**Understanding Routing In Express.js As Simple As Possible -** [Total nerd](https://medium.com/@devdotpy?source=post_page-----45b4a852c4a0--------------------------------) Jun 18, 2022

Let’s learn how to organize routes in Express in a functional and efficient way so we can extend our backend without losing our min

As we all know, the purpose of using Express.js is to build an api that can respond to upcoming requests, so we have *endpoints* AKA routes, and we need to come up with the logic to handle the upcoming requests to them

But how do we organize our files and folders in an elegant and organized way so it’s easier to both debug and maintain?

Routing is extremely crucial to a web application, it’s the most important part because if we can’t build a proper api to manage the requests and be able to know the pieces to the puzzle, we’ll end up with an atrocity of a code base that no dev want to look at

**IMAGINE A BLOG**

For the sake of demonstration, we’re going to build a dummy REST API for a blog web application, so we’re going to have routes for:

1. Users: routes to get all users, user by id, add a user, update, and delete a user
2. Articles: routes to do the above but for articles

Data is going to be just objects in an array so nothing fancy it’s only for demo purposes and trying to simulate a real-life situation where routing could get really complex

**EXPRESS ROUTER**

Firstly let’s explore another powerful tool of Express called express.router() so what does it do and how is it different from a traditional express app that handles routes?

Router helps us write route handlers for “parts” of our app, so think of a router for users, and another one for the articles, maybe we have a router for handling payments? Maybe for groups etc.

Once you’ve written your routes, you can then “import” them into your entry file AKA index.js or whatever you named it to use them.

Here is a quick example of how a router looks:

Text

Description automatically generated

This way we can arrange our routes for each part of the backend separately and we’ll see how we can use these routes in our main file, so our main file is going to look way simpler and more organized.

**IMPLEMENTING THE ROUTES**

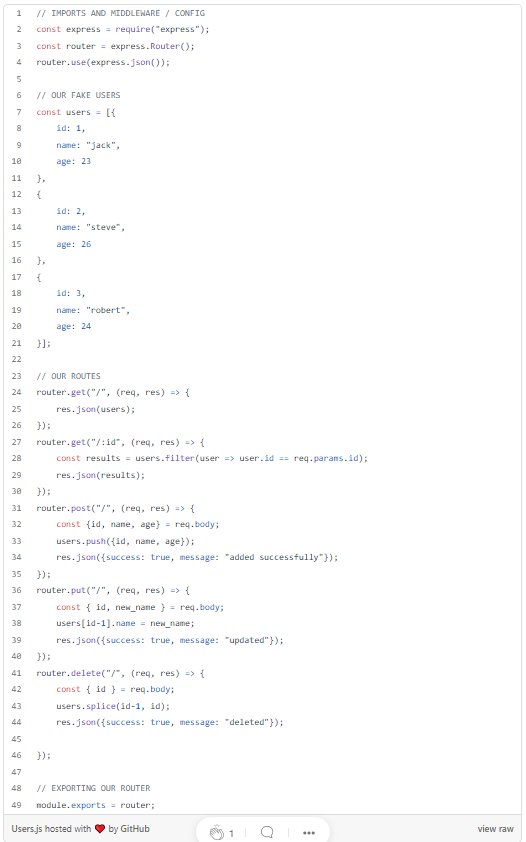
Now let’s create our routes for users and articles AKA our routers for the users and articles.

Make a new folder and name it routes with the command mkdir routes and navigate into it cd routes so your files structure should look like this:

Text

Description automatically generated

First let’s start with the users router which is a very simple CRUD API and the articles router will be fairly similar.



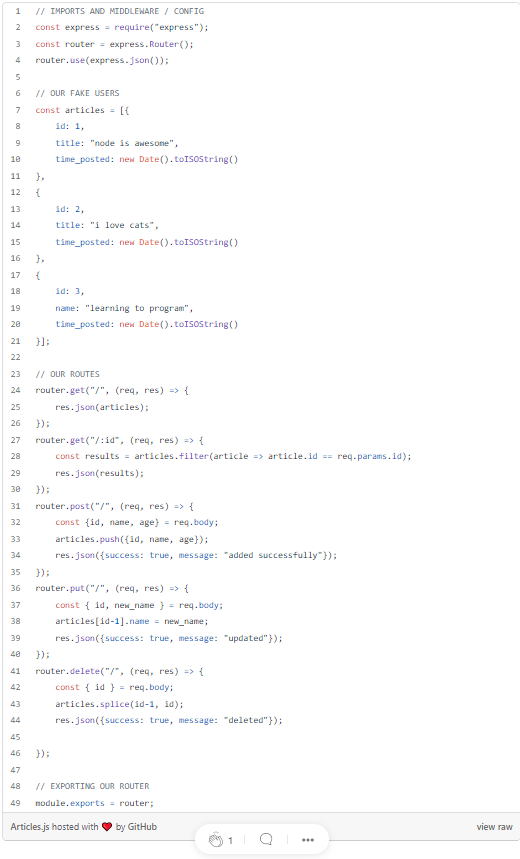
The code is super simple if you’ve read my [previous article on express](https://devdotpy.medium.com/learning-the-pern-stack-express-js-b30ba95c38a6) but let’s briefly break it down.

We begin by importing our required packages which is just express and then we initialize our router.

We create an array of objects to represent data we supposedly have stored in our imaginary database.

We then implement basic routes for the users AKA the array elements, finally we use module.exports to export our router so we can import and use it in any other file.

Now let’s implement some article routes so we next learn how to actually use those routes in our main file.



**IMPORTING OUR ROUTES INTO INDEX.JS**

Now that we’ve constructed routes for our users and articles endpoints, we need to use them in our server or if i may say “register” those routes, and it’s quite simple.

To use these routes in our index.js we simply import the routers and assign them to an endpoint prefix like the following:

const users = require("./routes/Users");  
const articles = require("./routes/Articles");  
app.use("/users", users);  
app.use("/articles", articles);

This means that every request to an endpoint that starts with /users or /articles they’ll be directed to the respective routers.

E.g., http://localhost:3000/users/ , <http://localhost:3000/users/2> <http://localhost:3000/articles/1>

So if we had maybe a shop and we finished a nice router we could use app.use("/shop", shop); so our server knows if it sees a request to anything that starts with /shop it’ll send it to the shop router.

**CONCLUSION**

express.router() is such a useful tool to help us organize our routes in an express application to help us stay being able to maintain the project as it scales more, organizing your routes and middleware files in their respective directories is also crucial so we separate concerns as much as possible and maintain an organized codebase that is easy to work with for both us and whoever will be in charge of the project.