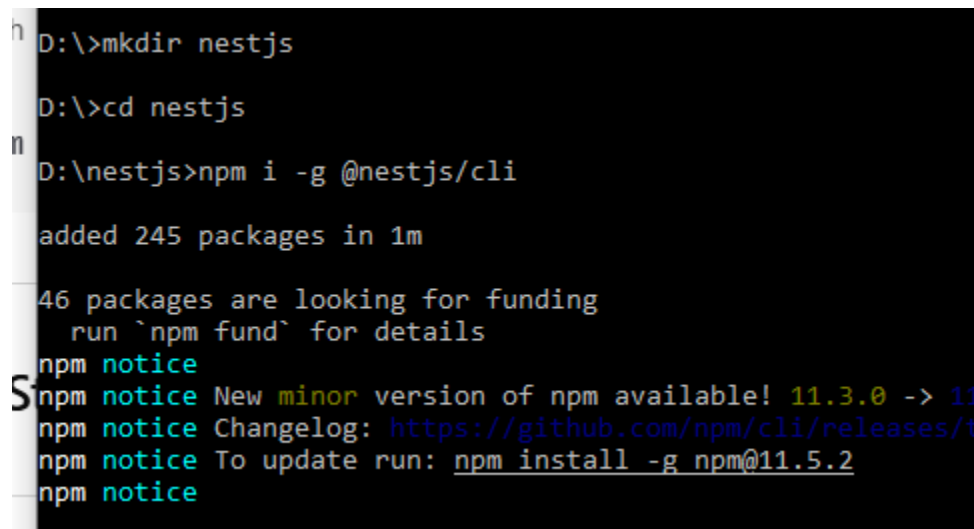


Sure! Here's A Quick And Practical Example Of How To Build A 5-6 Page Website (Multi-Page Restful Website) Using Nestjs.

□ Prerequisites

- Node.js installed
- NestJS CLI installed:

```
npm i -g @nestjs/cli
```



```
h D:\>mkdir nestjs
D:\>cd nestjs
n D:\nestjs>npm i -g @nestjs/cli
added 245 packages in 1m
46 packages are looking for funding
  run `npm fund` for details
npm notice
S npm notice New minor version of npm available! 11.3.0 -> 11.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.5.2
npm notice To update run: npm install -g npm@11.5.2
npm notice
```

□ Step-by-Step Setup

1. Create a NestJS Project

```
nest new my-website
cd my-website
```

```
D:\nestjs>
D:\nestjs>nest new my-website
  We will scaffold your app in a few seconds..

√ Which package manager would you  to use? npm
CREATE my-website/.prettierrc (54 bytes)
CREATE my-website/eslint.config.mjs (869 bytes)
CREATE my-website/nest-cli.json (179 bytes)
CREATE my-website/package.json (2052 bytes)
CREATE my-website/README.md (5126 bytes)
CREATE my-website/tsconfig.build.json (101 bytes)
CREATE my-website/tsconfig.json (702 bytes)
CREATE my-website/src/app.controller.ts (286 bytes)
CREATE my-website/src/app.module.ts (259 bytes)
CREATE my-website/src/app.service.ts (150 bytes)
CREATE my-website/src/main.ts (236 bytes)
CREATE my-website/src/app.controller.spec.ts (639 bytes)
CREATE my-website/test/jest-e2e.json (192 bytes)
CREATE my-website/test/app.e2e-spec.ts (699 bytes)

▶■■■■ Installation in progress... 
```

Then

```
C:\WINDOWS\system32\cmd.exe
CREATE my-website/package.json (2052 bytes)
CREATE my-website/README.md (5126 bytes)
CREATE my-website/tsconfig.build.json (101 bytes)
CREATE my-website/tsconfig.json (702 bytes)
CREATE my-website/src/app.controller.ts (286 bytes)
CREATE my-website/src/app.module.ts (259 bytes)
CREATE my-website/src/app.service.ts (150 bytes)
CREATE my-website/src/main.ts (236 bytes)
CREATE my-website/src/app.controller.spec.ts (639 bytes)
CREATE my-website/test/jest-e2e.json (192 bytes)
CREATE my-website/test/app.e2e-spec.ts (699 bytes)

√ Installation in progress...

  Successfully created project my-website
  Get started with the following commands:

$ cd my-website
$ npm run start

Thanks for installing Nest
Please consider donating to our open collective
to help us maintain this package.

  Donate: https://opencollective.com/nest

D:\nestjs>
```

2. Generate Pages (Controllers + Services)

Let's say you want 6 pages: **Home, About, Contact, Services, Blog, and FAQ.**

```
nest g controller pages/home
nest g controller pages/about
nest g controller pages/contact
nest g controller pages/services
nest g controller pages/blog
nest g controller pages/faq
```

```
C:\WINDOWS\system32\cmd.exe
D:\nestjs>cd my-website

D:\nestjs\my-website>nest g controller pages/home
CREATE src/pages/home/home.controller.ts (101 bytes)
CREATE src/pages/home/home.controller.spec.ts (496 bytes)
UPDATE src/app.module.ts (338 bytes)

D:\nestjs\my-website>nest g controller pages/about
CREATE src/pages/about/about.controller.ts (103 bytes)
CREATE src/pages/about/about.controller.spec.ts (503 bytes)
UPDATE src/app.module.ts (421 bytes)

D:\nestjs\my-website>nest g controller pages/contact
CREATE src/pages/contact/contact.controller.ts (107 bytes)
CREATE src/pages/contact/contact.controller.spec.ts (517 bytes)
UPDATE src/app.module.ts (512 bytes)

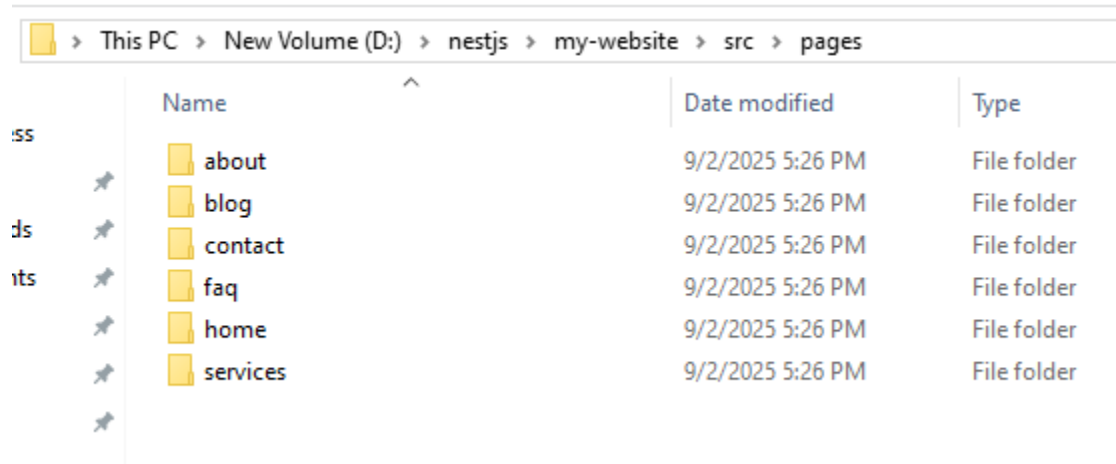
D:\nestjs\my-website>nest g controller pages/services
CREATE src/pages/services/services.controller.ts (109 bytes)
CREATE src/pages/services/services.controller.spec.ts (524 bytes)
UPDATE src/app.module.ts (607 bytes)

D:\nestjs\my-website>nest g controller pages/blog
CREATE src/pages/blog/blog.controller.ts (101 bytes)
CREATE src/pages/blog/blog.controller.spec.ts (496 bytes)
UPDATE src/app.module.ts (686 bytes)

D:\nestjs\my-website>nest g controller pages/faq
CREATE src/pages/faq/faq.controller.ts (99 bytes)
CREATE src/pages/faq/faq.controller.spec.ts (489 bytes)
```

(You don't need services unless doing logic, so we skip `nest g service`.)

After it open your project folder you will see like this :-



Name	Date modified	Type
about	9/2/2025 5:26 PM	File folder
blog	9/2/2025 5:26 PM	File folder
contact	9/2/2025 5:26 PM	File folder
faq	9/2/2025 5:26 PM	File folder
home	9/2/2025 5:26 PM	File folder
services	9/2/2025 5:26 PM	File folder

3. Define Routes in Controllers

Home page: `pages/home/home.controller.ts`

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

@Controller()
export class HomeController {
  @Get()
  getHome(): string {
    return 'Welcome to the Home Page!';
  }
}
```

About page: `pages/about/about.controller.ts`

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

@Controller('about')
export class AboutController {
  @Get()
  getAbout(): string {
    return 'About Us Page';
  }
}
```

❓ Folder Structure:-

Assuming we're placing all these under:

```
src/
├── pages/
│   ├── contact/
│   │   └── contact.controller.ts
│   ├── services/
│   │   └── services.controller.ts
│   ├── blog/
│   │   └── blog.controller.ts
│   └── faq/
│       └── faq.controller.ts
```

❑ 1 contact page. `pages/contact/contact.controller.ts`

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

@Controller('contact')
export class ContactController {
  @Get()
  getContact(): string {
    return 'This is the Contact Page.';
  }
}
```

❑ 2 services page. `pages/services/services.controller.ts`

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

@Controller('services')
export class ServicesController {
  @Get()
  getServices(): string {
    return 'This is the Services Page.';
  }
}
```

❑ 3 blog page. `pages/blog/blog.controller.ts`

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

@Controller('blog')
export class BlogController {
  @Get()
  getBlog(): string {
    return 'This is the Blog Page.';
  }
}
```

□ 4 **faq page.** `pages/faq/faq.controller.ts`

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

@Controller('faq')
export class FaqController {
  @Get()
  getFaq(): string {
    return 'This is the FAQ Page.';
  }
}
```

Add Controllers to `src/app.module.ts:-`

Edit `src/app.module.ts` and register the controllers:-

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

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

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

import { HomeController } from './pages/home/home.controller';

import { AboutController } from './pages/about/about.controller';

import { ContactController } from './pages/contact/contact.controller';

import { ServicesController } from './pages/services/services.controller';

import { BlogController } from './pages/blog/blog.controller';

import { FaqController } from './pages/faq/faq.controller';
```

```
@Module({
  imports: [],
  controllers: [HomeController, AboutController, ContactController, ServicesController,
BlogController, FaqController],
  providers: [AppService],
})
export class AppModule {}
```

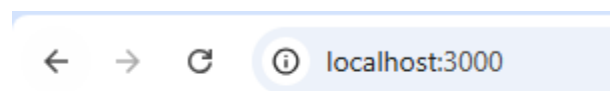
Run Your App

npm run start

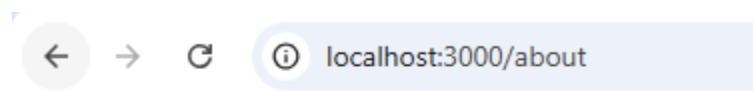
Open in browser:

- <http://localhost:3000/contact>
- <http://localhost:3000/services>
- <http://localhost:3000/blog>
- <http://localhost:3000/faq>

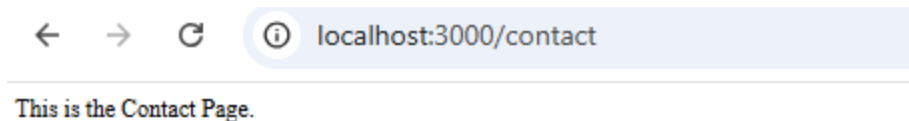
output :-



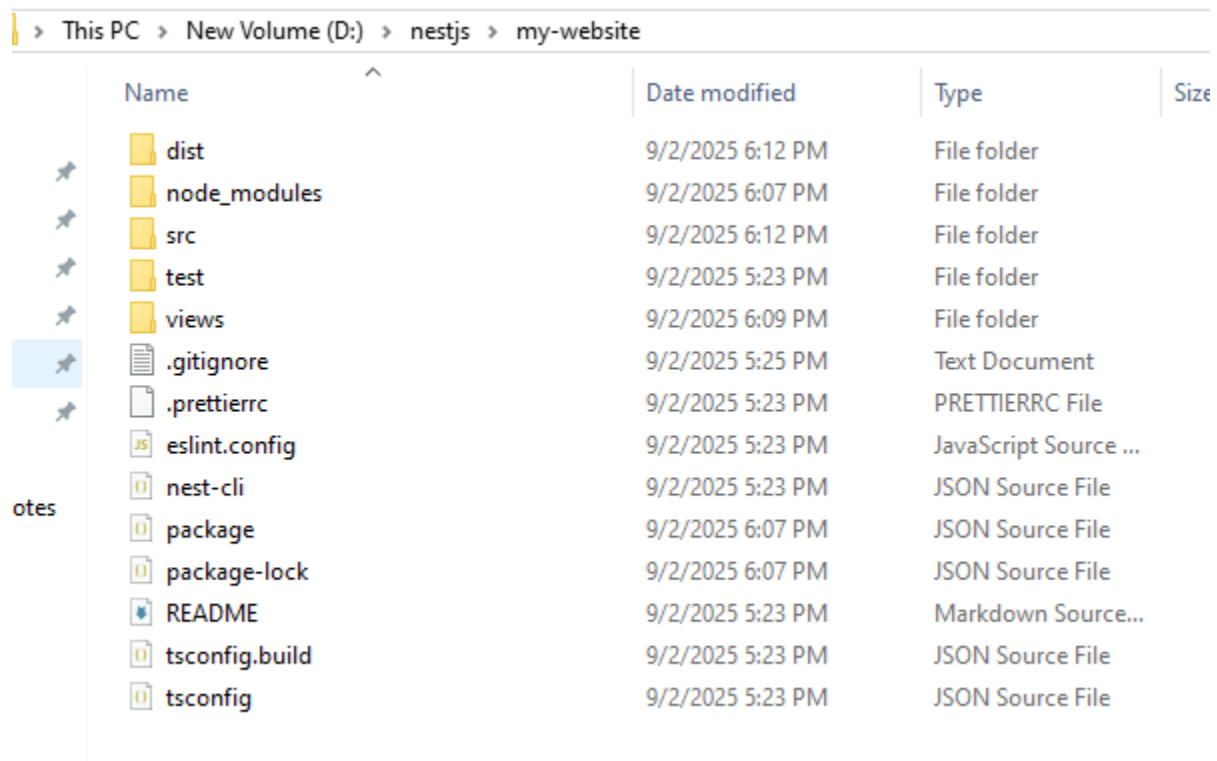
Welcome to the Home Page!



About Us Page



Now to display content with ejs format :-



Using **EJS** with NestJS allows you to serve **dynamic HTML pages** — perfect for building a real website with templates.

□ Goal

- Set up **EJS** as the view engine.

- Render a dynamic **Home Page** at `/`.
 - Use EJS for future pages like `/contact`, `/services`, etc.
-

□ **Step-by-Step: Setup EJS with NestJS**

□ **1. Install EJS**

In your NestJS project:

```
npm install ejs
```

□ **2. Enable View Engine in `main.ts`**

Open `src/main.ts` and modify it:

```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { NestExpressApplication } from '@nestjs/platform-express';
import { join } from 'path';

async function bootstrap() {
  const app = await NestFactory.create<NestExpressApplication>(AppModule);

  // Set EJS as the view engine
  app.setBaseViewsDir(join(__dirname, '..', 'views'));
  app.setViewEngine('ejs');

  await app.listen(3000);
}
bootstrap();
```

□ **3. Create `my-website/views` Folder & `home.ejs` File**

In your root directory, create:

```
/views/
└─ home.ejs
```

Add some HTML content to `home.ejs`:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
</head>
<body>
  <h1>Welcome to the Home Page</h1>
  <p>This page is rendered using <strong>EJS</strong>.</p>
</body>
</html>
```

4. Update `home.controller.ts` to Render the View

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

@Controller()
export class HomeController {
  @Get()
  @Render('home') // Looks for views/home.ejs
  getHome() {
    return {}; // Pass data here if needed
  }
}
```

5. Run the Server

```
npm run start
```

6. Open in Browser

Go to:

```
http://localhost:3000/
```

Optional: Pass Data to EJS Template

You can pass dynamic data like this pages/home/[home.controller.ts](#) :

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

@Controller()

export class HomeController {

  @Get()

  @Render('home') // Looks for views/home.ejs

  getHome() {

    return {title: 'My NestJS Website'}; // Pass data here if needed

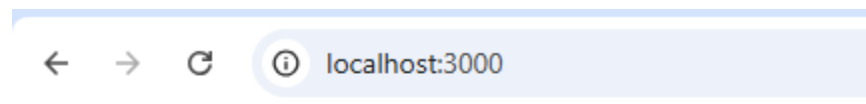
  }

}
```

Update my-website/views/home.ejs to use the data:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
</head>
<body>
  <h1><%= title %></h1>

  <p>This page is rendered using <strong>EJS</strong>.</p>
</body>
</html>
```



My NestJS Website

This page is rendered using EJS.