AbstractWsAdapter

August 17, 2026

AbstractWsAdapter

Kind: Class

Source: `packages/websockets/adapters/ws-adapter.ts`

Part of: Websockets

`AbstractWsAdapter` is the base class for WebSocket transport adapters in NestJS. It provides shared lifecycle behavior for client connection/disconnection events and server cleanup, while concrete adapters implement server creation and message-handler binding for a specific WebSocket library.

Implements: `WebSocketAdapter`

Methods

Method	Signature	Returns
`bindClientConnect`	`bindClientConnect(server: TServer, callback: Function)`	`void`
`bindClientDisconnect`	`bindClientDisconnect(client: TClient, callback: Function)`	`void`
`close`	`close(server: TServer)`	`void`
`dispose`	`dispose()`	`void`
`create`	`create(port: number, options: TOptions)`	`TServer`
`bindMessageHandlers`	`bindMessageHandlers(client: TClient, handlers: WsMessageHandler[], transform: (data: any) => Observable<any>)`	`void`

Properties

Property	Type
`httpServer`	`any`

Diagram

```
graph LR
  App[Nest Application] --> Adapter[AbstractWsAdapter]
  Adapter --> Create[create()]
  Adapter --> Connect[bindClientConnect()]
  Adapter --> Disconnect[bindClientDisconnect()]
  Adapter --> Messages[bindMessageHandlers()]
  Adapter --> Close[close() / dispose()]
  Create --> Server[WebSocket Server]
  Server --> Client[WebSocket Client]
  Client --> Messages
```

Usage

```
import { AbstractWsAdapter } from '@nestjs/websockets';

class CustomWsAdapter extends AbstractWsAdapter<MyServer, MyClient> {
  create(port: number): MyServer {
    return createMyWebSocketServer(port);
  }

  bindMessageHandlers(client: MyClient, handlers, transform) {
    client.on('message', async rawMessage => {
      const message = JSON.parse(rawMessage.toString());

      for (const handler of handlers) {
        if (handler.message === message.event) {
          const response = await transform(
            handler.callback(message.data, client),
          );

          client.send(JSON.stringify(response));
        }
      }
    });
  }
}

// main.ts
async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  app.useWebSocketAdapter(new CustomWsAdapter(app));
```

```
  await app.listen(3000);
}
```

AI Coding Instructions

- Extend `AbstractWsAdapter` rather than instantiating it directly; implement `create()` and `bindMessageHandlers()` for the chosen WebSocket library.
- Use `bindClientConnect()` and `bindClientDisconnect()` to attach lifecycle callbacks instead of duplicating connection event wiring.
- Ensure incoming messages are parsed and routed to the matching Nest gateway handler in `bindMessageHandlers()`.
- Always support adapter cleanup through `close()` and `dispose()`, especially when the underlying server keeps open client connections.
- Register the concrete adapter with `app.useWebSocketAdapter()` during application bootstrap.

Used by

2 references from 2 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (2)

- `IoAdapter` — `packages/platform-socket.io/adapters/io-adapter.ts`:14
- `WsAdapter` — `packages/platform-ws/adapters/ws-adapter.ts`:39