WebSocketGateway

August 17, 2026

WebSocketGateway

Kind: Function

Source: `packages/websockets/decorators/socket-gateway.decorator.ts`

Part of: Websockets

`WebSocketGateway()` marks a class as a NestJS WebSocket gateway, allowing it to receive and emit real-time, event-based messages. Nest reads the decorator metadata during application startup, creates the configured WebSocket server adapter, and routes matching client events to gateway handlers.

Signature

```
function WebSocketGateway(portOrOptions: number | T, options: T): ClassDecorator
```

Parameters

Name	Type
`portOrOptions`	`number
`options`	`T`

Returns: `ClassDecorator`

Diagram

```
graph LR
  Client[Browser / WebSocket Client] --> Server[WebSocket Server Adapter]
  Server --> Gateway[@WebSocketGateway() Class]
  Gateway --> Handler[@SubscribeMessage() Handler]
  Handler --> Gateway
  Gateway --> Client
```

Usage

```
import {
  ConnectedSocket,
  MessageBody,
  SubscribeMessage,
  WebSocketGateway,
  WebSocketServer,
} from '@nestjs/websockets';
import { Server, Socket } from 'socket.io';

@WebSocketGateway({
  cors: {
    origin: 'http://localhost:3000',
  },
})
export class ChatGateway {
  @WebSocketServer()
  server: Server;

  @SubscribeMessage('chat:message')
  handleMessage(
    @MessageBody() message: { text: string },
    @ConnectedSocket() client: Socket,
  ) {
    this.server.emit('chat:message', {
      senderId: client.id,
      text: message.text,
    });
  }
}
```

AI Coding Instructions

- Apply `@WebSocketGateway()` only to provider classes registered in a Nest module.
- Use `@SubscribeMessage('event-name')` methods to handle incoming client events.
- Configure gateway options, such as `cors`, `namespace`, or `transports`, in the decorator when required by the client integration.
- Use `@WebSocketServer()` to emit messages to connected clients instead of creating a server instance manually.
- Validate incoming payloads with DTOs and pipes before broadcasting or processing client-provided data.