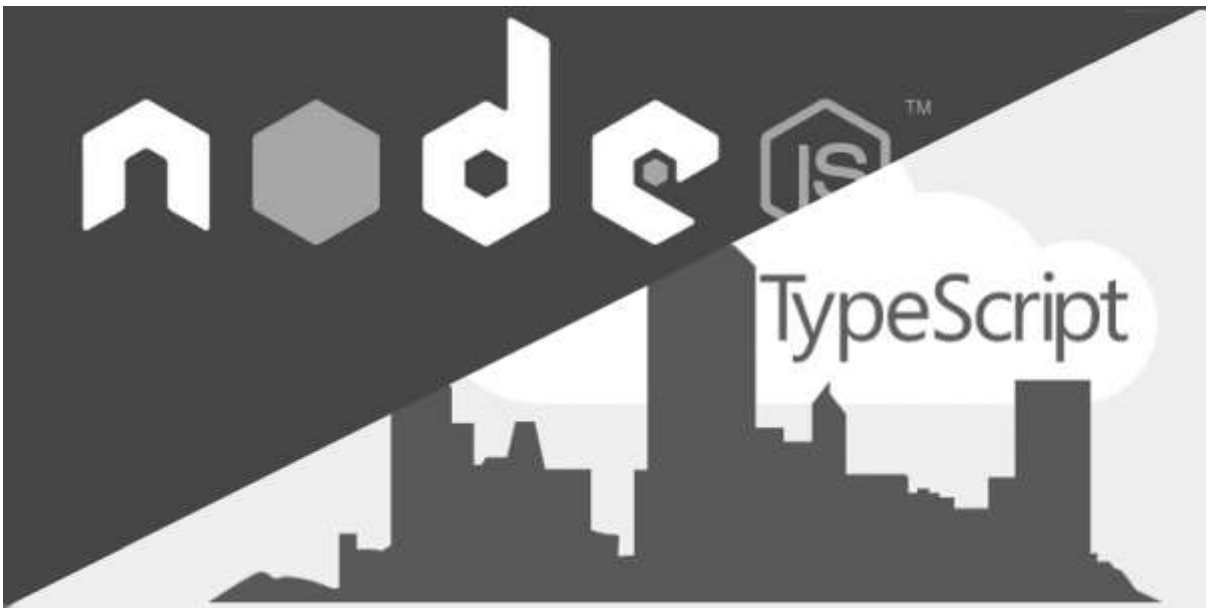
Building RESTful Web APIs with Node.js, Express, MongoDB and TypeScript — Part 2

Dale Nguyen
Apr 20, 2018 · 3 min read



(Image from OctoPerf)

There is a course about how to build a Web APIs on Lynda, but they didn't use TypeScript. So I decided to make one with TypeScript. There are lots of things that need to improve in this project. If you find one, please leave a comment. I'm appreciated that ;)

Part 1: Setting Up Project

Part 2: Implement routing and CRUD

Part 3: Using Controller and Model for Web APIs

Part 4: Connect Web APIs to MongoDB or others

Part 5: Security for our Web APIs

In part 2, I will build the routing for the API.

Step 1: Create TS file for routing

Remember in part 1 of this project. We save everything in **lib** folder. So I will create **routes** folder with a file named **crmRoutes.ts** that will save all the routes for this project.

```
// /lib/routes/crmRoutes.ts

import {Request, Response} from "express";

export class Routes {
    public routes(app): void {
        app.route('/')
        .get((req: Request, res: Response) => {
            res.status(200).send({
                message: 'GET request successfulll!!!!'
            })
        })
    }
}
```

After creating our first route, we need to import it to the **lib/app.ts**.

```
// /lib/app.ts

import * as express from "express";
import * as bodyParser from "body-parser";
import { Routes } from "./routes/crmRoutes";

class App {

    public app: express.Application;
    public routePrv: Routes = new Routes();

    constructor() {
        this.app = express();
        this.config();
        this.routePrv.routes(this.app);
    }
```

Now, you can send GET request to your application (http://localhost:3000) directly or by using Postman.

Step 2: Building CRUD for the Web APIs

I assume that you have a basic understanding of HTTP request (GET, POST, PUT and DELETE). If you don't, it is very simple:

- GET: for retrieving data
- POST: for creating new data
- PUT: for updating data
- DELETE: for deleting data

Now we will build the routing for building a contact CRM that saves, retrieves, updates and deletes contact info.

```
// /lib/routes/crmRoutes.ts

import {Request, Response} from "express";

export class Routes {

    public routes(app): void {

        app.route('/')
        .get((req: Request, res: Response) => {
            res.status(200).send({
                message: 'GET request successfulll!!!!'
            })
        })

        // Contact
        app.route('/contact')
        // GET endpoint
        .get((req: Request, res: Response) => {
        // Get all contacts
            res.status(200).send({
                message: 'GET request successfulll!!!!'
            })
```

```
        })

        // Contact detail
        app.route('/contact/:contactId')
        // get specific contact
        .get((req: Request, res: Response) => {
        // Get a single contact detail
            res.status(200).send({
                message: 'GET request successfulll!!!!'
            })
        })
        .put((req: Request, res: Response) => {
        // Update a contact
            res.status(200).send({
                message: 'PUT request successfulll!!!!'
            })
        })
        .delete((req: Request, res: Response) => {
        // Delete a contact
            res.status(200).send({
                message: 'DELETE request successfulll!!!!'
            })
        })
    }
}
```

Now the routes are ready for getting HTTP request. This is the end of **Part 2**. I will update **Part 3**, **Part 4 and Part 5** shortly. In case you need to jump a head. Please visit my github repository for the full code.

https://github.com/dalenguyen/rest-api-node-typescript

Nodejs Expressjs Mongodb Typescript API