nest

Exception filters

The only difference between HTTP exception filter layer and corresponding microservices layer is that instead of throwing `HttpException`, you should rather use `RpcException`.

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

HINT
The `RpcException` class is imported from the `@nestjs/microservices` package.

Nest will handle thrown exception and as a result, returns the `error` object with the following structure:

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

Filters

The exception filters work in the same fashion as the primary ones, with a one, small difference. The `catch()` method has to return an `Observable`.

rpc-exception.filter.ts JS

```
import { Catch, RpcExceptionFilter, ArgumentsHost } from '@nestjs/common';
import { Observable, throwError } from 'rxjs';
import { RpcException } from '@nestjs/microservices';

@Catch(RpcException)
export class ExceptionFilter implements RpcExceptionFilter<RpcException> {
  catch(exception: RpcException, host: ArgumentsHost): Observable<any> {
    return throwError(exception.getError());
  }
}
```

WARNING
It is impossible to set up the microservice exception filters globally when the hybrid application feature is being used.

Here is an example that makes use of a manually instantiated method-scoped filter (class-scoped works too):

JS

```
@UseFilters(new ExceptionFilter())
@MessagePattern({ cmd: 'sum' })
accumulate(data: number[]): number {
  return (data || []).reduce((a, b) => a + b);
}
```

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 `BaseExceptionFilter` and call inherited `catch()` method. Besides, `HttpServer` reference has to be injected and passed to the `super()` call.

JS

```
import { Catch, ArgumentsHost } from '@nestjs/common';
import { BaseRpcExceptionFilter } from '@nestjs/microservices';

@Catch()
export class AllExceptionsFilter extends BaseRpcExceptionFilter {
  catch(exception: any, host: ArgumentsHost) {
    return 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

Become a sponsor