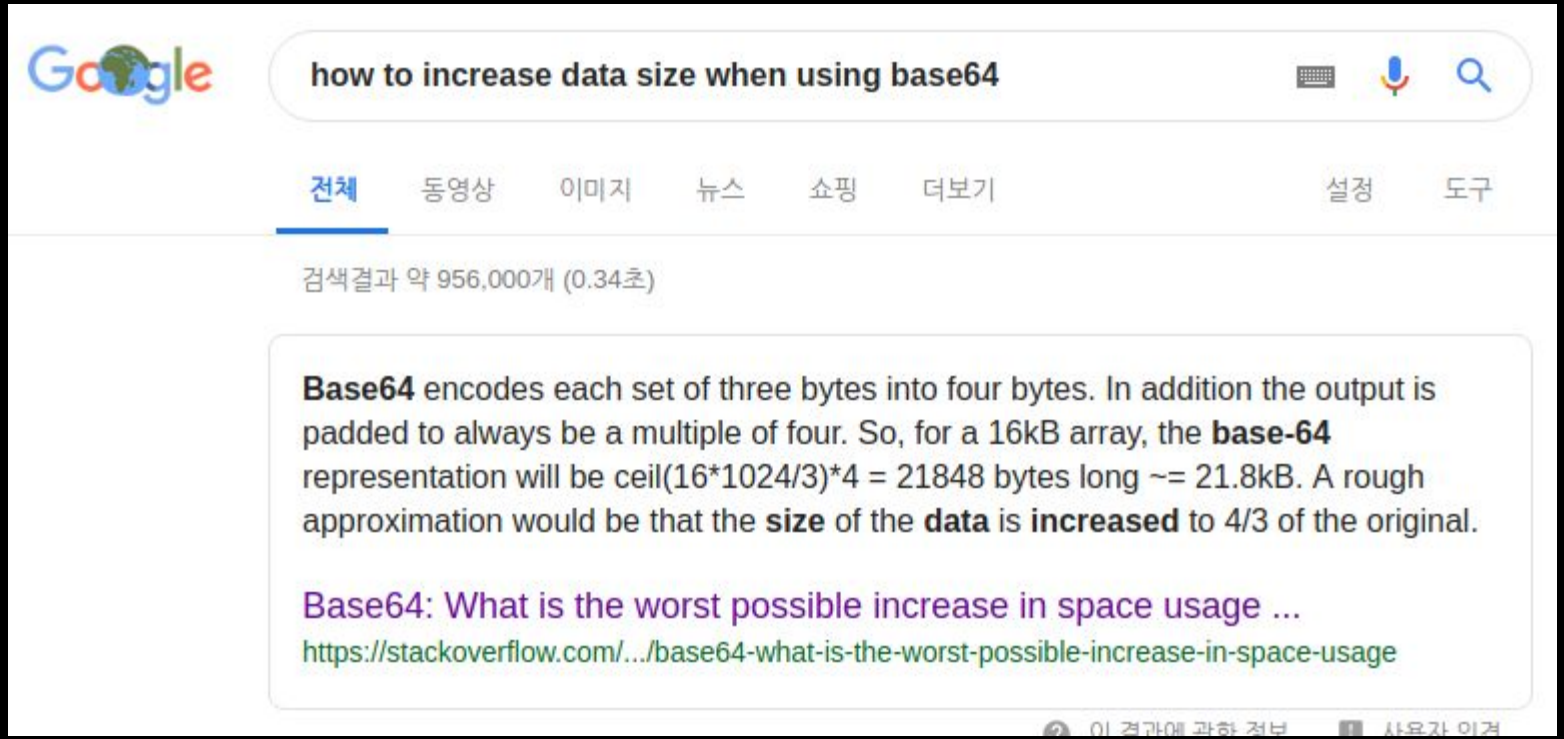

Google
websocket android
전체
동영상
이미지
뉴스
지도
더보기
설정
도구
검색결과 약 4,420,000개 (0.48초)
Learn to use WebSockets on Android with OkHttp – Sylvain Saurel ...
https://medium.com/.../learn-to-use-websockets-on-android-with-... 이 페이지 번역하기
2017. 2. 14. - Since the version 3.5 of the OkHttp library, you can also use WebSockets connection in your Android applications. In this tutorial, you are going ...
이 페이지를 19. 2. 28에 방문했습니다.
고 투 더 멘토 :: WebSocket 을 iOS와 Android native에서 사용하기
https://samse.tistory.com/entry/WebSocket-을-iOS와-Android-native에서-사용하기
2014. 6. 20. - 이 샘플은 WebSocket으로 실시간으로 browser, iOS, android 클라이언트와 통신하는 방법에 대한 것이다. 서버는 웹소켓서버를 운영하고 여러가지 ...
이 페이지를 19. 2. 24에 방문했습니다.
GitHub - koush/android-websockets: WebSockets (hybi13/RFC) and ...
https://github.com/koush/android-websockets 이 페이지 번역하기
WebSockets (hybi13/RFC) and socket.io client for Android - koush/android-websockets.
Mobile websocket example - GitHub
https://github.com/varvet/mobile-websocket-example 이 페이지 번역하기
Mobile websocket example. This is example code showing how to use websockets from iOS and Android, in the form of a very simple chat service.

binary 데이터 전송

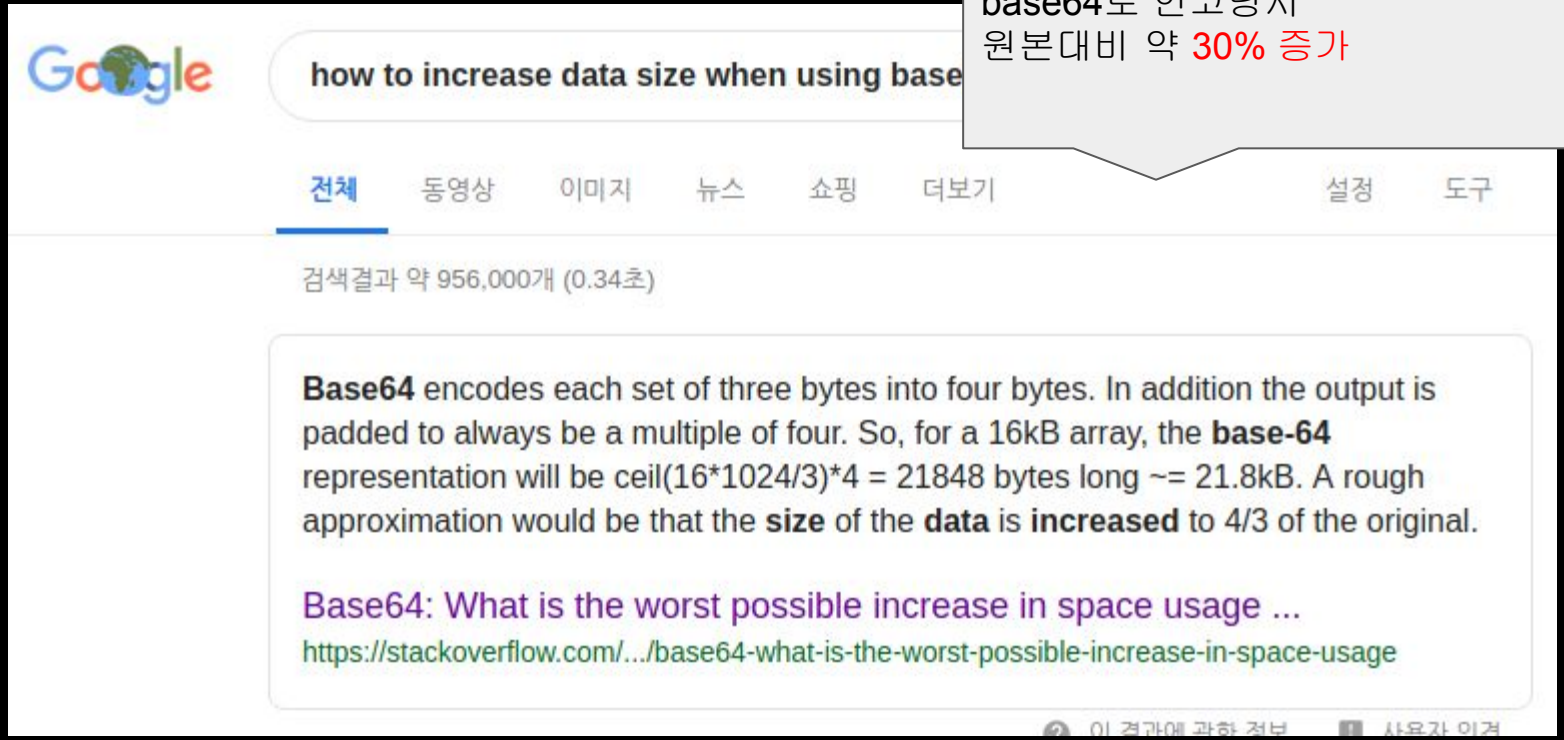
ajax로 바이너리 데이터 전송시
보통 base64를 이용하여 인코딩.

보통 암호화하여 데이터 주고 받을때
경험


```
{
 "name": "boojongmin",
 "hello": "world",
"base64":"7JWI64WV7ZWY7IS47JqULg0KDQrquIDroZzrsowg7Zes7Iqk7LyA7Ja0IOyerO2ZnCDsh
pTro6jshZgg7JeF7LK07J24IOuEpOyYpO2Ome2KuOyXkOyEnCBYWFgg6rCc67Cc7J6Q66W8IOy
xhOyaqSDspJHsl5Ag7J6I7Iq164uI64ukLg0K"

}
```


how to increase data size when using base64
전체
동영상
이미지
뉴스
쇼핑
더보기
설정
도구
검색결과 약 956,000개 (0.34초)
Base64 encodes each set of three bytes into four bytes. In addition the output is padded to always be a multiple of four. So, for a 16kB array, the base-64 representation will be ceil(16*1024/3)*4 = 21848 bytes long ~= 21.8kB. A rough approximation would be that the size of the data is increased to 4/3 of the original.
Base64: What is the worst possible increase in space usage ...
https://stackoverflow.com/.../base64-what-is-the-worst-possible-increase-in-space-usage

base64로 인코딩시
원본대비 약 30% 증가



socket api

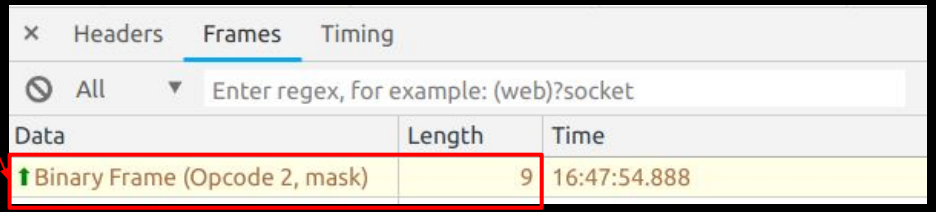
서버에 데이터 전송하기

한번 연결이 수립되면 이제부터는 서버로 데이터를 전송할 수 있습니다. 이것을 하기 위해서는 단순히 `WebSocket` 오브젝트의 `send()` 호출하여 보내고 싶은 메세지를 지정하기만 하면 됩니다.:

```
1 | exampleSocket.send("Here's some text that the server is urgently awaiting!");
```

보낼 수 있는 데이터는 Strin, `Blob`, 또는 `ArrayBuffer` 입니다.

```
60
61
62 const inputElement = document.getElementById("input");
63
64 inputElement.addEventListener("change", handleFiles);
65 function handleFiles() {
66     var reader = new FileReader();
67     reader.addEventListener('load', function() {
68         ws2.send(reader.result);
69     });
70     reader.readAsArrayBuffer(inputElement.files[0]);
71 }
72
73
74
75 </script>
```



rest api로 개발하는 것보다
웹소켓으로 개발하는게 불편하다?

프로젝트 구조

```
├── client
│   ├── node_modules
│   ├── public
│   └── src
├── server
│   ├── gradle
│   └── src
└── worker
    ├── gradle
    └── src
```

client

stomp

server

stomp

Java

worker

stomp

RabbitMQ

client

stomp

server

브라우저 - 서버 모두
stomp를 이용하여
pub/sub로 동작

stomp

Java

worker

stomp

RabbitMQ

client

stomp

server

stomp

fire and forgot

Java

worker

stomp

RabbitMQ

client

stomp

server

stomp

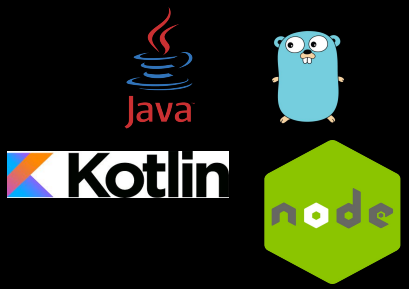
Java

worker

stomp

client

server



worker

client

stomp

server

메세지큐를 기반으로
메세지를 주고받으므로
worker는 언어 독립적으로 구현 가능

stomp

worker

stomp

간단히 코드로 보면

client

```
.$stomp
    .send(
      'DEPARTMENT.LIST',
      {page: 1}
    )
```

pub

```
mutations: {
  'DEPARTMENT.LIST'(state, data) {
    state.department = data;
  },
getters: {
  'DEPARTMENT.LIST': state => {
    return state.department;
  },
```

sub

server

```java
@MessageMapping("/event")
public void event(Principal principal, Event event) {
    String username = principal.getName();
    event.setUsername(username);
    service.send(event);
}
```

sub

pub

```java
@RabbitHandler
public void receive(String in) {
    Map map = null;
    try {
        map = mapper.readValue(in, Map.class);
        String username = (String)map.get("username");
        messagingTemplate.convertAndSendToUser(username, destination, in);
    } catch (IOException e) {
        logger.error(format("message: %s", in));
    }
}
```

sub

pub

server

```java
@MessageMapping("/event")
public void event(Principal principal, Event event) {
    String username = principal.getName();
    event.setUsername(username);
    service.send(event);
}
```

server는 단순히 worker에게
이벤트를 전달만해주는 역할.

한번 만들어 두면 코드 추가가 없음.

```java
@RabbitHandler
public void receive(String in) {
    Map map = null;
    try {
        map = mapper.readValue(in, Map.clas
        String username = (String)map.get("username");
        messagingTemplate.convertAndSendToUser(username, destination, in);
    } catch (IOException e) {
        logger.error(format("message: %s", in));
    }
}
```

sub
pub

sub
pub

worker

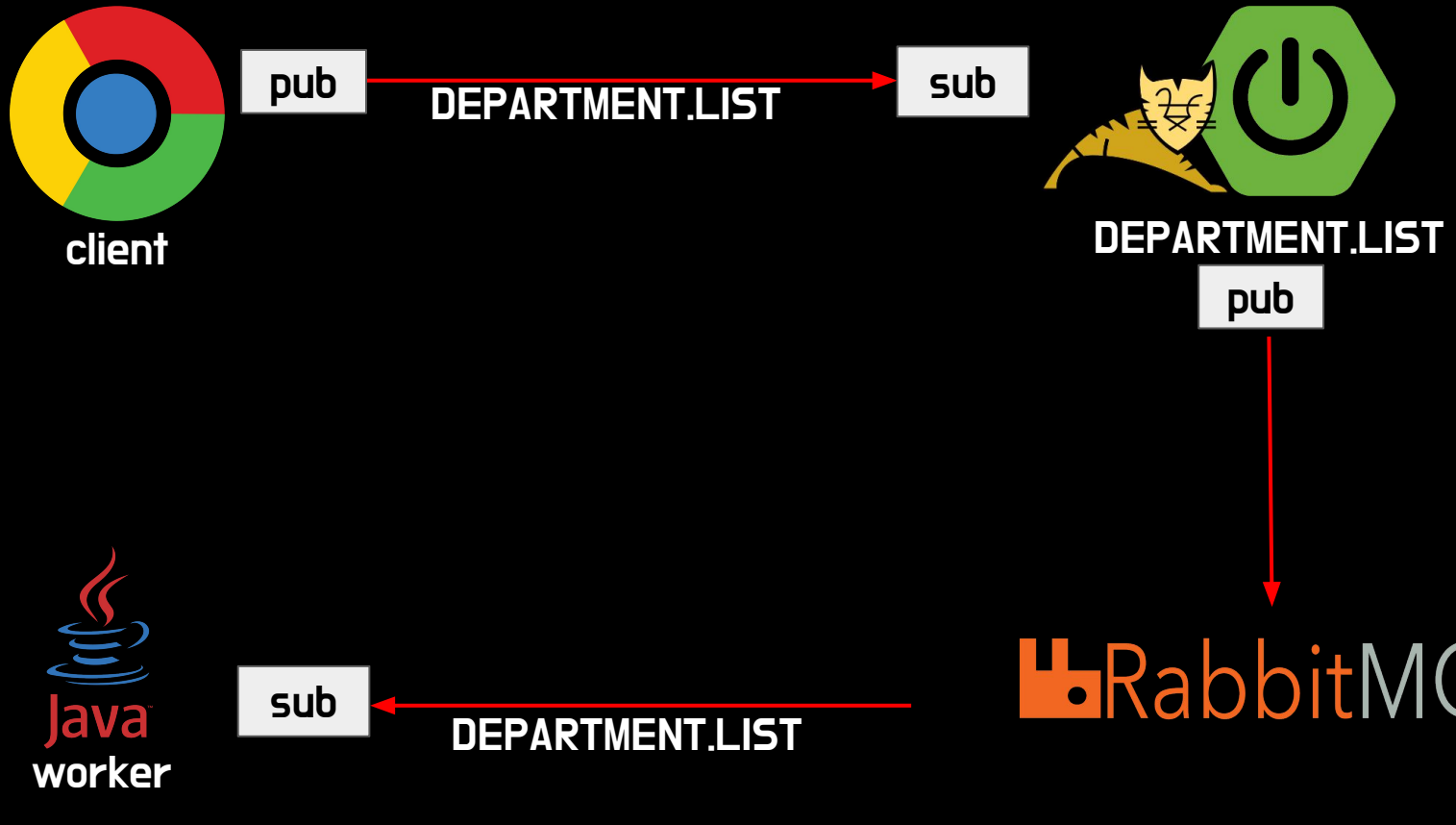
```kotlin
val response: Any? = when(event.type) {
    "$DEPARTMENT.LIST" -> companyService.departments(event)
    "$DEPARTMENT.CREATE" -> companyService.createDepartment(event)
    "$DEPARTMENT.DETAIL" -> companyService.detail(event)
    else -> undefinedType(event.type)
}
```

sub

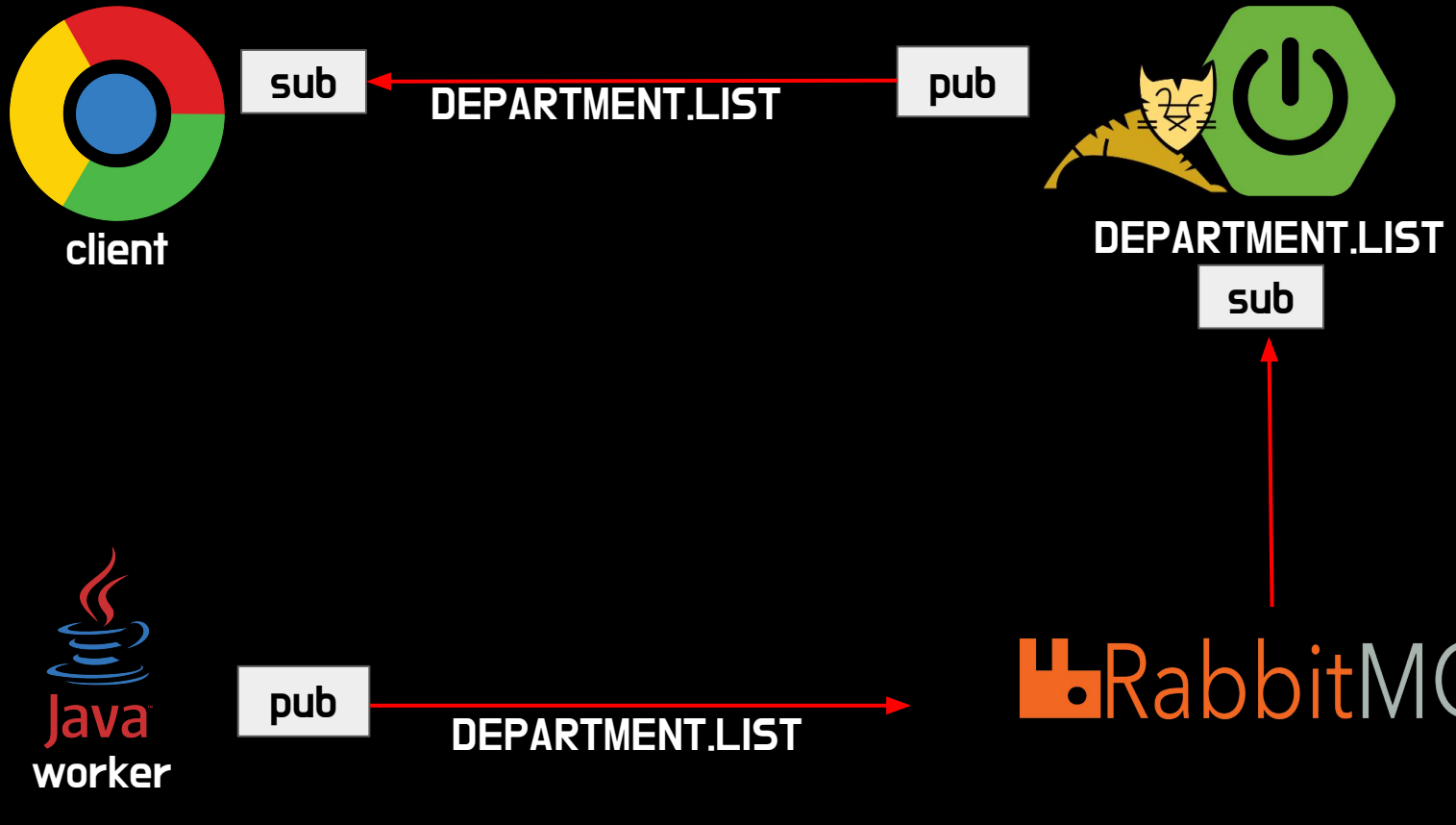
```kotlin
if(response != null) {
    event.data = response
    val eventJson = mapper.writeValueAsString(event)
    template.convertAndSend(queueName, eventJson)
}
```

pub

메세지의 흐름 - 클라이언트 요청



메세지의 흐름 - 서버의 응답



그리고 기존 ajax를 호출하는 방식과 비교해보면

기존의 ajax 방식

```
$.axios.get('/department')
  .then((data) => {
    $store.dispatch('DEPARTMENT.LIST', data)
  })
  .catch(() => {
    // TODO handle error
  })
```

ajax

```
mutations: {
  'DEPARTMENT.LIST'(state, data) {
    state.department = data;
  },
},

getters: {
  'DEPARTMENT.LIST': state => {
    return state.department;
  },
},
```

```
actions: {
  departmentList (context, data) {
    context.commit('DEPARTMENT.LIST', data)
  }
}
```

vuex

기존의 ajax 방식

```
$.axios.get('/department')
  .then((data) => {
    $store.dispatch('DEPARTMENT.LIST', data)
  })
  .catch(() => {
    // TODO handle error
  })
```

ajax

```
mutations: {
  'DEPARTMENT.LIST'(state, data) {
    state.department = data;
  },

getters: {
  'DEPARTMENT.LIST': state => {
    return state.department;
  },
```

```
actions: {
  departmentList (context, data) {
    context.commit('DEPARTMENT.LIST', data)
  }
}
```

vuex

stomp를 이용한 vuejs

```
.$stomp
    .send(
      'DEPARTMENT.LIST',
      {page: 1}
    )
```

요청

```
mutations: {
  'DEPARTMENT.LIST'(state, data) {
    state.department = data;
  },
getters: {
  'DEPARTMENT.LIST': state => {
    return state.department;
  },
```

응답

stomp를 이용한 vuejs

```
.$stomp
    .send(
      'DEPARTMENT.LIST',
      {page: 1}
    )
```

동일한 이벤트명

요청

```
mutations: {
  'DEPARTMENT.LIST'(state, data) {
    state.department = data;
  },
getters: {
  'DEPARTMENT.LIST': state => {
    return state.department;
  },
```

응답

서버는 프로젝트 시작시
tomcat을 띄우고 component들 scan하고 이 와 관련한 작업을 하면서
5초 이상걸리는 등 서버 리스타트할 때 가볍지는 않았다.
(특히 프로젝트 규모가 커질수록 더 길어짐)

하지만 worker는 그냥 큐에 연결하고 간단한 구조라
jar자체도 작고
기동 시간도 빠름.(1초 미만)

startup 속도가 빠르다는건 개발할때 생산성에 영향을 준다고 생각.

그런

REST API로

괜찮은가

출처링크

정리

- 오늘날 대부분의 "REST API"는 사실 REST를 따르지 않고 있다.
- REST의 제약조건 중에서 특히 **Self-descriptive**와 **HATEOAS**를 잘 만족하지 못한다.
- REST는 **긴 시간에 걸쳐(수십년) 진화**하는 웹 애플리케이션을 위한 것이다.
- REST를 따를 것인지는 API를 설계하는 이들이 스스로 판단하여 결정해야한다.
- REST를 따르겠다면, **Self-descriptive**와 **HATEOAS**를 만족시켜야한다.
  - Self-descriptive는 **custom media type**이나 **profile link relation** 등으로 만족시킬 수 있다.
  - HATEOAS는 HTTP 헤더나 본문에 **링크**를 담아 만족시킬 수 있다.
- REST를 따르지 않겠다면, "REST를 만족하지 않는 REST API"를 뭐라고 부를지 결정해야 할 것이다.
  - HTTP API라고 부를 수도 있고
  - 그냥 이대로 REST API라고 부를 수도 있다. ~~(roy가 싫어합니다)~~

▼ Request Headers

⚠ **Provisional headers are shown**

charset: utf-8

Content-Type: application/x-www-form-urlencoded; charset=utf-8

Referer: https://blog.naver.com/PostView.nhn?blogId=naverschool&log
=221300970747&redirect=Dlog&widgetTypeCall=true&topReferer=https%3
2F%2Fwww.naver.com%2F&directAccess=false

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/5
7.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36

▼ Query String Parameters view source view URL encoded

blogId: naverschool

viewdate:

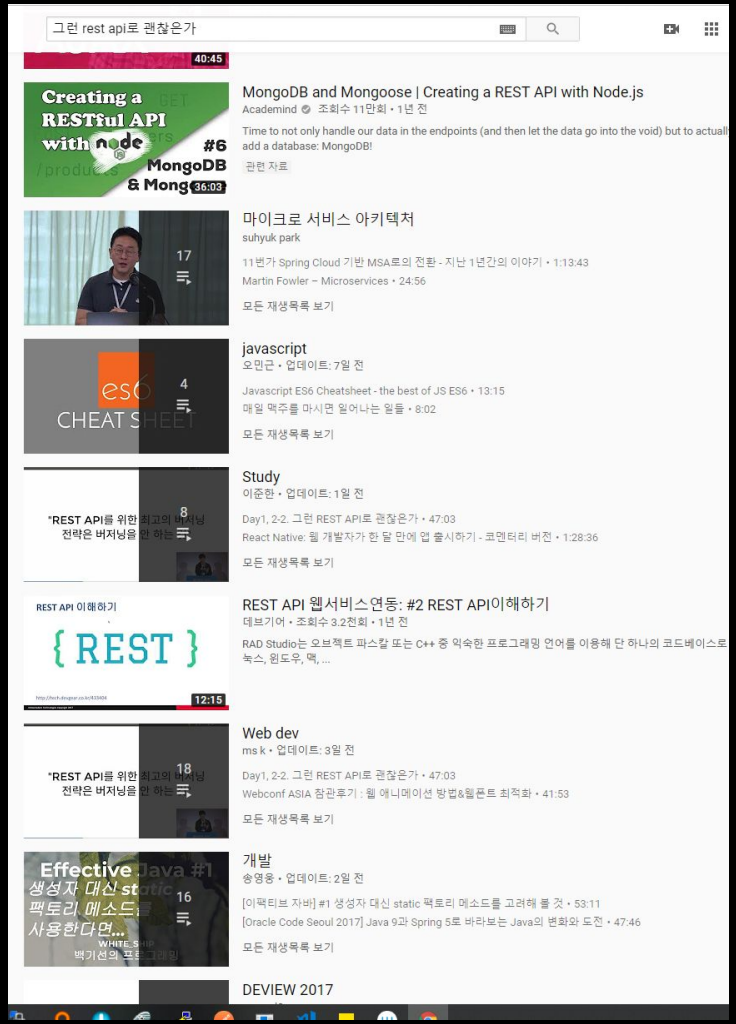
currentPage: 1

categoryNo: 21

parentCategoryNo: 0

countPerPage: 5


```
{
    "content": [ {
        "price": 499.00,
        "description": "Apple tablet device",
        "name": "iPad",
        "links": [ {
            "rel": "self",
            "href": "http://localhost:8080/product/1"
        } ],
        "attributes": {
            "connector": "socket"
        }
    }, {
        "price": 49.00,
        "description": "Dock for iPhone/iPad",
        "name": "Dock",
        "links": [ {
            "rel": "self",
            "href": "http://localhost:8080/product/3"
        } ],
        "attributes": {
            "connector": "plug"
        }
    } ],
    "links": [ {
        "rel": "product.search",
        "href": "http://localhost:8080/product/search"
    } ]
}
```



제대로 rest api를 공부하기 쉽지않다.

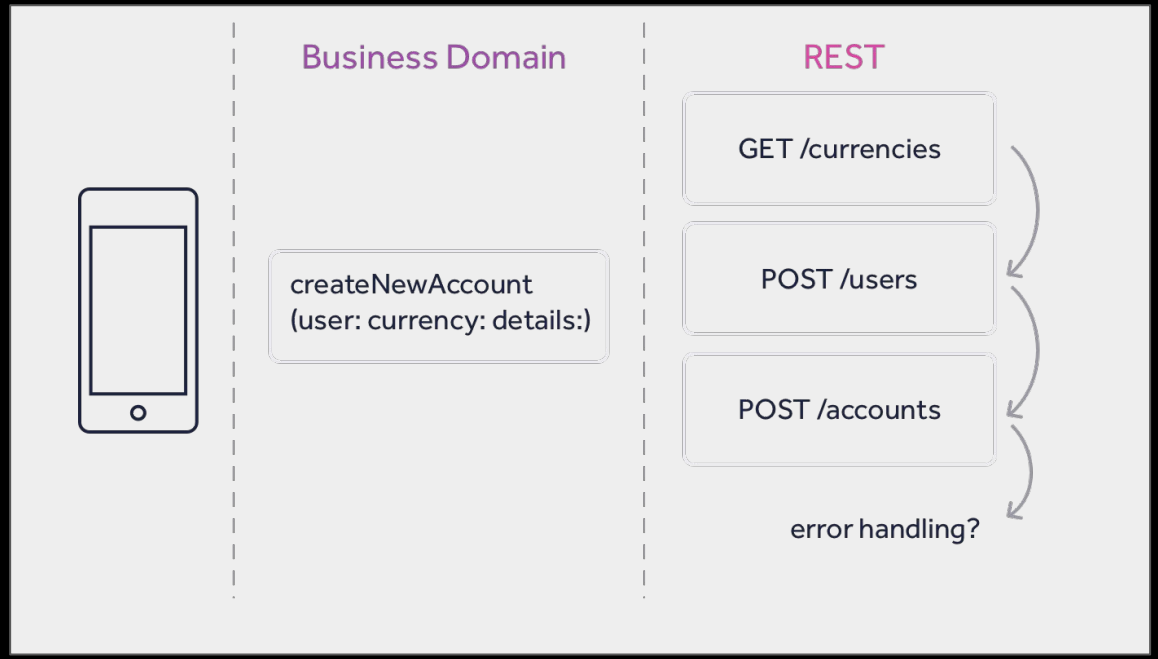
나도 아직 모르겠다고 생각.

만들고 나면 누가 이건 restful하지 않다고 트집잡음.

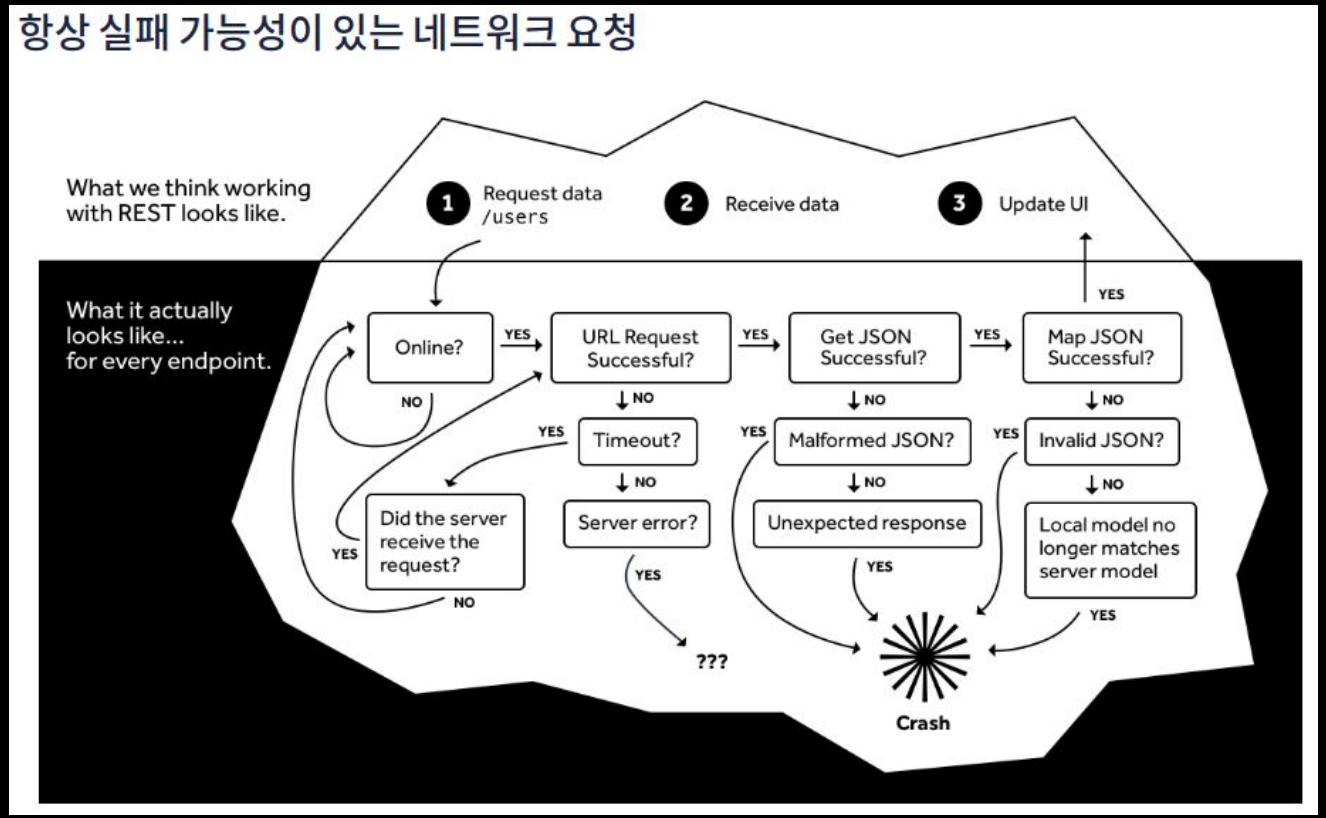
뭔가 아직 부족하다고 느낌.

rest api에 대해 클라이언트 입장도 들어보자

[Pain Points in Mobile Networking: REST API Failure]



rest api에 대해 클라이언트 입장도 들어보자



rest api에 대해 클라이언트 입장도 들어보자

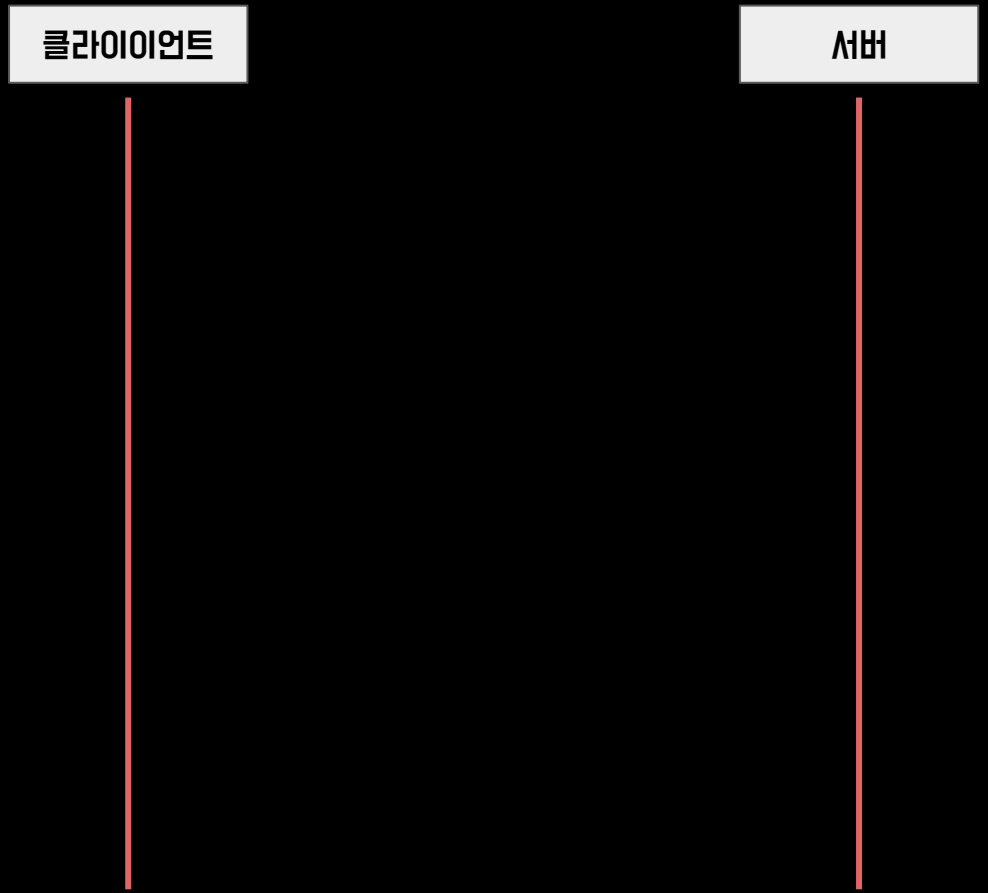
웹소켓을 통해 해결할 수 있는 내용은 아니지만 클라이언트 개발자가
restless란 단어에 대한 호감은 상당함.

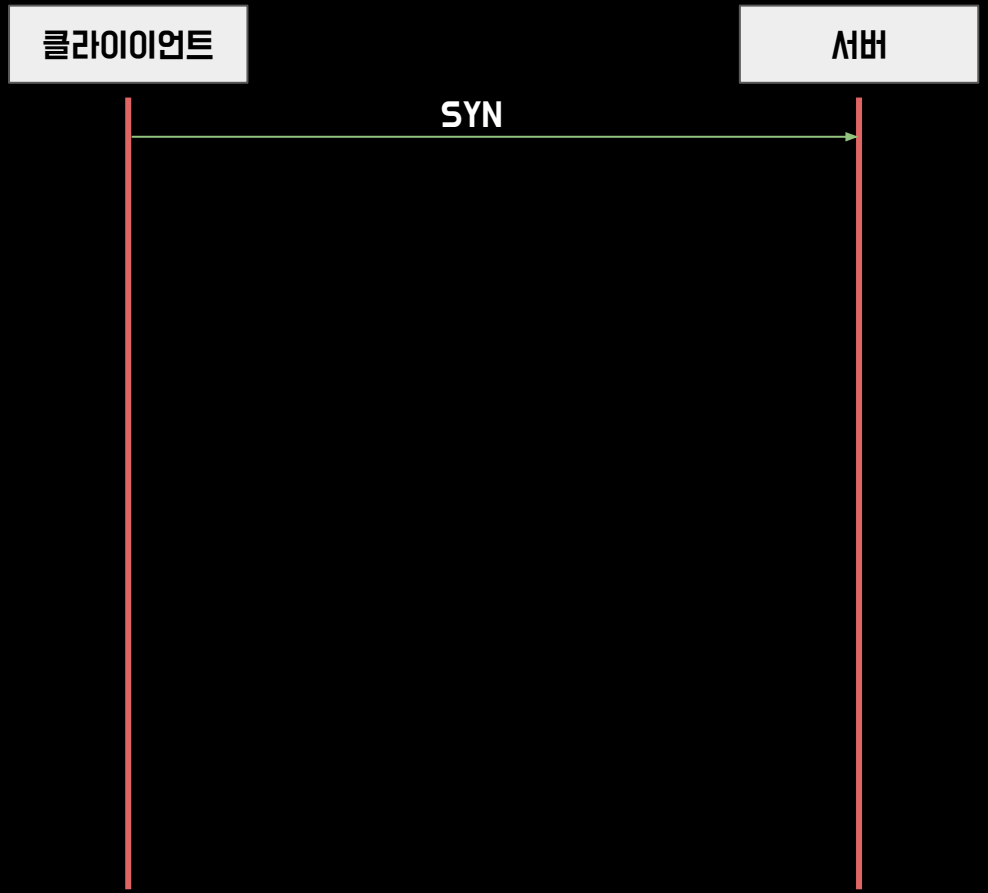
websocket + stomp를 쓰면
restful api
요청/응답 관점에서

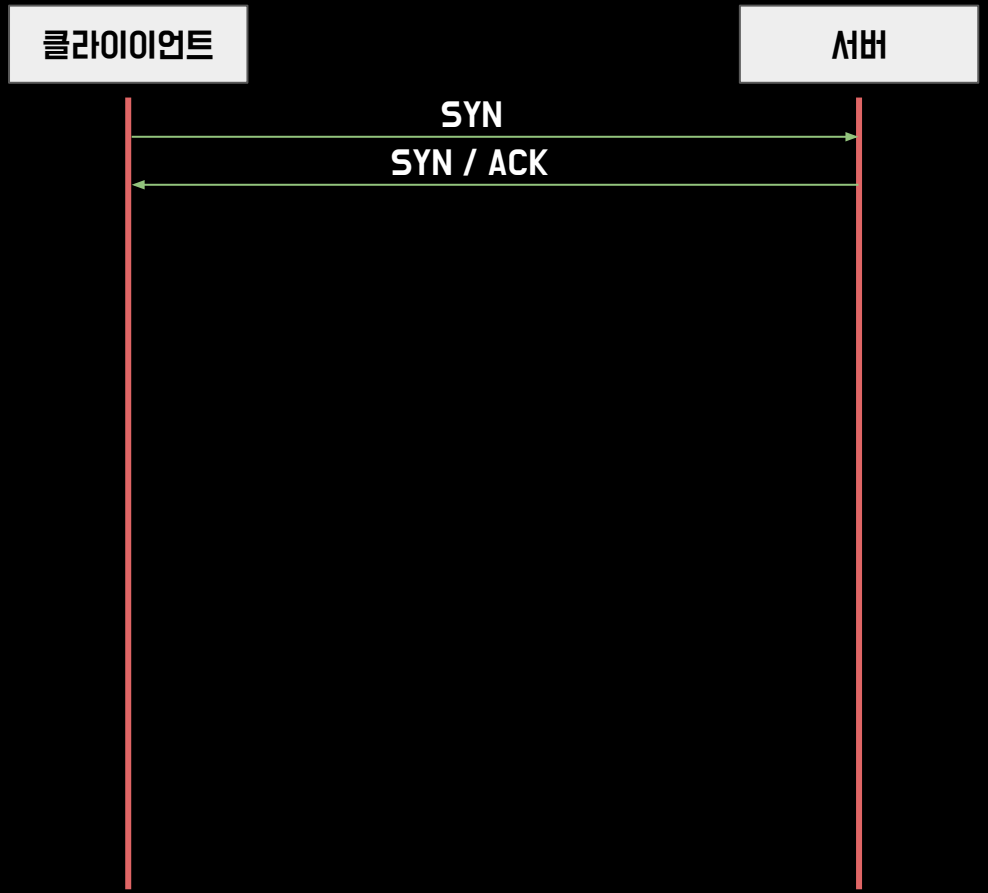
이벤트 기반
sub/pub으로 관점으로
바꿀수 있다고 생각.

웹소켓은 커넥션을 유지하므로 비용이 클까?

배경지식
tcp handshake


클라이이언트
서버


클라이언트
서버
SYN


클라이언트
서버
SYN
SYN / ACK


클라이언트
서버
SYN
SYN / ACK
ACK


클라이언트
서버
SYN
SYN / ACK
ACK
DATA


클라이언트
서버
SYN
SYN / ACK
ACK
DATA
ACK


클라이언트
서버
SYN
SYN / ACK
ACK
DATA
ACK
FIN


클라이언트
서버
SYN
SYN / ACK
ACK
DATA
ACK
FIN
ACK


클라이언트
서버
SYN
SYN / ACK
ACK
DATA
ACK
FIN
ACK
FIN


클라이언트
서버
SYN
SYN / ACK
ACK
DATA
ACK
FIN
ACK
FIN
ACK

커넥션을 유지하는 방식의 장점

나만 모르고 있던 - HTTP/2

심 천보
2016-12-04

소프트웨어 | http1.1, http2, spdy

자그마치 15년여의 시간을 웹 통신 프로토콜의 절대권좌의 자리에 올라 꿋꿋이 버텨오신 당신의 똥고집에 세삼 존경심 마져 듭니다.

하지만 이제 그 자리를 내려 놓으셔야 겠습니다. 드디어 우리에겐 당신의 불편함과 느림을 대체할 새로운 분이 찾아 오셨거든요~

HTTP(HyperText Transfer Protocol)/1.1

굿바이~ 그 동안 수고 많으셨습니다!

‘hello’ echo
http과 websocket

```
GET / HTTP/1.1
Host: www.http2demo.io
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://www.http2demo.io/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _ga=GA1.2.20991305777.1551581737; _gid=GA1.2.1138553140.1551581737
If-None-Match: W/"5aa91b6d-19b00"

hello

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Sun, 03 Mar 2019 03:14:33 GMT
ETag: W/"5aa91b6d-19b00"
Server: CDN77-Turbo
Transfer-Encoding: chunked
X-Age: 30549494
X-Cache: HIT

hello
```

```
GET /text HTTP/1.1
Host: localhost:8080
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: Hello
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
upgrade: websocket
connection: upgrade
sec-websocket-accept: y70AqGCGEjoPu6Ah1mY4X6i0inw=

hello
hello
```

GET / HTTP/1.1
Host: www.http2demo.io
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://www.http2demo.io/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _ga=GA1.2.209913057.1551581737; _gid=GA1.2.1138553140.1551581737
If-None-Match: W/"5aa91b6d-19b00"

hello

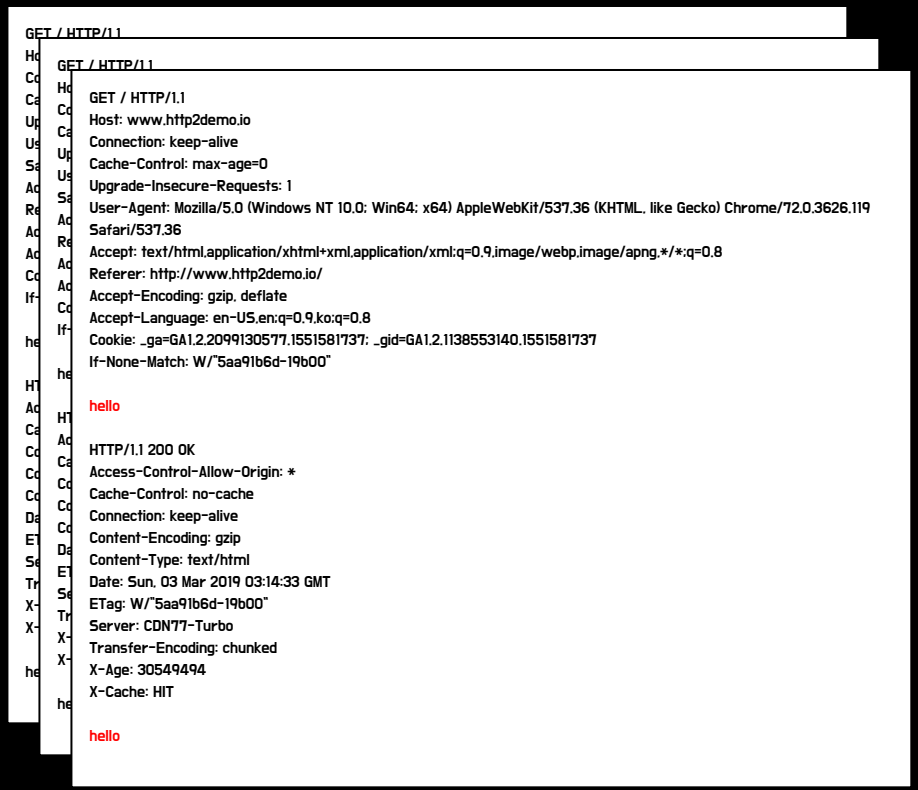
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Sun, 03 Mar 2019 03:14:33 GMT
ETag: W/"5aa91b6d-19b00"
Server: CDN77-Turbo
Transfer-Encoding: chunked
X-Age: 30549494
X-Cache: HIT

hello

GET /text HTTP/1.1
Host: localhost:8080
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: Hello
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
upgrade: websocket
connection: upgrade
sec-websocket-accept: y70AqGCGEjoPu6Ah1mY4X6i0inw=

hello
hello
hello
hello




```
GET /text HTTP/1.1
Host: localhost:8080
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: Hello
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
upgrade: websocket
connection: upgrade
sec-websocket-accept: y70AqGCGEjoPu6Ah1mY4X6i0inw=

hello
hello
hello
hello
hello
hello
```

GET / HTTP/1.1
Host: www.http2demo.io
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://www.http2demo.io/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _ga=GA1.2.2099130577.1551581737; _gid=GA1.2.1138553140.1551581737
If-None-Match: W/"5aa91b6d-19b00"

hello

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Sun, 03 Mar 2019 03:14:33 GMT
ETag: W/"5aa91b6d-19b00"
Server: CDN77-Turbo
Transfer-Encoding: chunked
X-Age: 30549494
X-Cache: HIT

hello

GET /text HTTP/1.1
Host: localhost:8080
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: Hello
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
upgrade: websocket
connection: upgrade
sec-websocket-accept: y70AqGCGEjoPu6Ah1mY4X6i0inw=

hello
hello
hello
hello
hello
hello
hello
hello

```
GET / HTTP/1.1
Host: www.http2demo.io
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://www.http2demo.io/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _ga=GA1.2.2099130577.1551581737; _gid=GA1.2.1138553140.1551581737
If-None-Match: W/"5aa91b6d-19b00"

hello

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Sun, 03 Mar 2019 03:14:33 GMT
ETag: W/"5aa91b6d-19b00"
Server: CDN77-Turbo
Transfer-Encoding: chunked
X-Age: 30549494
X-Cache: HIT

hello
```

hello 메세지를 받기 위해
매번 http 전문을 만들어서
반복적으로 서버에 전달해야함.

```
GET /text HTTP/1.1
Host: localhost:8080
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: Hello
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
upgrade: websocket
connection: upgrade
sec-websocket-accept: y70AqGCGEjoPu6Ah1mY4X6i0inw=

hello
hello
hello
hello
hello
hello
hello
hello
```

```
GET / HTTP/1.1
Host: www.http2demo.io
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://www.http2demo.io/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _ga=GA1.2.2099130577.1551581737; _gid=GA1.2.1138553140.1551581737
If-None-Match: W/"5aa91b6d-19b00"

hello

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Sun, 03 Mar 2019 03:14:33 GMT
ETag: W/"5aa91b6d-19b00"
Server: CDN77-Turbo
Transfer-Encoding: chunked
X-Age: 30549494
X-Cache: HIT

hello
```

```
GET /text HTTP/1.1
Host: localhost:8080
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: Hello
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
upgrade: websocket
connection: upgrade
sec-websocket-accept: y70AqGCGEjoPu6Ah1mY4X6i0inw=

hello
hello
hello
hello
hello
hello
hello
hello
```

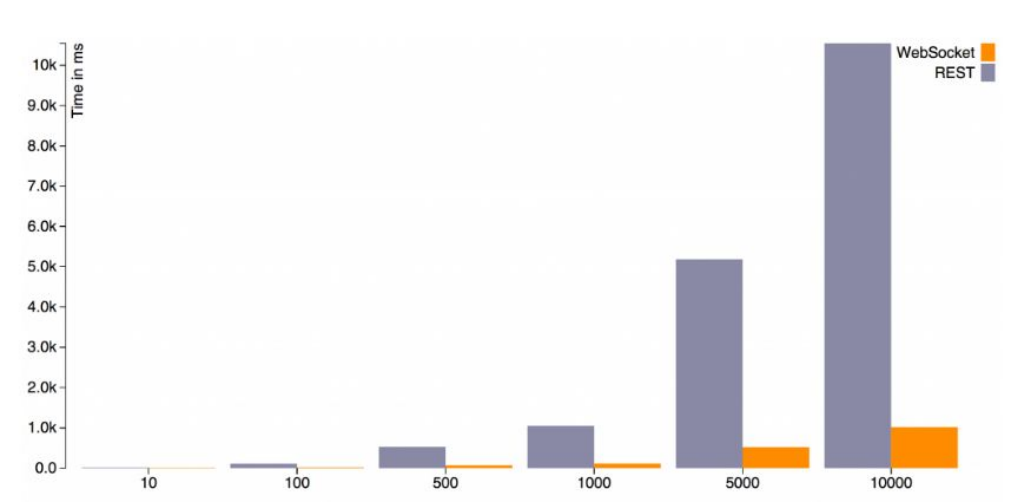
맺은 커넥션으로
보낼 데이터만 전달하면됨

```
GET / HTTP/1.1
Host: www.http2demo.io
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119
Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://www.http2demo.io/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ko;q=0.8
Cookie: _ga=GA1.2.2099130577.1551581737; _gid=GA1.2.1138553140.1551581737
If-None-Match: W/"5aa91b6d-19b00"

hello

HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Cache-Control: no-cache
Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html
Date: Sun, 03 Mar 2019 03:14:33 GMT
ETag: W/"5aa91b6d-19b00"
Server: CDN77-Turbo
Transfer-Encoding: chunked
X-Age: 30549494
X-Cache: HIT

hello
```

웹소켓 대비 http 방식은 네트워크 비용이 큼

```
GET /text HTTP/1.1
Host: localhost:8080
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: Hello
Sec-WebSocket-Version: 13

HTTP/1.1 101 Switching Protocols
upgrade: websocket
connection: upgrade
sec-websocket-accept: y70AqGCGEjoPu6Ah1mY4X6i0inw=

hello
hello
hello
hello
hello
hello
hello
hello
```

rest와 websocket 성능 비교 - Hello World echo 참고링크



Constant payload, increasing number of messages			
Messages	REST (in ms)	WebSocket (in ms)	x times
10	17	13	1.31
100	112	20	5.60
500	529	68	7.78
1000	1050	115	9.13
5000	5183	522	9.93
10000	10547	1019	10.35

Blocking IO

NIO(Non-blocking IO)

Web S...ks in Spring 5

Blocking

Servlet Stack

Spring MVC

Servlet API

Servl...tainer

Reactive Stack

Spring WebFlux

Reactive Streams

Netty, Servlet 3.1+, Undertow

Pivotal

Web S...ks in Spring 5

Blocking

Servlet Stack

~~Spring MVC~~

~~Servlet API~~

~~Servl...tainer~~

Reactive Stack

Spring WebFlux

Reactive Streams

Netty, Servlet 3.1+, Undertow

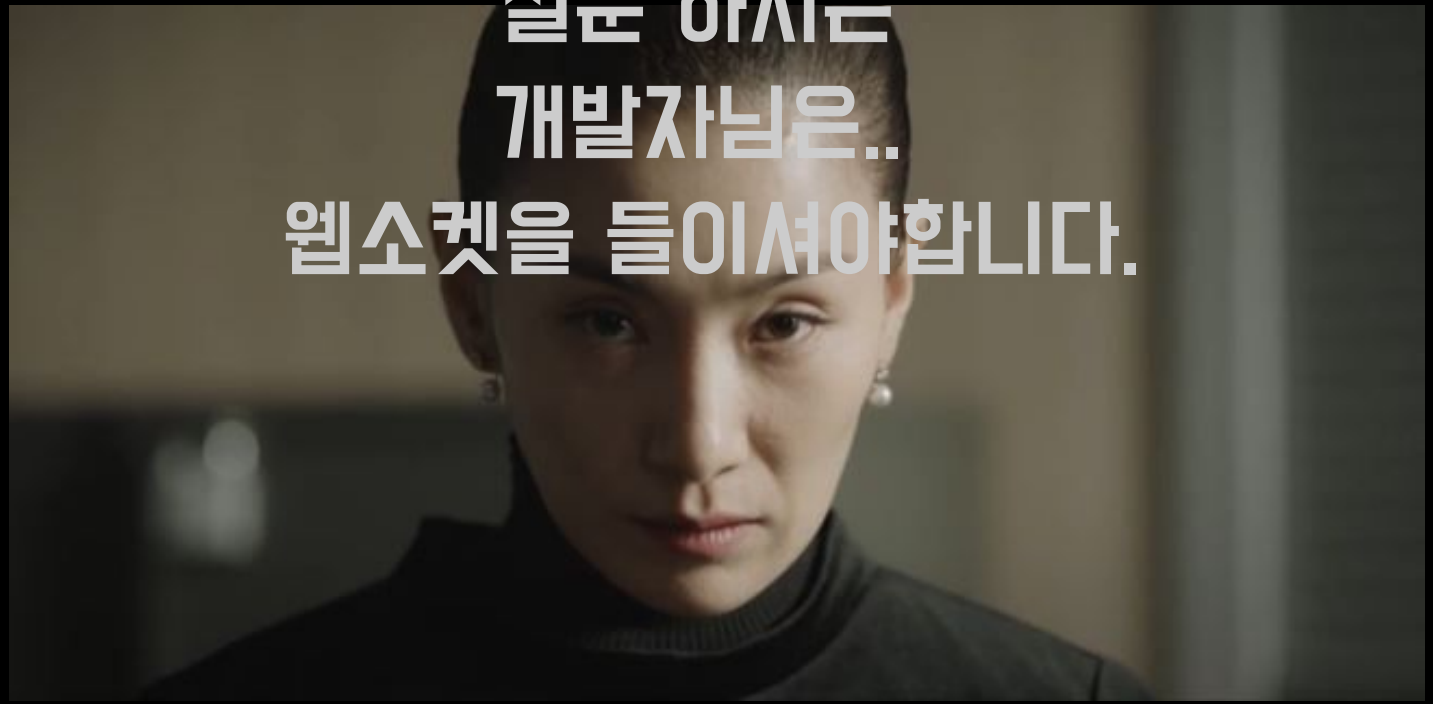

프로세스, 스레드,리액티브
부종민
슬라이드 URL - https://goo.gl/eNgMnS

마무리 정리..

기존의 것을 **바꾸자**라는 이야기가 아닙니다.
기술의 선택지를 **넓히자**는 이야기입니다.

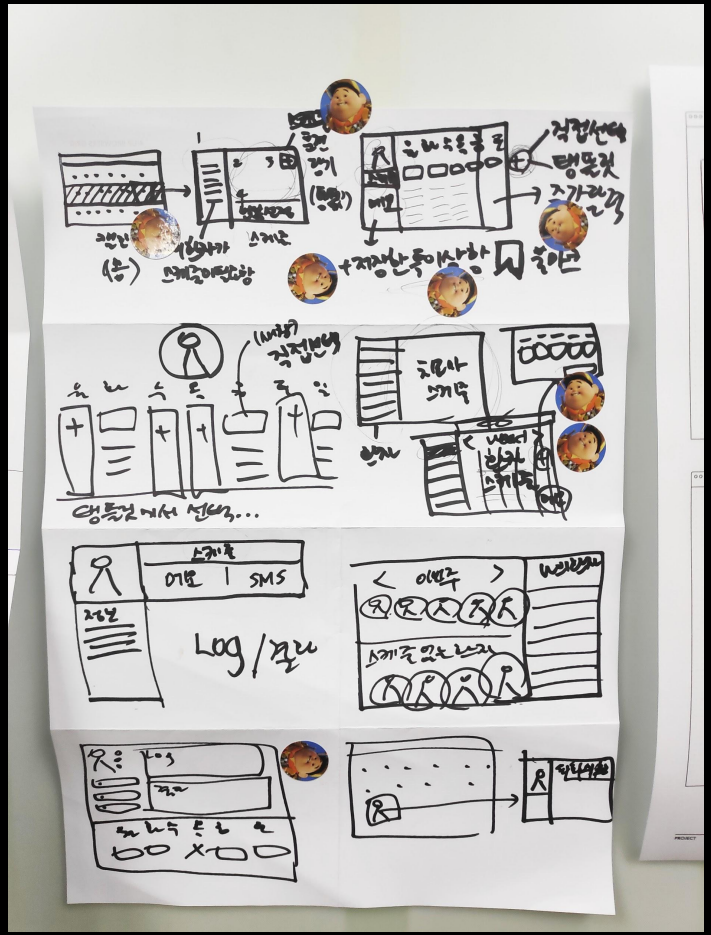
감사합니다.

Q&A

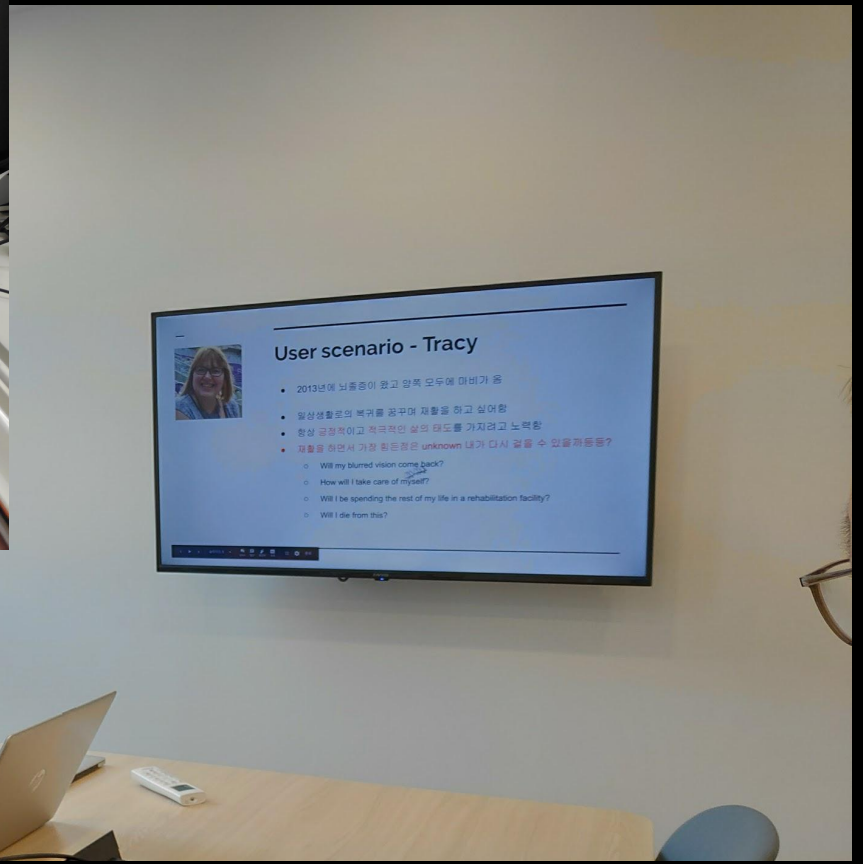


질문 하시는
개발자님은..
웹소켓을 들이셔야합니다.

NEOFECT


Log / 로그
SMS


게임 결과
플레이 시간 01:00 126
난이도 Rookie 1000
이동거리 190Km 760
최고속도 96Km/h 1050
최장거리 101Km 633
총점 3569
RAPAEL Clinic 테스트 시나리오
- 스마트 글러브 훈련하기
- 스마트 키즈 훈련하기


User scenario - Tracy
2013년에 뇌졸증이 왔고 양쪽 모두에 마비가 옴
일상생활로의 복귀를 꿈꾸며 재활을 하고 싶어함
항상 긍정적이고 적극적인 삶의 태도를 가지려고 노력함
재활을 하면서 가장 힘든점은 unknown 내가 다시 걸을 수 있을까등등?
Will my blurred vision come back?
How will I take care of myself?
Will I be spending the rest of my life in a rehabilitation facility?
Will I die from this?

혹시 이런 이런것에 관심있으신지요?

Java
Kotlin
Spring
Webflux
Kubernetes
Mongodb
Mysql
AWS
MSA
Devops

SPA
Vuejs
Android/IOS
Flutter
Agile스런 agile
Design sprint
Scrum
mvp

미국 대상으로하는 서비스
시장에 없는 서비스

야근 없음

애자일 신봉자 개발자
출신 상무(git 보면 밤에
코딩해서 push함)

좋은 동료

Who am I?

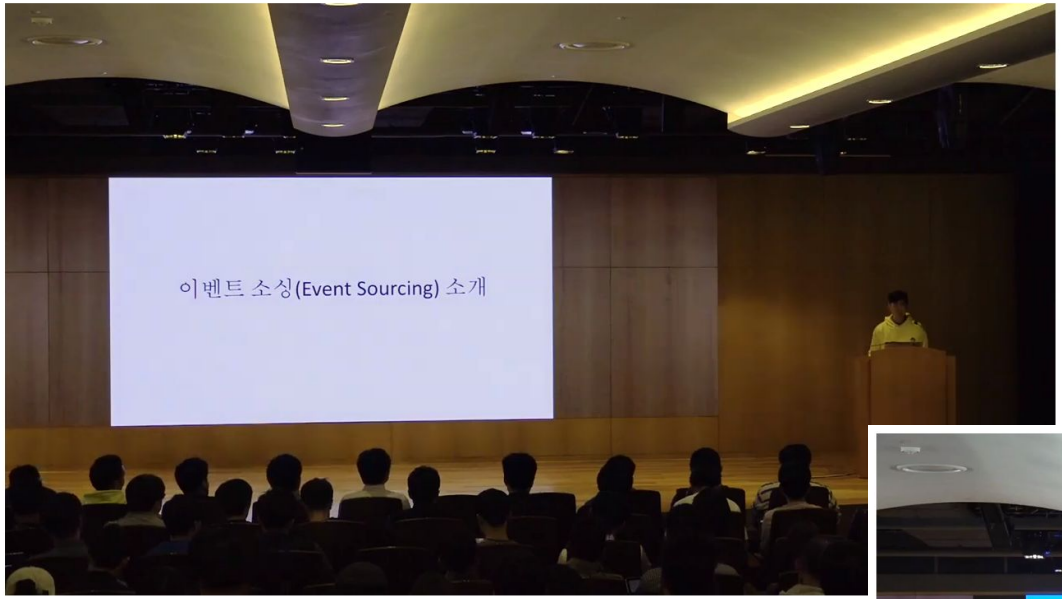
네오펙트 재직중.
자바. 스프링. jpa. 자바스크립트. vuejs. mysql

boojongmin@gmail.com
facebook.com/boojongmin1
github.com/boojongmin

transaction(BEGIN, COMMIT, ACK, ABORT), RECEIPT 있음.

데이터에 대해 중요한 비지니스 로직인 경우 트랜젝션을 사용하면 좋음.

특히 기존처럼 상태를 직접 변경하는 형태보다 EventSourcing을 이용하면 commit/rollback이 편리해짐.(ex: BINLOG)



스프링캠프 2017 [Day2 A2] : 이벤트 소싱 소개 (이론부)

스프링캠프 2017 [Day2 A3] : Implementing EventSourcing & CQRS (구현부)