Exception filters

The only difference between **HTTP exception filter** layer and corresponding web sockets layer is that instead of throwing `HttpException`, you should rather use `WsException`.

```
throw new WsException('Invalid credentials.');
```

HINT
The `WsException` class is imported from the `@nestjs/websockets` package.

Nest will handle thrown exception and as a result, emits the `exception` message with the following structure:

```
{
  status: 'error',
  message: 'Invalid credentials.'
}
```

Filters

The **custom filters** feature is supported as well and works equivalently. Here is an example that makes use of a manually instantiated method-scope filter (class-scoped works too):

```
@UseFilters(new WsExceptionFilter())
@SubscribeMessage('events')
onEvent(client, data: any): WsResponse<any> {
  const event = 'events';
  return { event, data };
}
```

Inheritance

Typically, you'll create fully customized exception filters crafted to fulfill your application requirements. There might be use-cases though when you would like to reuse an already implemented, **core exception filter**, and override the behavior based on certain factors.

In order to delegate exception processing to the base filter, you need to extend `BaseWsExceptionFilter` and call inherited `catch()` method.

```js
import { Catch, ArgumentsHost } from '@nestjs/common';
import { BaseWsExceptionFilter } from '@nestjs/websockets';

@Catch()
export class AllExceptionsFilter extends BaseWsExceptionFilter {
  catch(exception: unknown, host: ArgumentsHost) {
    super.catch(exception, host);
  }
}
```

Obviously, you should enhance above implementation with your tailored **business** logic (e.g. add various conditions).

Support us

Nest is an MIT-licensed open source project. It can grow thanks to the support by these awesome people. If you'd like to join them, please read more here.

Principal Sponsor

Sponsors / Partners