

You can now subscribe to get stories delivered directly to your inbox.

Got it

Nestjs and mysql in 5 Minutes

A quick tutorial for building scalable [Node.js](#) applications



Shai Ben Shimol Jan 26, 2019 · 4 min read



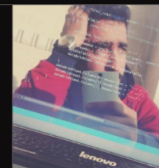
🔧 In this tutorial I will show you how easy is to setup and execute a Nestjs project using the following tech stack:

- 🔥 **Nestjs** (Modules, Controllers, Repositories , TypeORM and Entities)
- Node.js & NPM:** (<https://nodejs.org/en/download/>)
- Mysql 5.7:** (<https://dev.mysql.com/downloads/mysql/5.7.html>)
- Vscode:** (<https://code.visualstudio.com/download>)

Why Nest js? · Issue #693 · nestjs/nest

Whats Different about nest to another frameworks? Why we have to use nest.js? Please say your opinion.

github.com



Install and create Nestjs project

```
npm install -g typescript
npm install -g @nestjs/cli
```

```
⚡ Creating your Nest project...
👉 We have to collect additional information:

? description: description
? version: 0.0.0
? author:

🌟 Thank you for your time!
```

```
nest new my-nest-project
```

```
c:\> cd my-nest-project
```

Open “my-nest-project” in vscode with code . command

Project Structure

```
└─ node_modules
  └─ src
    └─ TS app.controller.spec.ts
    └─ TS app.controller.ts
    └─ TS app.module.ts
    └─ TS app.service.ts
    └─ TS main.ts
  └─ test
    └─ TS app.e2e-spec.ts
    └─ {} jest-e2e.json
  └─ .prettierrc
  └─ {} nest-cli.json
  └─ {} nodemon-debug.json
  └─ {} nodemon.json
  └─ {} package.json
  └─ README.md
  └─ {} tsconfig.build.json
  └─ {} tsconfig.json
  └─ {} tsconfig.spec.json
  └─ {} tslint.json
```

- `node_modules`: include the packages modules
- `src`: include the app source files
- `test`: end-to-end test app
- `nest-cli.json`: The root level of an `Nestjs` workspace provides workspace-wide and project-specific configuration defaults for build and development tools provided by the `Nestjs`. Path values given in the configuration are relative to the root workspace folder.
- `package.json`: lists the packages your project depends on.

. . .

Module, Service and Controller Meaning:

Modules — Architecture

Every `Nestjs` app has at least one `@Module()` class — root module. The root `@Module()` for an app is so named because it can include child `@Module()` in A hierarchy of any depth.

The most important properties are as follows:

- **imports**: other modules whose exported classes are needed by component templates declared in *this Module*.
- **controllers**: the set of controllers which have to be created.
- **providers**: creators of services that this `NgModule` contributes to the global collection of services; they become accessible in all parts of the app.
- **exports**: the subset of declarations that should be visible and usable in the *component templates* of other Modules.

Services

Service is a layer category encompassing any value, function, or feature that an app needs. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

Controllers

A controller is a class that handles HTTP requests. The public methods of the Controller are called action methods or simply actions. When the `Nestjs` Framework receives a request, it routes the request to an action.

To determine which action to invoke, the framework uses a routing table.

. . .

Let's Get Started

Let's Get Started

Create users module, service and controller.

```
nest g module users
nest g service users
nest g controller users
nest g class users/user.entity
```

```
src
└─ users
   TS user-entity.ts
   TS users.controller.ts
   TS users.module.ts
   TS users.service.ts
```

In this section we'll not use spec file.

Installing Mysql And typeORM

```
npm install --save @nestjs/typeorm typeorm mysql
```

Top highlight

Time to write some code!

- Open user.entity.ts file and Type

```
25
26   async deleteUser(user: User) {
27     this.usersRepository.delete(user);
28   }
29 }
```

users.service.ts hosted with ❤ by GitHub

[view raw](#)

- Open users.controller.ts and type

```
13     }
14
15     @Post()
16     create(@Body() user: User) {
17         return this.service.createUser(user);
18     }
19
20     @Put()
21     update(@Body() user: User) {
22         return this.service.updateUser(user);
23     }
24
25     @Delete('/:id')
26     deleteUser(@Param() params) {
27         return this.service.deleteUser(params.id);
28     }
29 }
```

users.controller.ts hosted with ❤ by GitHub

[view raw](#)

- Create ormconfig.json file in the root project with the following attributes

```
{
  "type": "mysql",
  "host": "localhost",
  "port": 3306,
  "username": "root",
  "password": "root",
  "database": "my_nestjs_project",
  "entities": ["src/**/*.entity{.ts,.js}"],
  "synchronize": true
}
```

- Open users.module.ts file and it looks like

```
10 })
11 export class AppModule {}
```

app.module.ts hosted with ❤ by GitHub

[view raw](#)

...

Open terminal in vscode and run

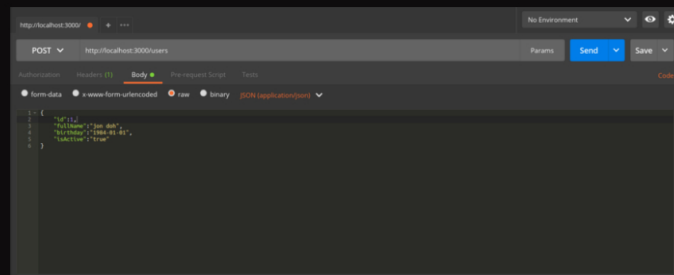
```
npm run start
```

```
[Nest] 29030 - 1/26/2019, 7:28:50 PM [NestFactory] Starting Nest application...
[Nest] 29030 - 1/26/2019, 7:28:50 PM [InstanceLoader] AppModule dependencies initialized +56ms
[Nest] 29030 - 1/26/2019, 7:28:50 PM [InstanceLoader] TypeOrmModule dependencies initialized +0ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [InstanceLoader] TypeOrmCoreModule dependencies initialized +150
[Nest] 29030 - 1/26/2019, 7:28:51 PM [InstanceLoader] TypeOrmModule dependencies initialized +0ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [InstanceLoader] UsersModule dependencies initialized +2ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [RoutesResolver] UsersController (/users): +39ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [RouterExplorer] Mapped [/ GET] route +2ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [RouterExplorer] Mapped [/ POST] route +0ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [RouterExplorer] Mapped [/ PUT] route +0ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [RouterExplorer] Mapped [/ DELETE] route +1ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [RouterExplorer] Mapped [/all GET] route +0ms
[Nest] 29030 - 1/26/2019, 7:28:51 PM [NestApplication] Nest application successfully started +1ms
```

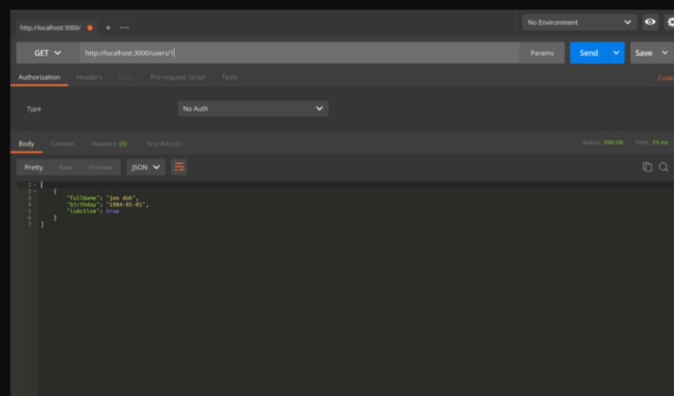
Nest compilation

Testing via POSTMAN

- Create User



- Get User by id



Conclusion

We made a good progress in this very first part of building Nestjs application. We got most of the architecture of Nestjs decisions.

[My ko-fi](#)

🍻 Cheers! 🍻

...

👉 My Others Stories:



Build Effective Web Application

🔥 With type-collector package 🔥

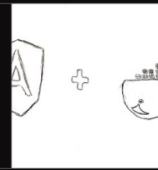
medium.com



Docker for Angular 7

Deploy Angular 7 to Docker Hub in 5 Minutes

medium.com



JavaScript

Nodejs

Nestjs

Expressjs

Typescript



452



9



More from Shai Ben Shimol

Software engineer, Software Architect Specialist in Web and Mobile development...

Follow

