NestGateway

August 18, 2026

NestGateway

Kind: Interface

Source: `packages/websockets/interfaces/nest-gateway.interface.ts`

Part of: Websockets

`NestGateway` defines lifecycle hooks for WebSocket gateway classes in NestJS. Implement these optional methods to access the initialized server, react to client connections, and clean up when clients disconnect.

Properties

Property	Type
`afterInit`	`(server: any) => void`
`handleConnection`	`(...args: any[]) => void`
`handleDisconnect`	`(client: any) => void`

Diagram

```
graph LR
  A[WebSocket Server Initialized] --> B[afterInit(server)]
  B --> C[Client Connects]
  C --> D[handleConnection(...args)]
  D --> E[Client Disconnects]
  E --> F[handleDisconnect(client)]
```

Usage

```
import { WebSocketGateway } from '@nestjs/websockets';
import type { NestGateway } from '@nestjs/websockets';
```

```
@WebSocketGateway()
export class EventsGateway implements NestGateway {
  afterInit(server: any): void {
    console.log('WebSocket server initialized');
  }

  handleConnection(client: any, ...args: any[]): void {
    console.log(`Client connected: ${client.id}`);
  }

  handleDisconnect(client: any): void {
    console.log(`Client disconnected: ${client.id}`);
  }
}
```

AI Coding Instructions

- Implement `NestGateway` on classes decorated with `@WebSocketGateway()` when lifecycle callbacks are needed.
- Use `afterInit` to configure or inspect the underlying WebSocket server after it is created.
- Keep `handleConnection(...args)` flexible because adapter-specific connection arguments may follow the client object.
- Release client-specific resources, subscriptions, and session state in `handleDisconnect`.
- Avoid assuming a specific client or server API when supporting multiple WebSocket adapters; narrow `any` values to the adapter type in use.