

7.1.1

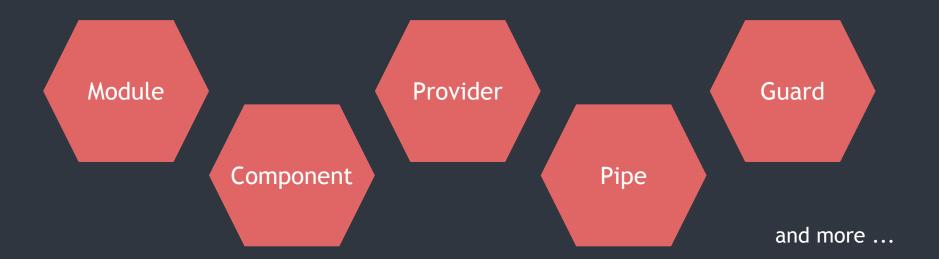
Wilson Su

Introduction

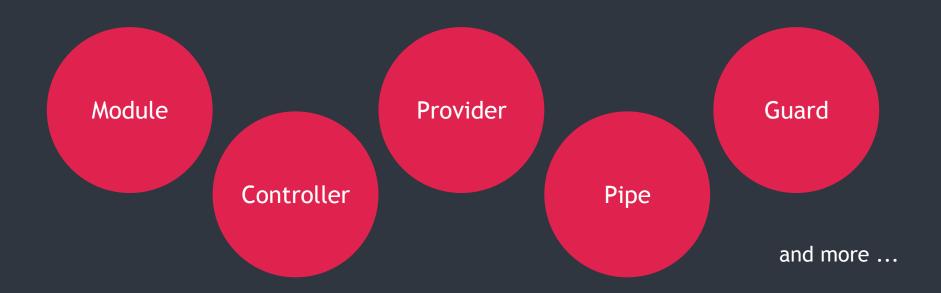
NestJS

- is a framework for building Node.js server-side applications
- is built with and fully supports TypeScript
- allows developers to create testable, scalable, loosely coupled, and easily maintainable applications
- the architecture is heavily inspired by Angular

Angular



NetsJS



Create a New Project

\$ npm i -g @nestjs/cli

```
$ nest new my-nest-app
```

my-nest-app

- ∟ src
 - □ app.controller.spec.ts
 - □ app.controller.ts
 - □ app.module.ts
 - □ app.service.ts
- ∟ test
 - □ app.e2e-spec.ts
 - ∟ jest-e2e.json
- □ nest-cli.json
- └ package.json
- ⊤ tsconfig.build.json
- ⊤ tsconfig.json

/src/app.service.ts

```
import { Injectable } from '@nestjs/common';
@Injectable()
export class AppService {
  getHello(): string {
   return 'Hello World!';
```

/src/app.controller.ts

```
import { Controller, Get } from '@nestjs/common';
import { AppService } from './app.service';
@Controller()
export class AppController {
  constructor(private readonly appService: AppService) { }
 @Get()
  getHello(): string {
    return this.appService.getHello();
```

/src/app.module.ts

```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
@Module({
  imports: [],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule { }
```

/src/main.ts

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  await app.listen(3000);
bootstrap();
```

<u>Nest CLI</u>

```
# Compiles and runs an application in watch mode
$ nest start --watch
# Compiles an application or workspace into an output folder.
$ nest build
# Generates files based on a schematic
$ nest generate module cats
$ nest generate controller cats
$ nest generate service cats
```

More Examples



Middleware

/src/middleware/logger.middleware

```
import { Injectable, NestMiddleware } from '@nestjs/common';
import { Request, Response } from 'express';
@Injectable()
export class LoggerMiddleware implements NestMiddleware {
  use(req: Request, res: Response, next: () => void) {
    const now = new Date().toISOString();
    const { method, originalUrl } = req;
    console.log(`${now} ${method}_${originalUrl}`);
   next();
```

/src/app.module.ts

```
import { Module, MiddlewareConsumer } from '@nestjs/common';
import { CatsModule } from './cats/cats.module';
import { CatsController } from './cats/cats.controller';
import { LoggerMiddleware } from './middleware/logger.middleware';
@Module({ ... })
export class AppModule {
  configure(consumer: MiddlewareConsumer) {
    consumer.apply(LoggerMiddleware)
      .forRoutes(CatsController);
```



Routing

/src/cats/cats.controller

```
@Controller('cats')
export class CatsController {
 @Get() // GET /cats
 fetchAll(): Cats { ... }
 @Post() // POST /cats
  create(): Promise<Cat> { ... }
  @Delete(':id') // DELETE /cats/1
  removeOne() { ... }
```



Pipe

/src/cats/cats.controller

```
import { ParseIntPipe, ... } from '@nestjs/common';
import { Response } from 'express';
import { cats } from './cats.mock';
@Controller('cats')
export class CatsController {
 @Get(':id')
 findOne(@Param('id', ParseIntPipe) id: number, @Res() res: Response) {
    const target = cats.find(cat => cat.id === id);
    res.status(HttpStatus.OK).json(target);
```

A Bad API Call

```
GET /cats/abc

/* Nest will throw an exception */
{
    "statusCode": 400,
    "message": "Validation failed (numeric string is expected)",
    "error": "Bad Request"
}
```