NESTJS

ENG. MOSTAFA MAHMOUD HELMY
ITI-SMART VILLAGE
01009215140

DAY 2

- Validation With DTO [Data Transfer Object]
- Pipes
- Middleware
- Connect With MongoDB [mongoose]

Validation

1) npm i class-validator
2) npm i class-transformer (Pipe)

Validation

Users.controller.ts

```
Import { UsePipes,ValidationPipe } from `@nestjs/common`

@UsePipes(ValidationPipe)
@Post()
  Create(@Body( ) user : UserWithDTO ){
      return this.userService.Create(user);
   }
```

Users.Service.ts

```
Create(user : UserWithDTO ){
      // Ur Logic To Add New User
   }
Update(id: number, user: UserWithDTO ){
     // Ur Logic To Update Exist User
}
```

Users.dto.ts

```
Import {IsNotEmpty, IsString, IsNumber } from `class-validator`;

export class UserWithDTO{
    @IsNotEmpty()
    @IsString()
    name: string;

   @IsNotEmpty()
    @IsNumber()
    age: number;

    ……
}
```

Pipes

Users.controller.ts

```
import {ParseIntPipe } from `@nestjs/common`

@Get(`:id`)
 findOne(@Param( `id` ) id, ParseIntPipe){
      return this.userService.findOne( id );
 }
```

- ValidationPipe
- ParseIntPipe
- ParseFloatPipe
- ParseBoolPipe
- ParseArrayPipe
- ParseUUIDPipe
- ParseEnumPipe
- DefaultValuePipe
- ParseFilePipe

Middleware

LoggerMW.ts

```
Import { NestMiddleware } from `@nestjs/common`;
Import { Request, Response, NextFunction } from `express`;
export class LoggerMW implements NestMiddleware{

  use(req : Request , res : Response , next : NextFunction){
        //Ur Logic ➔ next() ;
    }
}
```

app.module.ts

```
Import { NestModule, MiddlewareConsumer, RequestMethod } from `@nestjs/common`;

export class AppModule implements NestModule{

  configure(consumer : MiddlewareConsumer){
        consumer.apply(LoggerMW).forRoutes(`users`);
        consumer.apply(LoggerMW).forRoutes (
                {path:`users`, method: RequestMethod.GET}
            );
    }
}
```

Mongoose

1) npm i @nestjs/mongoose
2) npm i mongoose

Mongoose

app.module.ts

```
Import {MongooseModule} from `@nestjs/mongoose`
@Module({
    imports: [
        UsersModule,
        OrdersModule,
        ProductsModule,
        MongooseModule.forRoot( `mongodb:localhost:27017/Orders`)
     ],
    controllers: [ ],
    providers: [ ]
})
export class AppModule{ }
```

orders.schema.ts

```
Import * as mongoose from `mongoose`
export const OrderSchema =
            new mongoose.Schema({
                orderId: Number,
                userId: Number, …..
            })
```

orders.module.ts

```
import {MongooseModule} from `nestjs/mongoose`
@Module({
    imports: [
      MongooseModule.forFeature( [{
          name: `orders`,
          schema: OrderSchema
      }] )
    ]
})
```

orders.service.ts

```
Import {InjectModel} from `@nestjs/mongoose`
export class OrderService{
   constructor(
        @InjectModel(`orders`) private OrderModel
    ){ }
}
```

Mongoose

orders.service.ts

```
Import {InjectModel} from `@nestjs/mongoose`
Import {Model} from `mongoose`;
export class OrderService{
   constructor(
        @InjectModel(`orders`) private Model<IOrder>
    ){ }
}
```

orders.interface.ts

```
Import {Document} from `mongoose`;
export interface IOrder extends Document{
   orderId: number;
   userId: number;
   …..
}
```