NestJS Interview Questions with Answers

1. Basic Questions

Q: NestJS là gì? Ưu điểm chính của NestJS là gì?

A: NestJS là framework dùng TypeScript để phát triển server-side application theo mô hình OOP, DI và module hóa. Ưu điểm: code rõ ràng, dễ test, hỗ trợ REST & GraphQL, tích hợp WebSocket, microservices.

Q: So sánh NestJS và ExpressJS.

A: NestJS là lớp trừu tượng cao hơn, cung cấp cấu trúc module hóa rõ ràng. ExpressJS đơn giản hơn, linh hoạt nhưng dễ rối khi project lớn.

Q: Module trong NestJS là gì?

A: Module là đơn vị tổ chức logic trong NestJS, gom nhóm các controller, service liên quan và hỗ trợ DI.

Q: Controller và Service trong NestJS có vai trò gì?

A: Controller nhận request và gọi service xử lý. Service chứa logic nghiệp vụ chính.

Q: Decorator là gì? Kể tên một số decorator thường dùng trong NestJS.

A: Là hàm đặc biệt định nghĩa metadata cho class/method. Ví dụ: @Controller, @Injectable, @Get, @Post.

Q: Injectable() decorator có tác dụng gì?

A: Giúp Nest biết class này là provider và có thể inject vào nơi khác thông qua DI.

NestJS Interview Questions with Answers

Q: Cơ chế Dependency Injection trong NestJS hoạt động như thế nào?

A: NestJS dùng container nội bộ để quản lý và cung cấp dependencies tự động vào constructor khi cần.

Q: @Get(), @Post(), @Param(), @Body(), @Query() dùng để làm gì?

A: Là các decorator để ánh xạ HTTP method, lấy dữ liệu từ route, query, body của request.

Q: Cấu trúc dự án NestJS chuẩn gồm những gì?

A: Gồm main.ts, app.module.ts, các module, controller, service, dto, pipe, guard, filter theo module.

Q: Làm sao để tạo một RESTful API đơn giản với NestJS?

A: Tạo controller với route, tạo service xử lý logic, dùng decorator @Get/@Post cho endpoint.

2. Intermediate Questions

Q: Lifecycle Hooks trong NestJS là gì?

A: Là các hàm như onModuleInit(), onModuleDestroy() dùng để chạy logic khi module khởi tạo hoặc hủy.

Q: Middleware là gì? Làm sao để áp dụng middleware vào route?

A: Middleware xử lý request trước controller. Áp dụng bằng consumer.apply(Middleware).forRoutes(...).

Q: Guard là gì? So sánh Guard với Middleware.

NestJS Interview Questions with Answers

A: Guard kiểm soát quyền truy cập route. Middleware không kiểm soát quyền, chỉ xử lý request thô.

Q: Pipe là gì? Dùng để làm gì trong NestJS?

A: Pipe biến đổi và validate dữ liệu trước khi vào controller. Dùng cho input validation.

Q: Interceptor là gì? Khi nào nên dùng Interceptor?

A: Interceptor dùng để modify response, log, transform, hoặc thêm logic trước/sau handler.

Q: Cách xử lý exception trong NestJS như thế nào?

A: Sử dụng built-in hoặc custom ExceptionFilter để bắt và xử lý lỗi theo chuẩn RESTful.

Q: Custom Exception Filter là gì? Viết ví dụ.

A: Là class implements ExceptionFilter để xử lý lỗi tùy biến, ví dụ trả JSON lỗi rõ ràng.

Q: Validation trong NestJS thường dùng thư viện nào?

A: Thường dùng class-validator và class-transformer kèm DTO.

Q: Cách kết nối với cơ sở dữ liệu trong NestJS (TypeORM, Prisma)?

A: Import module như TypeOrmModule/PrismaModule, cấu hình connection và inject repository/service.

Q: DTO là gì? Có tác dụng gì trong dự án NestJS?

A: DTO (Data Transfer Object) định nghĩa cấu trúc dữ liệu vào/ra, giúp dễ validate và clean

code.

3. Authentication and Security

Q: Cách tích hợp JWT Authentication trong NestJS?

A: Dùng @nestjs/jwt, tạo AuthService để sign & verify token, kết hợp với Guard để bảo vệ route.

Q: Làm sao để bảo vệ một route chỉ cho phép người dùng đã đăng nhập?

A: Tạo JwtAuthGuard extends AuthGuard('jwt'), áp dụng bằng @UseGuards(JwtAuthGuard).

Q: Làm sao để refresh token trong NestJS?

A: Lưu refresh token riêng, cấp lại access token khi hết hạn thông qua endpoint /refresh.

Q: Cách tích hợp OAuth2, Google Login, Facebook Login với NestJS?

A: Dùng passport và các strategy như passport-google-oauth20 để login bằng mạng xã hội.

Q: CSRF và CORS trong NestJS được xử lý như thế nào?

A: Dùng middleware hoặc trong main.ts để cấu hình CORS. CSRF xử lý thủ công hoặc dùng helmet.

4. Advanced Questions

Q: Viết một Custom Decorator trong NestJS.

A: Sử dụng createParamDecorator hoặc applyDecorators để tạo decorator tùy chỉnh cho param hoặc route.

NestJS Interview Questions with Answers

Q: Dynamic Module là gì? Khi nào nên dùng?

A: Module có thể nhận tham số cấu hình. Dùng khi module cần custom cấu hình khi import (ví dụ: DB, Email).

Q: Làm sao để test controller/service trong NestJS (unit test)?

A: Dùng Jest, Test.createTestingModule() để inject mock dependencies và test function.

Q: Tích hợp WebSocket với NestJS như thế nào?

A: Dùng @WebSocketGateway() và @SubscribeMessage để tạo socket server và xử lý event.

Q: Làm sao để tạo Cron job trong NestJS?

A: Dùng @nestjs/schedule với decorator @Cron hoặc @Interval để chạy task định kỳ.

Q: Microservice trong NestJS hoạt động ra sao?

A: Nest hỗ trợ TCP, Redis, Kafka... Giao tiếp qua MessagePattern, ClientProxy.

Q: NestJS hỗ trợ message broker nào? (e.g. Redis, NATS, Kafka)

A: Hỗ trợ Redis, NATS, Kafka, RabbitMQ, MQTT... thông qua microservice module.

Q: Cách tạo task background processing với Bull Queue trong NestJS?

A: Dùng @nestjs/bull, tạo Queue, Processor, thêm job và xử lý không đồng bộ.

Q: Làm sao để tích hợp GraphQL với NestJS?

A: Dùng @nestjs/graphql, tạo resolver thay vì controller, định nghĩa schema bằng code-first hoặc schema-first.

Q: Khi nào nên dùng Request-scoped provider thay vì Singleton?

A: Khi provider phụ thuộc vào dữ liệu request như user hiện tại, token, v.v.

5. Real-World Scenarios

Q: Bạn sẽ xử lý như thế nào nếu một route mất nhiều thời gian xử lý và gây nghẽn server?

A: Dùng queue để xử lý bất đồng bộ, cache kết quả, hoặc tách route sang microservice.

Q: Làm sao để tách cấu trúc module theo domain để dự án lớn vẫn dễ maintain?

A: Áp dụng DDD: chia theo domain, mỗi domain là module độc lập với controller, service, dto riêng.

Q: Cách triển khai rate limiting trong NestJS?

A: Dùng middleware hoặc thư viện như express-rate-limit, hoặc tích hợp Redis để giới hạn IP.

Q: Bạn đã bao giờ gặp lỗi circular dependency chưa? Làm sao để xử lý?

A: Dùng forwardRef() khi inject lẫn nhau giữa 2 service/module.

Q: Làm thế nào để cấu hình multi-database với TypeORM trong NestJS?

A: Dùng TypeOrmModule.forRoot() nhiều lần với name khác nhau, inject theo connection name.