Brunno Vodola

21 Followers · About Follow

CRUD routes generator with Node + Express.js + Mongoose

Brunno Vodola Aug 9, 2018 · 3 min read

That's right. We will create a reusable CRUD, so that you never have to write that s**t again! Keep on reading. =D

Pre-requisites

- Node.js 6+
- Yarn or NPM (This tutorial uses Yarn)

Starting our project

Here's the complete and commented code for this project.

Whenever I'm following a tutorial, I usually don't like when I have to clone some codebase as the starting point. I have this impulse to build from the ground up so I know how everything works. Of course, that's not always possible or practical. But that's what we are doing today.

Navigate to your projects folder and run the following:

`mkdir crud-proj`

`cd crud-proj`

`yarn init -y` or `npm init -y`

We should now have our `package.json` file ready.

Adding express

`yarn add express` or `npm install --save express`

Starting the server

Open your project in your favorite IDE. Inside your project, in its root folder, create a new file called `index.js`. Then add the following code to it:

This shoudn't be new to you. We've just written the simplest node server on Earth. In the terminal, go to the root folder of the project and run: `node index.js` just to check that everything's fine. You should see:

```
$ listening on port 4000
```

Creating our Mongoose model

On your terminal, run:

`yarn add mongoose` or `npm install --save mongoose`

I chose mongoose for this tutorial. But you can, of course, choose any package you want to deal with MongoDB.

Now we are going to create our model. For this tutorial, we will be creating a collection of jokes. Oh, and as the MongoDB, feel free to use my mlab.com connection string, as it was created just for tutorials.

Create a file called `models.js`:

Now, add `const models = require('./models');` to your `index.js` file.

Adding body-parser

Run `yarn add body-parser` `npm install --save body-parser`

Now, we are adding the body parser to our middleware, so we can parse JSON and URL-Encoded requests from our clients. Your `index.js` should look like this now:

Creating the reusable CRUD

Finally! Now, it's time to create our reusable CRUD code.

Take a look and try to understand the complete code and we will discuss it below:

So we have the following here:

On line 5, we have `module.exports = (Collection) =>`

That is the key that will allow us to pass any collection we want and generate the crud for it. Basically, we are exporting a function that takes `Colection` as argument and returns the CRUD routes for it, as we will see later.

On lines 10, 25, 41and 57, we have the `create()` `readOne(), readMany()` `update()` and `remove()` functions. They are middleware functions, which in this case means that:

- they take those arguments `req`and `res`
- they send a response to the client, using either `res.send()` or `res.sendStatus()`

Each of these functions is obviously responsible for one type of database interaction and that should be pretty straightforward. Checkout the Mongoose docs on queries if you have any doubt.

On lines 85–89, we define our routes with their parameters and middlewares to handle the request.

Finally, on line 91, we return the **express router** object with the CRUD routes for our model.

Putting our reusable CRUD to use

All you have to do now is add this line to your `index.js`

app.use(‘/api/jokes’, require(‘./crud’)(models.Jokes));

And bam! CRUD routes for our jokes model!

I think you already got it by now, but to generate CRUD routes for a different Collection, all you have to do is:

app.use(‘/api/jokes’, require(‘./crud’)(models.CollectionName));

Where CollectionName is the name you chose.

That's it! Please, share this if you found it useful. Also, If you have any question, ask below or email me at bvodola@gmail.com. Happy coding and many claps for all of us!

308 5

JavaScript Nodejs Expressjs Mongoose

More from Brunno Vodola

Follow

Full Stack Developer

More From Medium

Top 10 Most Popular JavaScript Libraries to Use in 2021

Md Kamaruzzaman in JavaScript In Plain English

Simple Pagination with Node.js, Mongoose, and Express

Kevin Luu in JavaScript In Plain English

Restful API with NodeJS, Express, PostgreSQL, Sequelize, Travis, Mocha, Coveralls and Code Climate.

victor steven

Utilizing Square’s Delayed Capture: Node.js, Express, MongoDB

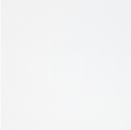
Eric Ngo

Storing User Sessions on the Server with Express-Session

John Au-Yeung in JavaScript In Plain English

Basics of Authentication using Passport and JWT with Sequelize and MySQL Database

Mustapha Issa Toyin in DevC Kano



Implementing Authentication in Node.js With Express and JWT

Deepak Kumar in Better Programming

Is Deno Already Dead?

Louis Petrik in JavaScript In Plain English

[redacted]