NestJS Crud with DB and Validator

Nest BUILD Backend

"@nestjs/class-validator": "^0.13.4",

"@nestjs/common": "^10.0.0",

"@nestjs/core": "^10.0.0",

"@nestjs/platform-express": "^10.0.0",

"@nestjs/typeorm": "^10.0.1",

"class-transformer": "^0.5.1",

"class-validator": "^0.14.1",

"mysql2": "^3.9.1",

"nodemon": "^3.0.3",

———————————

<https://docs.nestjs.com>

1. Install NestJS.>. npm i -g @nestjs/cli
2. Create Nest JS Project
   * ashujauhari@Ashus-MacBook-Air NestJS % **nest new nest-product-service**
   * **sudo npm install --save @nestjs/common**

* A screenshot of a computer

  Description automatically generated

1. Install “Material Icon Theme” in VSCode
2. nest-product-service % **npm install g --save nodemon**
3. Explain the bootstrap process.

//————————————————————Setup main controller

1. Add the request mapping at controller level

**@Controller(‘shop’)**

export class AppController {

constructor(private readonly appService: AppService) {}

**@Get('/hello')**

getHello(): string {

return this.appService.getHello();

}

**@Get('/name')**

getHelloName(): { name: string } {

//any

return { name: 'Cognizant' };

}

}

Run the application : > nest Start

Test : <http://localhost:3000/shop/hello>

<http://localhost:3000/shop/name>

nest g controller [name]

//————————————————————Setup Product controller

1. **Create Product controller, ProductService**

a. ***Create src/products/product.service.ts***

import { Injectable } from '@nestjs/common';

@Injectable()

export class ProductService {

**getHello(): string {**

return 'Hello Product from Product Service !';

}

}

***b. Create src/products/product.controller.ts and inject the ProductService in constructor***

import { Controller, Get } from '@nestjs/common';

import { ProductService } from './product.service';

@Controller('products')

export class PorductController {

constructor(private readonly productService: ProductService) {}

@Get('/hello')

getHello(): string {

**return this.productService.getHello();**

}

}

***c. Create src/products/product.module.ts***

import { Module } from '@nestjs/common';

import { ProductService } from './product.service';

import { PorductController } from './product.controller';

@Module({

imports: [],

controllers: [PorductController],

providers: [ProductService],

})

export class ProductModule {}

//——————————Import ProductModule in app.module.ts\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

a. import { Module } from '@nestjs/common';

import { AppController } from './app.controller';

import { AppService } from './app.service';

import { ProductModule } from './products/product.module';

@Module({

**imports: [ProductModule],**

controllers: [AppController],

providers: [AppService],

})

export class AppModule {}

Test it : <http://localhost:3000/products/hello>

//——————Create a Product

a. Create product.model.ts

export class Product {

constructor(

public id: number,

public pname: string,

public price: number,

) {}

}

b. Create product.service.ts

@Injectable()

export class ProductService {

products: Product[] = [];

insertProduct(id: number, pname: string, price: number): number {

const newProduct = new Product(id, pname, price);

this.products.push(newProduct);

console.log(newProduct.id + newProduct.pname + newProduct.price);

return id;

}

}

C. Create product.controller.ts

import { Body, Controller, Get, Post } from '@nestjs/common';

import { ProductService } from './product.service';

@Controller('products')

export class PorductController {

constructor(private readonly productService: ProductService) {}

@Post()

addProduct(

@Body('id') id: number,

@Body('pname') pname: string,

@Body('price') price: number,

) {

return { id: this.productService.insertProduct(id, pname, price) };

}

}

Test and verify

A screenshot of a web page

Description automatically generated

**1. Install and setup**

@nestjs/typeorm": "^10.0.1",

"class-transformer": "^0.5.1",

"class-validator": "^0.14.1",

"mysql2": "^3.9.1",

"nodemon": "^3.0.3",

npm install class-validator --save  
npm install class-transformer –save

1. **Set up DB configuration in app.module.ts**

import { Module } from '@nestjs/common';

import { AppController } from './app.controller';

import { AppService } from './app.service';

import { ProductModule } from './products/product.module';

import { TypeOrmModule } from '@nestjs/typeorm';

@Module({

imports: [

ProductModule,

TypeOrmModule.forRoot({

type: 'mysql',

host: 'localhost',

port: 3306,

username: 'root',

password: 'rootroot',

database: 'testdb',

autoLoadEntities: true,

synchronize: true,

entities: [],

}),

],

controllers: [AppController],

providers: [AppService],

})

export class AppModule {}

1. **Update Product.module.ts**

import { Module } from '@nestjs/common';

import { ProductService } from './product.service';

import { ProductController } from './product.controller';

import { TypeOrmModule } from '@nestjs/typeorm';

import { Product } from './product.model';

@Module({

imports: [TypeOrmModule.forFeature([Product])],

controllers: [ProductController],

providers: [ProductService],

exports: [ProductService, TypeOrmModule],

})

export class ProductModule {}

1. **Create ProductDTO .js**

import { IsNotEmpty } from 'class-validator';

export class ProductDto {

@IsNotEmpty()

public id: string;

@IsNotEmpty()

public pname: string;

@IsNotEmpty()

public price: number;

}

1. **Create ProductEntity**

import { Column, Entity, PrimaryGeneratedColumn } from 'typeorm';

@Entity('Product4')

export class Product {

@PrimaryGeneratedColumn('uuid')

id: string;

@Column()

pname: string;

@Column()

price: number;

}

1. **Create Product repository**

import { EntityRepository, Repository } from 'typeorm';

import { Product } from './product.model';

@EntityRepository(Product)

export class ProductRepository extends Repository<Product> {}

1. **Create ProductService.ts**

import { Injectable } from '@nestjs/common';

import { Product } from './product.model';

import { ProductRepository } from './product.respository';

import { InjectRepository } from '@nestjs/typeorm';

import { ProductDto } from './product-dto';

@Injectable()

export class ProductService {

constructor(

@InjectRepository(Product)

private productRepository: ProductRepository,

) {}

//Create a new Product

async insertProduct(productDto: ProductDto): Promise<Product> {

const { pname, price } = productDto;

const product = this.productRepository.create({ pname, price });

await this.productRepository.save(product);

return product;

}}

1. **Create ProductController.ts**

import { Body, Controller, Get, Post } from '@nestjs/common';

import { ProductService } from './product.service';

import { ProductDto } from './product-dto';

@Controller('products')

export class ProductController {

constructor(private readonly productService: ProductService) {}

@Get('/hello')

getHello(): string {

return this.productService.getHello();

}

@Post()

addProduct(@Body() productDto: ProductDto): Promise<ProductDto> {

return this.productService.insertProduct(productDto);

}

}

Test: <http://localhost:3000/products>

A screenshot of a web page

Description automatically generated

//————————————————————————Get All Products and GetByProductId

1. **Create ProductController.ts**

@Get()

getAllProducts(): Promise<Product[]> {

return this.productService.getAllProducts();

}

@Get('/:id')

getTaskById(@Param('id') id: string): Promise<Product> {

return this.productService.getProductById(id);

}

**10.Create ProductService.ts**

//Get all Products

async getAllProducts(): Promise<Product[]> {

const found = await this.productRepository.find();

if (!found) throw new NotFoundException('None of task assigned');

return found;

}

//------Get Product BY ID

async getProductById(id: string): Promise<Product> {

const found = await this.productRepository.findOneBy({ id: id });

if (!found) throw new NotFoundException(`The task id ${id} NOT found`);

return found;

}

Test : <http://localhost:3000/products/a1437017-f5f3-49c8-b5d5-cfe5f453c585>

<http://localhost:3000/products>

//————————————————————————Delete Products——————————

11. **Create ProductController.ts**

//Delete Task

@Delete('/:id')

async deleteProduct(@Param('id') id: string): Promise<void> {

console.log('Id is :' + id);

const result = await this.productService.deleteTask(id);

console.log(result);

}

**12.Create ProductService.ts**

//Delete Task

async deleteTask(id: string): Promise<void> {

const result = await this.productRepository.delete(id);

console.log(result);

if (result.affected === 0) {

throw new NotFoundException(`Task with id "${id}" not found `);

}

}

<http://localhost:3000/products/dfdfh6bfhgfjghjj>

//————————————Update Products—————————

12. **Create ProductController.ts**

@Patch('/:id/price')

updateTask(

@Param('id') id: string,

@Body() updateProductDto: UpdateProductDto,

): Promise<Product> {

const { price } = updateProductDto;

return this.productService.updateProduct(id, price);

}

13**.Create ProductService.ts**

async updateProduct(id: string, price: number): Promise<Product> {

console.log('Id and price:' + id + ' ' + price);

const product = await this.getProductById(id);

product.price = price;

await this.productRepository.save(product);

return product;

}

A screenshot of a computer

Description automatically generated

Apply Validator to Enum for only specified value to be enter during the updating in context to Task Project

1. Install class-validator
2. Apply decorator
3. **Update-dto.ts**

@IsEnum(TaskStatus)

status: TaskStatus;

A screenshot of a computer

Description automatically generated