Beginner's Guide to Building a Simple API with Nest.js

Introduction

In this tutorial, we'll explore how to create a basic API using Nest.js, a powerful and extensible Node.js framework. We'll build an API for managing a collection of books, similar to our previous Express.js example.

Prerequisites

Before we get started, make sure you have the following:

- Node.js and npm installed
- Basic knowledge of JavaScript

Step 1: Setting Up Your Project

First, create a new directory for your project and navigate to it in your terminal.

- Install Nest CLI globally:

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

- Create a new Nest.js project:

```
nest new book-api
```

Step 2: Project Structure

Inside your project folder (book-api), you'll find a structure similar to this:

```
book-api/
|-- src/
|   |-- app.controller.ts
|   |-- app.module.ts
|   |-- app.service.ts
|-- main.ts
|-- package.json
|-- tsconfig.json
```

Step 3: Create a Book Module

- Generate a book module:

```
nest generate module books
```

- Inside the books folder, create a books.controller.ts file to define your API routes for managing books.

Step 4: Define Book Entity

Create a book.entity.ts file inside the books folder to define the Book entity:

```
// books/book.entity.ts
export class Book {
  id: number;
  title: string;
  author: string;
}
```

Step 5: Implement the Book Service

Inside the books folder, create a books.service.ts file to implement your book service logic:

```ts
// books/books.service.ts
import { Injectable } from '@nestjs/common';
import { Book } from './book.entity';

@Injectable()
export class BooksService {
  private readonly books: Book[] = [];

  getAllBooks(): Book[] {
    return this.books;
  }

  createBook(book: Book): void {
    this.books.push(book);
  }
}
```

Step 6: Implement the Book Controller

In the books folder, open the books.controller.ts file and define your API routes for managing books using decorators:

```ts
// books/books.controller.ts
import { Controller, Get, Post, Body } from '@nestjs/common';
import { Book } from './book.entity';
import { BooksService } from './books.service';

@Controller('books')
export class BooksController {
  constructor(private readonly booksService: BooksService) {}

  @Get()
  getAllBooks(): Book[] {
    return this.booksService.getAllBooks();
  }

  @Post()
  createBook(@Body() book: Book): void {
    this.booksService.createBook(book);
  }
}
```

Step 7: Configure the App Module

Open app.module.ts in the src folder and import the BooksModule:

```ts
import { Module } from '@nestjs/common';
import { BooksModule } from './books/books.module';

@Module({
  imports: [BooksModule],
})
export class AppModule {}
```

Step 8: Running the Application

Start the Nest.js application:

```
npm run start
```

Step 9: Testing the API

You can test your API using tools like Postman or by sending HTTP requests using your preferred method.

Conclusion:

In this tutorial, we've explored the basics of building a simple API using Nest.js, a powerful and flexible framework for Node.js. We've created routes for managing books, similar to our previous Express.js example. Nest.js offers many features and is well-suited for building complex APIs and applications.