27/02/2024

Desarrollo De Aplicaciones Distribuidas

Investigación:

Ciclo De Vida De Socket IO

Integrantes:

Perea Valdez Leonardo Alberto

El ciclo de vida de Socket.io, una biblioteca de JavaScript para aplicaciones web en tiempo real, comienza con la creación de una conexión entre el cliente y el servidor a través de un socket. Esta conexión se mantiene abierta para permitir la comunicación bidireccional entre ambos. Durante esta comunicación, los eventos son enviados y recibidos, lo que permite la actualización en tiempo real de la información mostrada en la interfaz de usuario. Un ejemplo de esto es una aplicación de mensajería en la que los mensajes son enviados y recibidos instantáneamente.

- El cliente se conecta al servidor a través de Socket.IO. Esto activa el evento "connection" en el servidor.
- El servidor puede enviar datos al cliente en cualquier momento utilizando socket.emit(). Esto activa un evento correspondiente en el cliente.
- El cliente puede enviar datos al servidor utilizando socket.emit(). Esto activa un evento correspondiente en el servidor.
- Si la conexión se interrumpe debido a problemas de red, se activan los eventos "disconnect" en el servidor y "disconnect" en el cliente.
- Si la conexión se restablece, se activa nuevamente el evento "connection" en el servidor. El ID de sesión permanece intacto si la desconexión fue breve.
- Cuando el cliente se desconecta intencionalmente, se activa el evento "disconnect" en el servidor pero no en el cliente.
- En cualquier momento, el servidor puede emitir eventos a sockets individuales usando socket.to(id).emit() o a todos los sockets conectados usando io.emit().
- Los sockets también pueden unirse a "rooms" para permitir la comunicación grupal.

Ese es un resumen general del ciclo de vida básico de los sockets en Socket.IO, enfocándose en los principales eventos en el cliente y el servidor.

El ciclo de vida de Socket.IO se divide en dos partes: la conexión y la comunicación de eventos.

- Conexión: primero, el cliente se conecta al servidor a través de una URL específica. El servidor, a su vez, acepta la conexión y devuelve un objeto de socket al cliente. Este objeto de socket permite la comunicación entre el cliente y el servidor.
- Comunicación de eventos: una vez establecida la conexión, el cliente y el servidor pueden comunicarse a través de eventos. El cliente puede emitir eventos al servidor y el servidor puede hacer lo mismo. Cuando un evento es emitido, el objeto de socket se encarga de enviarlo al otro extremo de la conexión. El otro extremo puede estar escuchando por ese evento específico y, si lo está, ejecutará la función correspondiente.

Aquí hay un ejemplo guiado en español de cómo usar Socket.IO en un servidor y un cliente:

Ex. En Visual.

En este ejemplo, el servidor emite un evento de greetings cuando un cliente se conecta. El cliente, a su vez, está escuchando por este evento y lo imprime en la consola. El cliente también emite un evento de salutations al servidor cuando se conecta, pero en este ejemplo no estamos haciendo nada con ese evento en el servidor.

Espero que esto te ayude a entender el ciclo de vida de Socket.IO. ¡Buena suerte con tus proyectos!