Introduction

Nest (NestJS) is a framework for building efficient, scalable Node.js server-side applications. It uses progressive JavaScript, is built with and fully supports TypeScript (yet still enables developers to code in pure JavaScript) and combines elements of OOP (Object Oriented Programming), FP (Functional Programming), and FRP (Functional Reactive Programming).

Under the hood, Nest makes use of robust HTTP Server frameworks like Express (the default) and optionally can be configured to use Fastify as well!

Nest provides a level of abstraction above these common Node.js frameworks (Express/Fastify), but also exposes their APIs directly to the developer. This gives developers the freedom to use the myriad of third-party modules which are available for the underlying platform.

Philosophy

In recent years, thanks to Node.js, JavaScript has become the “lingua franca” of the web for both front and backend applications. This has given rise to awesome projects like Angular, React and Vue, which improve developer productivity and enable the creation of fast, testable, and extensible frontend applications. However, while plenty of superb libraries, helpers, and tools exist for Node (and server-side JavaScript), none of them effectively solve the main problem of - **Architecture**.

Nest provides an out-of-the-box application architecture which allows developers and teams to create highly testable, scalable, loosely coupled, and easily maintainable applications. The architecture is heavily inspired by Angular.

Installation

To get started, you can either scaffold the project with the Nest CLI, or clone a starter project (both will produce the same outcome).

To scaffold the project with the Nest CLI, run the following commands. This will create a new project directory, and populate the directory with the initial core Nest files and supporting modules, creating a conventional base structure for your project. Creating a new project with the **Nest CLI** is recommended for first-time users. We'll continue with this approach in First Steps.

```
$ npm i -g @nestjs/cli
$ nest new project-name
```

> info **Hint** To create a new TypeScript project with stricter feature set, pass the `--strict` flag to the `nest new` command.

Alternatives

Alternatively, to install the TypeScript starter project with **Git**:

```
$ git clone https://github.com/nestjs/typescript-starter.git project
$ cd project
$ npm install
$ npm run start
```

> info **Hint** If you'd like to clone the repository without the git history, you can use degit.

Open your browser and navigate to `http://localhost:3000/`.

To install the JavaScript flavor of the starter project, use `javascript-starter.git` in the command sequence above.

You can also manually create a new project from scratch by installing the core and supporting files with **npm** (or **yarn**). In this case, of course, you'll be responsible for creating the project boilerplate files yourself.

```
$ npm i --save @nestjs/core @nestjs/common rxjs reflect-metadata
```