실시간 알림 서비스

문자열 데이터 + JSON 데이터 전송

> index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Spring Boot SSE Example</title>
</head>
<body>
<h1>Server-Sent Events</h1>
<div id = "events"></div>

<script>
  const eventSource = new EventSource("/emitter");

  eventSource.onmessage = (event) => {
    const div = document.createElement("div");

    // 문자열 데이터 전송
```

```
    div.textContent = `Event received: ${event.data}`;

    // JSON 데이터 전송
    const eventData = JSON.parse(event.data);
    div.textContent = `Message: ${eventData.message}, TimeSta

    document.getElementById("events").appendChild(div);
  }

  eventSource.onerror = (error) => {
    console.error("error", error);
    eventSource.close();
  }
</script>
</body>
</html>
```

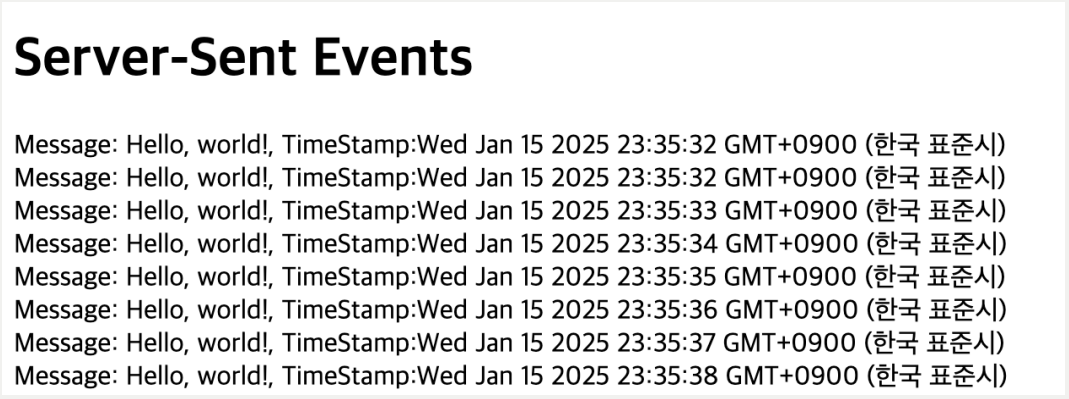
EmitterController.java

```
@RestController
@RequiredArgsConstructor
public class EmitterController {

    private final EmitterService emitterService;

    @GetMapping(path = "/emitter", produces = MediaType.TEXT_
    public SseEmitter sub() {
        SseEmitter emitter  = new SseEmitter();
        emitterService.addEmitter(emitter);
        emitterService.sendEvent();
        return emitter;
    }
}
```

EmitterService.java

```
public class EmitterService {

    private final List<SseEmitter> emitters = new CopyOnWrite
    private final ObjectMapper objectMapper;

    public void addEmitter(SseEmitter emitter) {
        // 새로운 SSE 연결
        emitters.add(emitter);
        // 완료 콜백, emitters 리스트에서 emitter 제거
        emitter.onCompletion(() -> emitters.remove(emitter));
        // 시간 초과 콜백, emitters 리스트에서 emitter 제거
        emitter.onTimeout(() -> emitters.remove(emitter));
    }

    @Scheduled(fixedRate = 1000)
    public void sendEvent() {
        for (SseEmitter emitter : emitters) {
            try {
                // 문자열 데이터 전송
                emitter.send("연결 완료.");

                // JSON 데이터 전송
                Map<String, Object> eventData = new HashMap<>
                eventData.put("message", "Hello, world!");
                eventData.put("timestamp", System.currentTime
                String json = objectMapper.writeValueAsString
                emitter.send(json, MediaType.APPLICATION_JSON

            } catch (IOException e) {
                emitter.complete();
                emitters.remove(emitter);
            }
        }
    }
}
```

💡 Scheduled 사용, 1초에 한 번씩 메세지 이벤트를 보내는 것을 확인 !



💡 **SSE를 사용하며 고려해야 할 점이 생겼음!**

팀장님 왈: 단일 WAS가 아닌 가용성을 위해 **멀티 WAS 환경**으로 구축된 환경을 사용할 수도 있습니다!

이러한 상황에서는 ***SSE만으로는 구현이 불가***

→ ***Redis Pub/Sub***을 활용하여 구현하는 방식으로 구현 해보려 한다!