Fifth assignment Microservices:

1. First, create 2 nest projects – make sure you add those to separate workspace folders in VS code.

Nest new micro-services

Nest new micro-service-client-app

1. Npm Install in both:

Npm I @nestjs/microservices

1. Create a microservice:

**micro-service/src/main.ts**

import { NestFactory } from '@nestjs/core';

import { AppModule } from './app.module';

import { Transport, MicroserviceOptions } from '@nestjs/microservices';

const msOptions: MicroserviceOptions = {

transport: Transport.TCP,

options: {

host: '127.0.0.1',

port: 3030,

}

}

async function bootstrap() {

const app = await NestFactory.createMicroservice(AppModule, msOptions);

app.listen(()=> console.log(`Microservice is listening on 127.0.0.1:3030`));

}

bootstrap();

// async function bootstrap() {

// const app = await NestFactory.create(AppModule);

// await app.listen(3000);

// }

// bootstrap();

1. Microservice controller now we use MessagePattern and EventPattern instead of the http method.

MessagePattern – request response

EventPattern – only request

micro-service/src/app.controller.ts

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

import { MessagePattern } from '@nestjs/microservices';

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

@Controller()

export class AppController {

constructor(private readonly appService: AppService) {}

@MessagePattern('hello')

getHello(): string {

console.log('MS controller getHello()');

return this.appService.getHello();

}

@MessagePattern('sum')

async sum(data: number[]): Promise<number> {

console.log('Microservice - sum - ', {data});

return this.appService.sum(data);

}

}

1. micro-service/src/app.service.ts

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

@Injectable()

export class AppService {

getHello(): string {

return 'Hello Microservices!';

}

async sum(data: number[]): Promise<number> {

return (data || []).reduce((a, b) => Number(a) + Number(b));

}

}

**Client APP**

micro-service-client-app/src/app.controller.ts

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

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

@Controller()

export class AppController {

constructor(private readonly appService: AppService) {}

@Get()

async getHello() {

console.log(`App Controller getHello()`);

return await this.appService.getHello();

}

@Post('sum')

async sum(@Body('data') data: number[]){

console.log({data});

return await this.appService.sum(data);

}

}

First option (commented out) config in service

micro-service-client-app/src/app.service.ts

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

import { ClientProxy, ClientProxyFactory, Transport } from '@nestjs/microservices';

@Injectable()

export class AppService {

constructor(@Inject('MATH\_SERVICE') private client: ClientProxy) {}

// private client: ClientProxy;

// constructor() {

// this.client = ClientProxyFactory.create({

// transport: Transport.TCP,

// options: {

// host: '127.0.0.1',

// port: 3030,

// },

// });

// }

async getHello() {

console.log('App service getHello()');

return await this.client.send<string>('hello', {})

}

async sum(data: number[]){

return await this.client.send<number, number[]>('sum', data);

}

async createUser(data: any) {

this.client.emit('user\_created', data);

}

}

second option config in module

micro-service-client-app/src/app.module.ts

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

import { ClientsModule, Transport } from '@nestjs/microservices';

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

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

@Module({

imports: [

ClientsModule.register([

{

name: 'MATH\_SERVICE',

transport: Transport.TCP,

options:{

host:'127.0.0.1',

port: 3030

}

}

])

],

controllers: [AppController],

providers: [AppService],

})

export class AppModule {}

**TEST**

**Add Redis**

1. npm install redis : in both microservice and microservice- client
2. Create local redis on docker : docker run --name recorder-redis -p 6379:6379 -d redis:alpine

micro-service/src/main.ts

import { NestFactory } from '@nestjs/core';

import { AppModule } from './app.module';

import { Transport, MicroserviceOptions } from '@nestjs/microservices';

// const msOptions: MicroserviceOptions = {

// transport: Transport.TCP,

// options: {

// host: '127.0.0.1',

// port: 3030,

// }

// }

const msOptions: MicroserviceOptions = {

transport: Transport.REDIS,

options: {

url: 'redis://localhost:6379',

},

}

async function bootstrap() {

const app = await NestFactory.createMicroservice(AppModule, msOptions);

app.listen(() => console.log(`Microservice is listening to redis...`));

}

bootstrap();

**micro-service/src/app.controller.ts**

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

import { EventPattern, MessagePattern } from '@nestjs/microservices';

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

@Controller()

export class AppController {

constructor(private readonly appService: AppService) {}

@MessagePattern('hello')

getHello(): string {

console.log('MS controller getHello()');

return this.appService.getHello();

}

@MessagePattern('sum')

async sum(data: number[]): Promise<number> {

console.log('Microservice - sum - ', {data});

return this.appService.sum(data);

}

@EventPattern('user\_created')

async handleUserCreated(data: any): Promise<void> {

console.log('Microservice - handleUserCreated - ', { data });

// run other business logic statments here...

// no need to return a response

}

}

**micro-service-client-app/src/app.module.ts**

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

import { ClientsModule, Transport } from '@nestjs/microservices';

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

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

@Module({

imports: [

ClientsModule.register([

// {

// name: 'MATH\_SERVICE',

// transport: Transport.TCP,

// options:{

// host:'127.0.0.1',

// port: 3030

// }

// }

{

name: 'MATH\_SERVICE',

transport: Transport.REDIS,

options: {

url: 'redis://localhost:6379'

}

}

])

],

controllers: [AppController],

providers: [AppService],

})

export class AppModule {}

**micro-service-client-app/src/app.controller.ts**

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

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

@Controller()

export class AppController {

constructor(private readonly appService: AppService) {}

@Get()

async getHello() {

console.log(`App Controller getHello()`);

return await this.appService.getHello();

}

@Post('sum')

async sum(@Body('data') data: number[]){

console.log({data});

return await this.appService.sum(data);

}

@Post('user')

async createUser(@Body() data: any) {

console.log('App Controller createUser()', { data });

return await this.appService.createUser(data);

}

}

**micro-service-client-app/src/app.service.ts**

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

import { ClientProxy, ClientProxyFactory, Transport } from '@nestjs/microservices';

@Injectable()

export class AppService {

constructor(@Inject('MATH\_SERVICE') private client: ClientProxy) {}

// private client: ClientProxy;

// constructor() {

// this.client = ClientProxyFactory.create({

// transport: Transport.TCP,

// options: {

// host: '127.0.0.1',

// port: 3030,

// },

// });

// }

async getHello() {

console.log('App service getHello()');

return await this.client.send<string>('hello', {})

}

async sum(data: number[]){

return await this.client.send<number, number[]>('sum', data);

}

async createUser(data: any) {

this.client.emit('user\_created', data);

}

}

**Test**