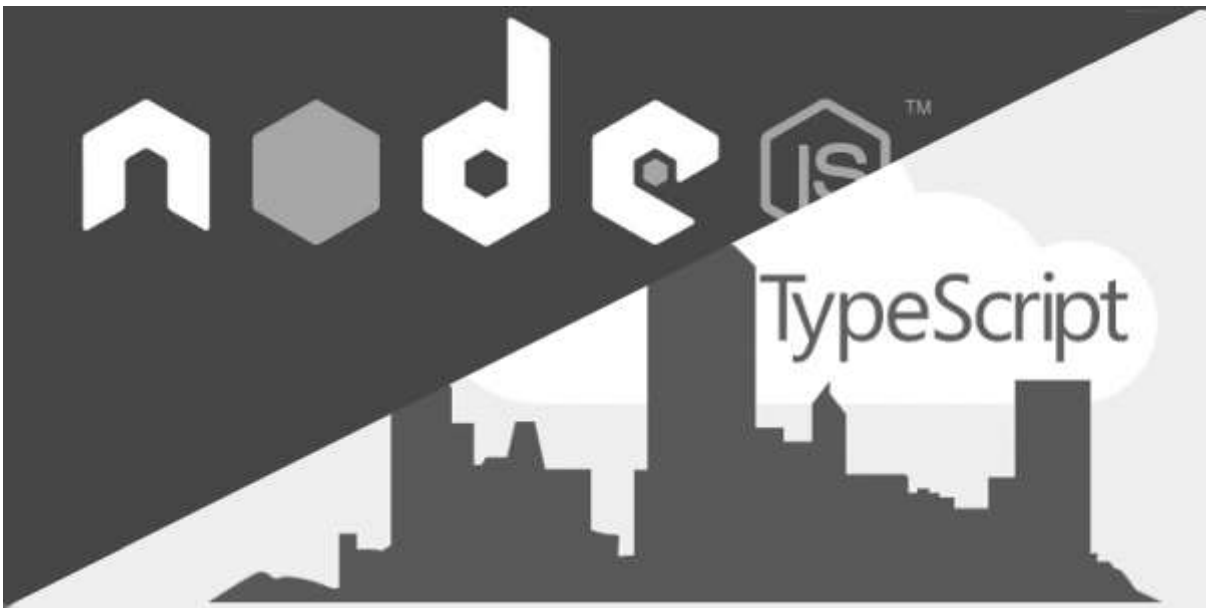
Building RESTful Web APIs with Node.js, Express, MongoDB and TypeScript — Part 5

Dale Nguyen
Jun 4, 2018 · 5 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 this part, I will show you various methods to secure your RESTful Web APIs. You should use at least one or combine those methods for a more secure API application.

And if you want to use services like mLab, compose…, they have already implemented a secured system on their end. All that you need to do is to follow their instructions to hook the database to your app.

Method 1: The first and foremost is that you should always use HTTPS over HTTP

For local testing, I will use OpenSSL on Windows to generate the key and certificate for HTTPS configuration. The process is similar on Mac or Linux.

After installing OpenSSL, I will open OpenSSL and start generating key and cert files.

```
OpenSSL> req -newkey rsa:2048 -nodes -keyout keytemp.pem -x509 -days
365 -out cert.pem
OpenSSL> rsa -in keytemp.pem -out key.pem
```

After that, we will move **key.pem** and **cert.pem** files to our project. They will be in the config folder.

Then we will edit the server.ts file to enable https.

```
import app from './app';
import * as https from 'https';
import * as fs from 'fs';
const PORT = 3000;

const httpsOptions = {
    key: fs.readFileSync('./config/key.pem'),
    cert: fs.readFileSync('./config/cert.pem')
}

https.createServer(httpsOptions, app).listen(PORT, () => {
    console.log('Express server listening on port ' + PORT);
})
```

From now on, our application will always run over HTTPS.

Getting data over HTTPS (Postman)

You will get no response and an error if trying to access over HTTP

Method 2: Using secret key for authentication

```
"console.log(require('crypto').randomBytes(20).toString('hex'))"
```

Now, we will use middleware to check for the key before responding to a request. For example, if you want to get all contacts, you need to pass a key.

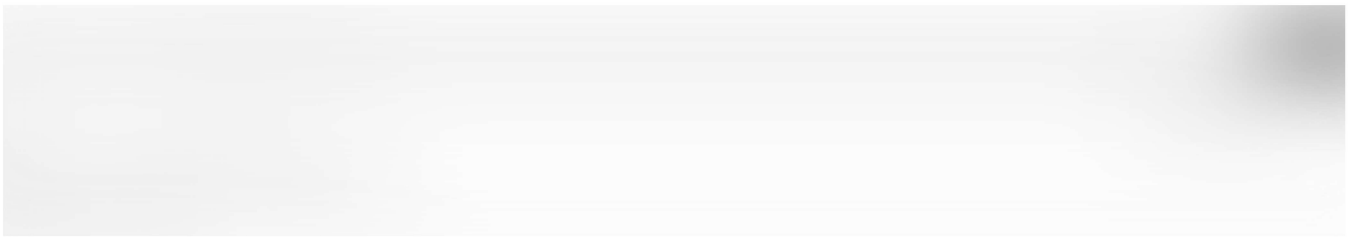
```
// GET request
https://127.0.0.1:3000?key=78942ef2c1c98bf10fca09c808d718fa3734703e
```

We will edit the **/lib/routes/crmRouters.ts** before sending the request. *Remember that, in production, you should pass the key in the environment, not directly like in the example.*

```
// lib/routes/crmRouters.ts
// get all contacts

app.route('/contact')
.get((req: Request, res: Response, next: NextFunction) => {
 // middleware
 if(req.query.key !== '78942ef2c1c98bf10fca09c808d718fa3734703e'){
  res.status(401).send('You shall not pass!');
 } else {
  next();
 }
}, this.contactController.getContacts)
```

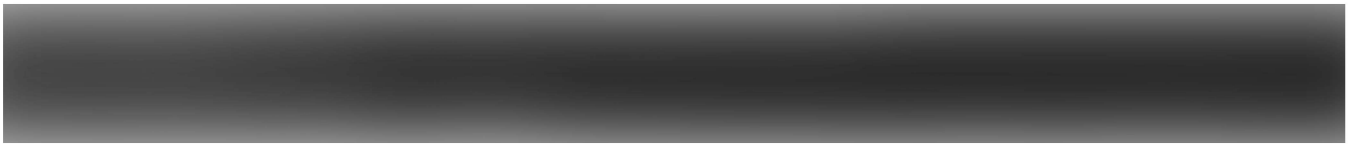
We are allowed to get the data with key



You cannot access without a key

Method 3: Secure your MongoDB

It's sad that by default, there is no security for MongoDB like at all. If you want to check your current configuration. Go to your mongo installation directory and type mongo.



As you can see, there is no Access control for the database and anyone can do anything with the database. So we will enable authentication feature for MongoDB.

First, we need to create an account in order to authenticate with Mongodb.

After that, we will stop and restart MongoDB with authentication. Remember to check your dbpath.

Now, if we login to the mongo shell, there is no warning about access control.

Or you can connect to the mongo shell with username and password you just created.

```
mongo --port 27017 -u dalenguyen -p 123123  --authenticationDatabase
CRMdb
```

Now, if we try to access the database even with the key, we are not able to.

That's why we need to edit the mongodb URL in order for the app to work. *Again, you should put the mongodb URI to the environment.*

```
// lib/app.ts

class App {
...

public mongoUrl: string =
'mongodb://dalenguyen:123123@localhost:27017/CRMdb';
```

After this, now we have a fully secure and working RESTful Web APIs application with TypeScript and Nodejs. If you want to check all the code, please visit my github repository for the full code.

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

Nodejs Typescript Rest Api Mongodb Expressjs