MỤC LỤC

[MỤC LỤC 1](#_Toc16312)

[1](#_Toc25987)

[1.1. Socket io là gì? 2](#_Toc28361)

[1.2. Cơ chế hoạt động của socketio 2](#_Toc21489)

[1.3. Demo 2](#_Toc3773)

[2.1 SSE là gì? 5](#_Toc6142)

[2.2 SSE và Web sockets 5](#_Toc9723)

[2.3 Sử dụng SSE trên nestjs 6](#_Toc15034)

* 1. Socket io là gì?

Socket io là một module cho phép giao tiếp hai chiều giữa client và server. Socket io sử dụng tính năng giao tiếp hai chiều để phát triển các ứng dụng trò chuyện. Dữ liệu gửi về máy chủ sẽ được gửi trực tiếp cho client mà không cần yêu cầu trung gian.

* 1. Cơ chế hoạt động của socketio

Để lắng nghe data, ta sử dụng lệnh socket.on(), để gửi dữ liệu thì sử dụng lệnh socket.emit(). Ví dụ: client gửi đi 1 đoạn chat bằng socket.emit(), thì khi đó phía server cần socket.on() để nhận đoạn chat đó, sau đó truyền dữ liệu đoạn chat đó đi socket.emit(). Đồng thời ở tất cả client cũng cần lắng nghe đoạn chat đó bằng socket.on() từ server vừa mới gửi đi.

* Phía server

Thiết lập một connection điều khiển chức năng của socket và lắng nghe socket phía client.

* Phía client

Tạo socket ở phía người dùng thông qua thư viện của socket.

Tiếp đó gửi gói tin lên server thì sử dụng lệnh socket.emit(key,data). Tham số đầu tiên là một event. Nếu server phát hiện ra event phù hợp thì ngay lập tức thực hiện hàm, tham số thứ 2 là giá trị truyền vào event đó. Mục đích của việc gửi tin lên server là để server có thể lắng nghe và gửi tin lại về tất cả người dùng.

* Phía server

Lắng nghe sự kiện từ người dùng bằng lệnh socket.on(key,callback). Trong callback ta sẽ gửi dữ liệu cho phía người dùng thông qua lệnh socket.emit(key,data).

* 1. Demo

Các lệnh socket sử dụng trong demo:

* socket.emit(‘event’,data): gửi cho tất cả client.
* socket.to(‘room’).emit(): gửi cho tất cả client trong room.
* socket.on(‘event’,function): lắng nghe sự kiện từ người gửi rồi thực hiện gì đó.
* socket.join(‘room): kết nối tới một room.
* Phía Server

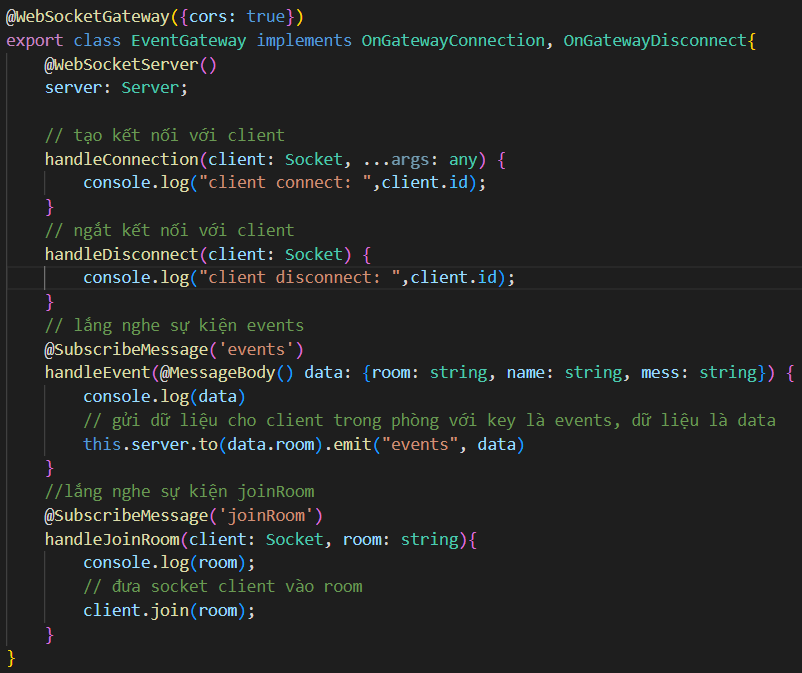
Bằng cách gắn decorator @WebsocketGateway() khiến lớp này có thể tương thích với thư viện Websocket. Ở đây ta sẽ dùng socket.io thay vì ws.

Sau đó implement các interface OnGatewayConnection, OnGatewayDisconnect. Hàm handleConnection của lớp OnGatewayConnection giúp server connect với client thay vì phải dùng lệnh on.(‘connection’) như trong nodejs, biết được client nào đang kết nối đến server. Tương tự lớp OnGatewayDisconnect cung cấp hàm handleDisconnect cho biết client mất kết nối.

Với decorator @SubcribeMessage sẽ lắng nghe sự kiện phía client gửi đến sau đó thực hiện hàm handle phía dưới: Gửi lại tất cả dữ liệu về phía client bằng lệnh emit(). Thay vì phải dùng lệnh on() để lắng nghe sự kiện từ client gửi đến như trong nodejs thì @SubcribeMessage giúp ta lắng nghe luôn, sau đó chỉ cần gửi emit() lại cho tất cả client.

Hàm handleEvent (Không nhất thiết phải tên này) có thể nhận các tham số như là:

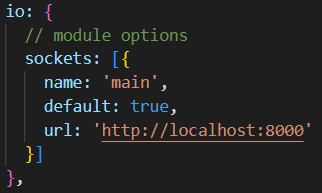
* @MessageBody: Lấy ra dữ liệu phía client gửi đến event.
* Socket: lấy ra được socket id của client. Ta có thể sử dụng socket này để tạo room cho người dùng.



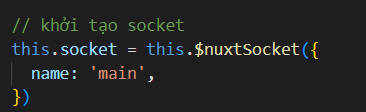
* Phía Client

Cài đặt thư viện nuxt-socket-io.

Thực hiện cấu hình socket trong nuxt.config.js: đặt name và cổng cho socket.



Khởi tạo socket theo name đã đặt trong config.



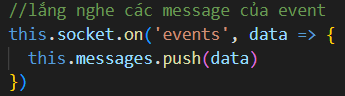
Để gửi một sự kiện lên server ta sử dụng lệnh socket.emit(key, data) trong đó key là tên sự kiện, data là dữ liệu muốn gửi đến server.



Để vào một phòng sử dụng lệnh emit gửi event joinroom lên server. Server lắng nghe và đưa client vào room.



Để lắng nghe sự liện mà server gửi tới, dùng lệnh socket.on(key, function) trong đó key là tên sự kiện, còn function đảm nhiệm in ra data mà server gửi tới hoặc dùng với mục đích khác.



2.1 SSE là gì?

Server sent event là công nghệ server push cho phép client nhận dữ liệu một cách tự động từ phía server thông qua một kết nối HTTP. SSE cho phép kết nối thời gian thực giữa client và server. Tuy nhiên chỉ có 1 chiều dữ liệu từ server đến client, client sẽ không thể gửi dữ liệu lên server.

SSE được sử dụng trong các ứng dụng bảng dashboard (trạng tái chuyến bay, chứng khoán), news feeds, kết quả thể thao, kết quả sổ xố.

2.2 SSE và Web sockets

Ưu điểm của SSE là dễ thực hiện, không yêu cầu việc cài đặt phức tạp như Web sockets.

Client chỉ cần tạo kết nối HTTP lên server một lần, và server giữ kết nối đó để liên tực gửi dữ liệu cho client. Client sẽ tự động kết nối lại với server nếu bị mất kết nối. Khi thực hiện kết nối lại, client sẽ tự động gửi ID của event cuối cùng nó nhận được, nhờ đó server có thể biết được client đã bỏ qua message nào để gửi lại những message đó.

2.3 Sử dụng SSE trên nestjs

Route của sse sẽ được đăng ký trong lớp controller. Với decorator @Sse sẽ kích hoạt SSE. Và sẽ trả về kiểu Observable phát ra nhiều MessageEvent mỗi giây. Đối tượng MessagEvent phải là các thuộc tính sau đây:

Data: string | object

Id: string

Type: string

Retry: number

*Lắng nghe sự kiện trên nuxtjs:*

Khởi tạo một EventSource, mỗi lần gửi sự kiện đều có dạng text/event-stream.

Để lắng nghe sự kiện từ server sử dụng onmessage.

* Server sẽ nhận sự kiện được client gửi lên thông qua socket. Thay vì sử dụng socket để lắng nghe phía server, thì sử dụng sse để lắng nghe phía server.
* Server sẽ cài đặt một route sse có decorator là @Sse và trả về cho client kiểu Observable<MessageEvent>.
* Khi client gửi lên thì sẽ lưu vào một biến để gửi qua route sse. Cứ mỗi 1s server sẽ trả về cho client một dữ liệu. Khi dữ liệu phía client gửi lên được cập nhật thì sse cũng cập nhật lại dữ liệu và gửi xuống cho client.

Client

Emit(event) gửi **data**

Server

Nhận **data** thông qua socket lắng nghe event->

Route sse gửi **data**