HTTP Exception & Mailer

Course Code: CSC 4182 Course Title: Advanced Programming In Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**1**	**Week No:**	**06**	**Semester:**	
Lecturer:	*Sazzad Hossain; sazzad@aiub.edu*				

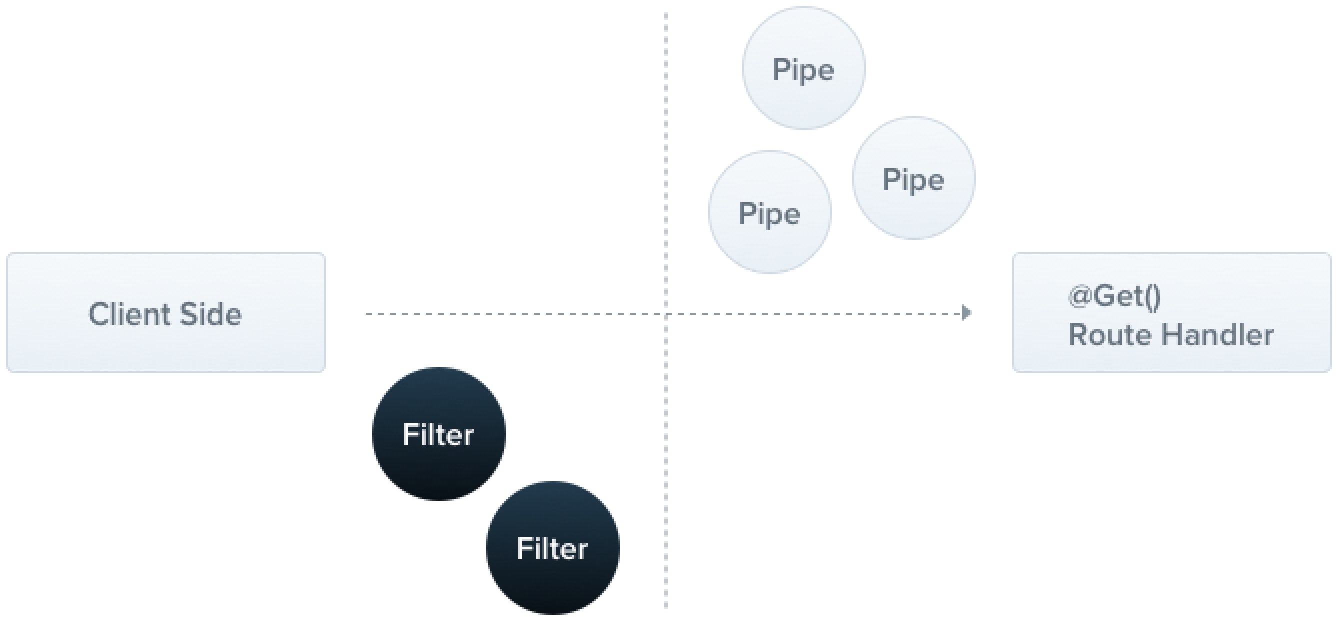
Lecture Outline

- ✓ Exception filters
- ✓ Mailer

Exception filters

- Nest comes with **a built-in exceptions layer** which is responsible for processing all unhandled exceptions across an application.
- exception is not handled by your application code, it is caught by this layer, which then automatically sends **an appropriate user-friendly response.**

Exception filters



Exception filters

Throwing standard exceptions

- Nest provides a built-in **HttpException** class, exposed from the @nestjs/common package. For typical HTTP REST API based applications, it's best practice to send standard HTTP response objects when certain error conditions occur.

```
throw new HttpException('Forbidden', HttpStatus.FORBIDDEN);
```

```
{ "statusCode": 403, "message": "Forbidden" }
```

Exception filters

The **HttpException constructor** takes two required arguments which determine the response:

- The **status** argument defines the HTTP status code.

By default, the JSON response body contains two properties:

- **statusCode**: defaults to the HTTP status code provided in the status argument
- **message**: a short description of the HTTP error based on the status

Exception filters

```
@Get()
async findAll() {
  try {
    await this.service.findAll()
  } catch (error) {
    throw new HttpException({
      status: HttpStatus.FORBIDDEN,
      error: 'This is a custom message',
    });
  }
}
```

Exception filters

BadRequestException: This exception represents a **400** Bad Request HTTP status. It is typically used when the client sends a malformed or invalid request.

UnauthorizedException: This exception represents a **401** Unauthorized HTTP status. It is used when the client is not authenticated or does not have the necessary credentials to access a resource.

NotFoundException: This exception represents a **404** Not Found HTTP status. It is used when the requested resource is not found on the server.

ForbiddenException: This exception represents a **403** Forbidden HTTP status. It is used when the client is authenticated but does not have sufficient permissions to access a resource.

Exception filters

NotAcceptableException: This exception represents a **406** Not Acceptable HTTP status. It is used when the server cannot produce a response that is acceptable according to the client's requested content types.
RequestTimeoutException: This exception represents a **408** Request Timeout HTTP status. It is used when the server times out waiting for the client to send a request.
ConflictException: This exception represents a **409** Conflict HTTP status. It is typically used in situations where there is a conflict with the current state of the resource, such as a concurrent update.
GoneException: This exception represents a **410** Gone HTTP status. It is used when the requested resource is no longer available and has been intentionally removed.

Exception filters

PayloadTooLargeException: This exception represents a **413** Payload Too Large HTTP status. It is used when the server refuses to process a request because the payload size exceeds the server's limit.
UnsupportedMediaTypeException: This exception represents a **415** Unsupported Media Type HTTP status. It is used when the server does not support the media type specified in the request.
UnprocessableEntityException: This exception represents a **422** Unprocessable Entity HTTP status. It is typically used when the server understands the request but cannot process it due to semantic errors or validation failures.

More Exception can be found here
https://docs.nestjs.com/exception-filters#built-in-http-exceptions

Exception filters

PayloadTooLargeException: This exception represents a **413** Payload Too Large HTTP status. It is used when the server refuses to process a request because the payload size exceeds the server's limit.
UnsupportedMediaTypeException: This exception represents a **415** Unsupported Media Type HTTP status. It is used when the server does not support the media type specified in the request.
UnprocessableEntityException: This exception represents a **422** Unprocessable Entity HTTP status. It is typically used when the server understands the request but cannot process it due to semantic errors or validation failures.

More Exception can be found here
https://docs.nestjs.com/exception-filters#built-in-http-exceptions

Mailer

- **Nodemailer** is a popular email sending library for Node.js applications. It allows you to send emails using various email providers, such as **SMTP, SendGrid, Mailgun, and more**.
- Nodemailer provides a simple and straightforward API to compose and send email messages with features like attachments, HTML content, and sending emails using templates.
- It supports various email protocols and transports, making it versatile and suitable for different email sending requirements.

Mailer

First install Node Mailer package

```
npm install --save @nestjs-modules/mailer nodemailer
```

add following configuration on specific user module

```
import { MailerModule } from "@nestjs-modules/mailer";
MailerModule.forRoot({
transport: {
host: 'smtp.gmail.com',
port: 465,
ignoreTLS: true,
secure: true,
auth: {
user: 'your gmail account',
pass: 'app generated password'
},
}})
```

Mailer

Use Mailer Service on user specific **service** file;
Create a Mailer Service object

```
//import MailerService from nodemail modules
import { MailerService } from "@nestjs-modules/mailer/dist";

//create object of MailerService class
private mailerService: MailerService

//use sendMail to send email
await this.mailerService.sendMail({to: 'receiver’s email',
subject: 'subject of email',
text: 'text to receiver',
});
```

Mailer

Use following instruction to configure your Gmail account;

https://miracleio.me/snippets/use-gmail-with-nodemailer/

References

1. W3Schools Online Web Tutorials, URL: http://www.w3schools.com
2. Node.js, URL: https://nodejs.org/en/
3. Next.js, URL: https://nextjs.org/
4. TypeScript URL: https://www.typescriptlang.org/
5. MDN Web Docs URL: https://developer.mozilla.org/

Thank You!