REST API using TypeScript and Node.js

### **Installing Typescript**

### npm install -g typescript

### **Initialize Node.js**

npm init –y

### **Install project dependencies**

npm install typescript ts-node express @types/express morgan @types/morgan axios @types/axios nodemon

* **TypeScript**: A TypeScript compiler with static set type definitions.
* **Ts-node:** Allows us to run and configure Typescript execution environments.
* **Express**: Node.js web application framework for setting and managing web-based server.
* **@types/express**: Type definitions for Express.
* **Morgan**: A Node.js HTTP request logger middleware for Node.js.
* **@types/morgan**: Type definitions for Morgan.
* **Axios**: A Node.js promise-based HTTP client library for Node.js, for sending HTTP requests to query and consume resources from APIs.
* **@types/Axios**: Type definitions for Axios.
* **Nodemon**: A server utility library for monitoring changes of the code on a text editor. It automatically restarts the server whenever code changes are detected.

To install all these libraries, run the following command.

### **Initialize Typescript**

tsc –init

{

"compilerOptions": {

"forceConsistentCasingInFileNames": true,

"module": "commonjs",

"esModuleInterop": true,

"outDir": "./build",

"rootDir": "./source",

"target": "es6",

"skipLibCheck": true,

"strict": true

}

}

### **Modify package.json**

"main": "source/server.ts",

"scripts": {

"dev": "nodemon source/server.ts",

"build": "rm -rf build/ && prettier --write source/ && tsc"

}

### **Setting up the application structure**

| package-lock.json

| package.json

| tsconfig.json

\---source

| server.ts

\---controllers

| posts.ts

\---routes

posts.ts

**controllers/posts.ts**

/\*\* source/controllers/posts.ts \*/

import { Request, Response, NextFunction } from 'express';

import axios, { AxiosResponse } from 'axios';

interface Post {

userId: Number;

id: Number;

title: String;

body: String;

}

// getting all posts

const getPosts = async (req: Request, res: Response, next: NextFunction) => {

// get some posts

let result: AxiosResponse = await axios.get(`https://jsonplaceholder.typicode.com/posts`);

let posts: [Post] = result.data;

return res.status(200).json({

message: posts

});

};

// getting a single post

const getPost = async (req: Request, res: Response, next: NextFunction) => {

// get the post id from the req

let id: string = req.params.id;

// get the post

let result: AxiosResponse = await axios.get(`https://jsonplaceholder.typicode.com/posts/${id}`);

let post: Post = result.data;

return res.status(200).json({

message: post

});

};

// updating a post

const updatePost = async (req: Request, res: Response, next: NextFunction) => {

// get the post id from the req.params

let id: string = req.params.id;

// get the data from req.body

let title: string = req.body.title ?? null;

let body: string = req.body.body ?? null;

// update the post

let response: AxiosResponse = await axios.put(`https://jsonplaceholder.typicode.com/posts/${id}`, {

...(title && { title }),

...(body && { body })

});

// return response

return res.status(200).json({

message: response.data

});

};

// deleting a post

const deletePost = async (req: Request, res: Response, next: NextFunction) => {

// get the post id from req.params

let id: string = req.params.id;

// delete the post

let response: AxiosResponse = await axios.delete(`https://jsonplaceholder.typicode.com/posts/${id}`);

// return response

return res.status(200).json({

message: 'post deleted successfully'

});

};

// adding a post

const addPost = async (req: Request, res: Response, next: NextFunction) => {

// get the data from req.body

let title: string = req.body.title;

let body: string = req.body.body;

// add the post

let response: AxiosResponse = await axios.post(`https://jsonplaceholder.typicode.com/posts`, {

title,

body

});

// return response

return res.status(200).json({

message: response.data

});

};

export default { getPosts, getPost, updatePost, deletePost, addPost };

**routes/posts.ts**

/\*\* source/routes/posts.ts \*/

import express from 'express';

import controller from '../controllers/posts';

const router = express.Router();

router.get('/posts', controller.getPosts);

router.get('/posts/:id', controller.getPost);

router.put('/posts/:id', controller.updatePost);

router.delete('/posts/:id', controller.deletePost);

router.post('/posts', controller.addPost);

export = router;

**server.ts**

import http from 'http';

import express, { Express } from 'express';

import morgan from 'morgan';

import routes from './routes/posts';

const router: Express = express();

/\*\* Logging \*/

router.use(morgan('dev'));

/\*\* Parse the request \*/

router.use(express.urlencoded({ extended: false }));

/\*\* Takes care of JSON data \*/

router.use(express.json());

/\*\* RULES OF OUR API \*/

router.use((req, res, next) => {

// set the CORS policy

res.header('Access-Control-Allow-Origin', '\*');

// set the CORS headers

res.header('Access-Control-Allow-Headers', 'origin, X-Requested-With,Content-Type,Accept, Authorization');

// set the CORS method headers

if (req.method === 'OPTIONS') {

res.header('Access-Control-Allow-Methods', 'GET PATCH DELETE POST');

return res.status(200).json({});

}

next();

});

/\*\* Routes \*/

router.use('/', routes);

/\*\* Error handling \*/

router.use((req, res, next) => {

const error = new Error('not found');

return res.status(404).json({

message: error.message

});

});

/\*\* Server \*/

const httpServer = http.createServer(router);

const PORT: any = process.env.PORT ?? 6060;

httpServer.listen(PORT, () => console.log(`The server is running on port ${PORT}`));

### **Starting the development server**

npm run dev

API List

<http://localhost:6060/posts>

<http://localhost:5000/posts/id>

**Node with TypeScript.postman\_collection**

{

"info": {

"\_postman\_id": "d247bdbc-8a3b-4b19-beb0-f52e2810793e",

"name": "Node with TypeScript",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "18048027"

},

"item": [

{

"name": "Fetch all posts",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "http://localhost:6060/posts",

"protocol": "http",

"host": [

"localhost"

],

"port": "6060",

"path": [

"posts"

]

}

},

"response": []

},

{

"name": "Fetch a single post",

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "http://localhost:6060/posts/3",

"protocol": "http",

"host": [

"localhost"

],

"port": "6060",

"path": [

"posts",

"3"

]

}

},

"response": []

},

{

"name": "Update a post",

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"title\": \"Another post title\",\r\n \"body\": \"Another post body\"\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "http://localhost:6060/posts/1",

"protocol": "http",

"host": [

"localhost"

],

"port": "6060",

"path": [

"posts",

"1"

]

}

},

"response": []

},

{

"name": "Delete a single post",

"request": {

"method": "DELETE",

"header": [],

"url": {

"raw": "http://localhost:6060/posts/1",

"protocol": "http",

"host": [

"localhost"

],

"port": "6060",

"path": [

"posts",

"1"

]

}

},

"response": []

},

{

"name": "Add a new post",

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"title\": \"New post\",\r\n \"body\": \"New post body\"\r\n}",

"options": {

"raw": {

"language": "javascript"

}

}

},

"url": {

"raw": "http://localhost:6060/posts",

"protocol": "http",

"host": [

"localhost"

],

"port": "6060",

"path": [

"posts"

]

}

},

"response": []

}

]

}