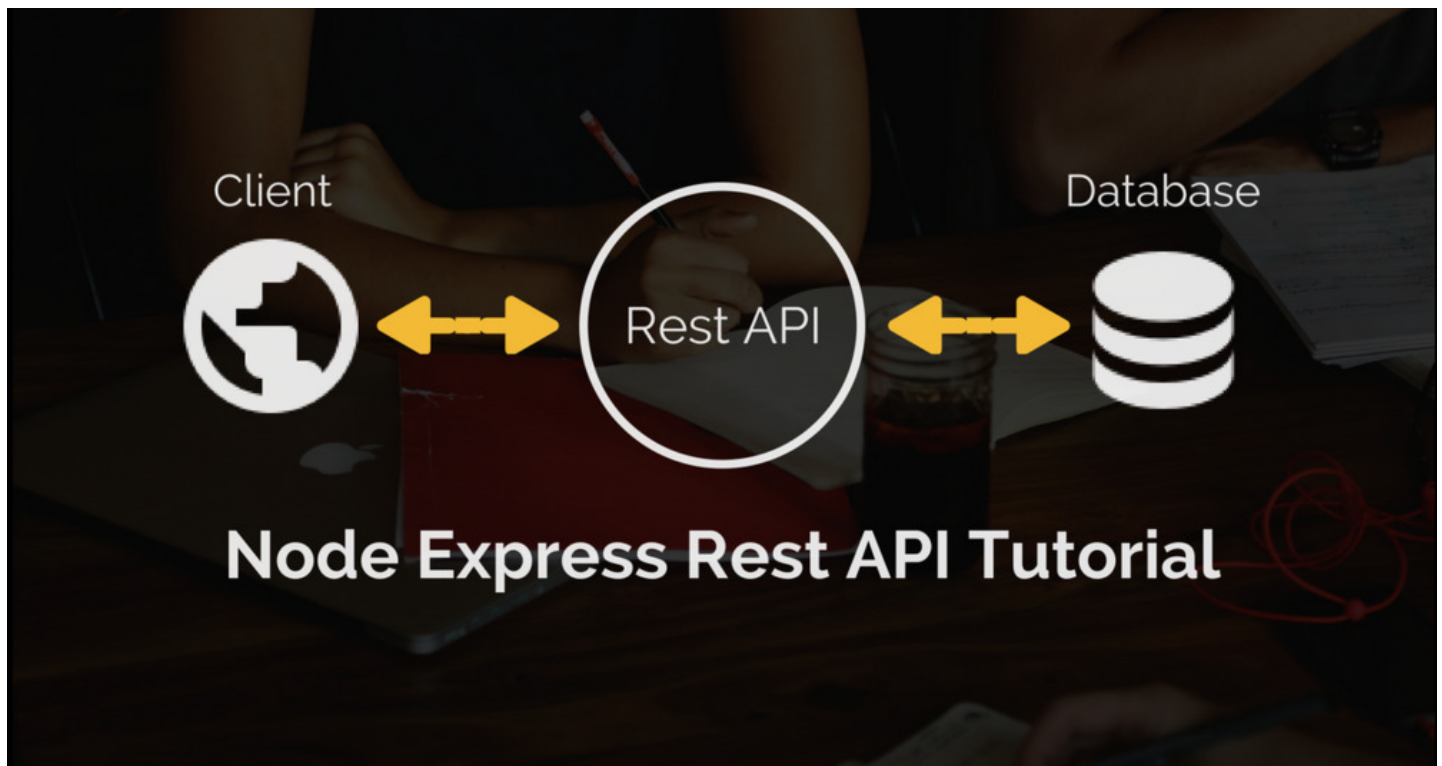


# Building a Restful CRUD API with Node.js, Express and MongoDB



Rajeev Kumar Singh • Node.js • Jun 13, 2017 • 10 mins read



In this tutorial, we'll be building a RESTful CRUD (Create, Retrieve, Update, Delete) API with Node.js, Express and MongoDB. We'll use Mongoose for interacting with the MongoDB instance.

**Express** is one of the most popular web frameworks for node.js. It is built on top of node.js http module, and adds support for routing, middleware, view system etc. It is very simple and minimal, unlike other frameworks that try to do way too much, thereby reducing the flexibility for developers to have their own design choices.

**Mongoose** is an ODM (Object Document Mapping) tool for Node.js and MongoDB. It helps you convert the objects in your code to documents in the database and vice versa.

Before proceeding to the next section, Please install MongoDB in your machine if you have not done already. Checkout the [official MongoDB installation manual](#) for any help with the installation.



## Our Application

In this tutorial, We will be building a simple Note-Taking application. We will build Rest APIs for creating, listing, editing and deleting a Note.

We'll start by building a simple web server and then move on to configuring the database, building the `Note` model and different routes for handling all the CRUD operations.

Finally, we'll test our REST APIs using Postman.

Also, In this post, we'll heavily use ES6 features like `let`, `const`, `arrow functions`, `promises` etc. It's good to familiarize yourself with these features. I recommend [this re-introduction to Javascript](#) to brush up these concepts.

Well! Now that we know what we are going to build, We need a cool name for our application. Let's call our application `EasyNotes`.

## Creating the Application

1. Fire up your terminal and create a new folder for the application.

```
$ mkdir node-easy-notes-app
```

2. Initialize the application with a `package.json` file

Go to the root folder of your application and type `npm init` to initialize your app with a `package.json` file.

```
$ cd node-easy-notes-app
$ npm init
```

```
name: (node-easy-notes-app)
version: (1.0.0)
```



```
test command:
git repository:
keywords: Express RestAPI MongoDB Mongoose Notes
author: callicoder
license: (ISC) MIT
About to write to /Users/rajeevkumarsingh/node-easy-notes-app/package.json:

{
  "name": "node-easy-notes-app",
  "version": "1.0.0",
  "description": "Never miss a thing in Life. Take notes quickly. Organize and keep track of a
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Express",
    "RestAPI",
    "MongoDB",
    "Mongoose",
    "Notes"
  ],
  "author": "callicoder",
  "license": "MIT"
}

Is this ok? (yes) yes
```

Note that I've specified a file named `server.js` as the entry point of our application. We'll create `server.js` file in the next section.

### 3. Install dependencies

We will need express, mongoose and body-parser modules in our application. Let's install them by typing the following command -



I've used `--save` option to save all the dependencies in the `package.json` file. The final `package.json` file looks like this -

```
{
  "name": "node-easy-notes-app",
  "version": "1.0.0",
  "description": "Never miss a thing in Life. Take notes quickly. Organize and keep track of a",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Express",
    "RestAPI",
    "MongoDB",
    "Mongoose",
    "Notes"
  ],
  "author": "callicoder",
  "license": "MIT",
  "dependencies": {
    "body-parser": "^1.18.2",
    "express": "^4.16.3",
    "mongoose": "^5.0.11"
  }
}
```

Our application folder now has a `package.json` file and a `node_modules` folder -

```
node-easy-notes-app
├── node_modules/
└── package.json
```



Let's now create the main entry point of our application. Create a new file named `server.js` in the root folder of the application with the following contents -

```
const express = require('express');
const bodyParser = require('body-parser');

// create express app
const app = express();

// parse requests of content-type - application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: true }));

// parse requests of content-type - application/json
app.use(bodyParser.json());

// define a simple route
app.get('/', (req, res) => {
  res.json({ "message": "Welcome to EasyNotes application. Take notes quickly. Organize and k
});

// listen for requests
app.listen(3000, () => {
  console.log("Server is listening on port 3000");
});
```

**First,** We import express and body-parser modules. [Express](#), as you know, is a web framework that we'll be using for building the REST APIs, and [body-parser](#) is a module that parses the request (of various content types) and creates a `req.body` object that we can access in our routes.

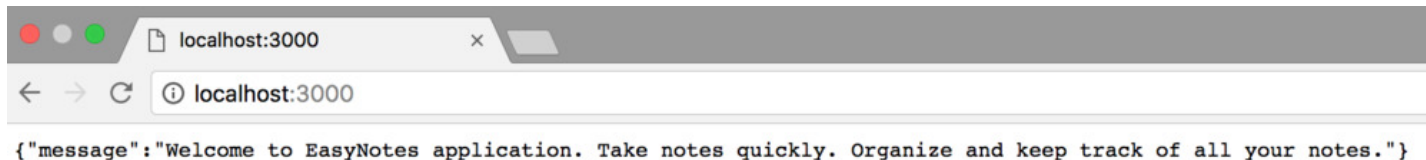
**Then,** We create an express app, and add two `body-parser` middlewares using express's `app.use()` method. A [middleware](#) is a function that has access to the `request` and `response` objects. It can execute any code, transform the request object, or return a response.

**Then,** We define a simple `GET` route which returns a welcome message to the clients.

**Finally,** We listen on port 3000 for incoming connections.



```
$ node server.js  
Server is listening on port 3000
```



## Configuring and Connecting to the database

I like to keep all the configurations for the app in a separate folder. Let's create a new folder `config` in the root folder of our application for keeping all the configurations -

```
$ mkdir config  
$ cd config
```

Now, Create a new file `database.config.js` inside `config` folder with the following contents -

```
module.exports = {  
  url: 'mongodb://localhost:27017/easy-notes'  
}
```

We'll now import the above database configuration in `server.js` and connect to the database using mongoose.

Add the following code to the `server.js` file after `app.use(bodyParser.json())` line -

```
// Configuring the database  
const dbConfig = require('./config/database.config.js');  
const mongoose = require('mongoose');
```



```
// Connecting to the database
mongoose.connect(dbConfig.url)
.then(() => {
  console.log("Successfully connected to the database");
}).catch(err => {
  console.log('Could not connect to the database. Exiting now...');
  process.exit();
});
```

Please run the server and make sure that you're able to connect to the database -

```
$ node server.js
Server is listening on port 3000
Successfully connected to the database
```



## Defining the Note model in Mongoose

Next, We will define the `Note` model. Create a new folder called `app` inside the root folder of the application, then create another folder called `models` inside the `app` folder -

```
$ mkdir -p app/models
$ cd app/models
```



```
const mongoose = require('mongoose');

const NoteSchema = mongoose.Schema({
  title: String,
  content: String
}, {
  timestamps: true
});

module.exports = mongoose.model('Note', NoteSchema);
```

The `Note` model is very simple. It contains a `title` and a `content` field. I have also added a `timestamps` option to the schema.

Mongoose uses this option to automatically add two new fields - `createdAt` and `updatedAt` to the schema.

## Defining Routes using Express

Next up is the routes for the Notes APIs. Create a new folder called `routes` inside the `app` folder.

```
$ mkdir app/routes
$ cd app/routes
```

Now, create a new file called `note.routes.js` inside `app/routes` folder with the following contents

```
module.exports = (app) => {
  const notes = require('../controllers/note.controller.js');

  // Create a new Note
  app.post('/notes', notes.create);

  // Retrieve all Notes
  app.get('/notes', notes.findAll);
}
```





```
app.get('/notes/:noteId', notes.findOne);

// Update a Note with noteId
app.put('/notes/:noteId', notes.update);

// Delete a Note with noteId
app.delete('/notes/:noteId', notes.delete);
}
```

Note that We have added a `require` statement for `note.controller.js` file. We'll define the controller file in the next section. The controller will contain methods for handling all the CRUD operations.

Before defining the controller, let's first include the routes in `server.js`. Add the following `require` statement before `app.listen()` line inside `server.js` file.

```
// .....

// Require Notes routes
require('./app/routes/note.routes.js')(app);

// .....
```

If you run the server now, you'll get the following error -

```
$ node server.js
module.js:472
  throw err;
  ^

Error: Cannot find module '../controllers/note.controller.js'
```

This is because we haven't defined the controller yet. Let's do that now.

## Writing the Controller functions



```
const Note = require('../models/note.model.js');

// Create and Save a new Note
exports.create = (req, res) => {

};

// Retrieve and return all notes from the database.
exports.findAll = (req, res) => {

};

// Find a single note with a noteId
exports.findOne = (req, res) => {

};

// Update a note identified by the noteId in the request
exports.update = (req, res) => {

};

// Delete a note with the specified noteId in the request
exports.delete = (req, res) => {

};
```

Let's now look at the implementation of the above controller functions one by one -

## Creating a new Note

```
// Create and Save a new Note
exports.create = (req, res) => {
  // Validate request
  if(!req.body.content) {
```



```
    });  
  }  
  
  // Create a Note  
  const note = new Note({  
    title: req.body.title || "Untitled Note",  
    content: req.body.content  
  });  
  
  // Save Note in the database  
  
  note.save()  
    .then(data => {  
      res.send(data);  
    }).catch(err => {  
      res.status(500).send({  
        message: err.message || "Some error occurred while creating the Note."  
      });  
    });  
};
```

## Retrieving all Notes

```
// Retrieve and return all notes from the database.  
exports.findAll = (req, res) => {  
  Note.find()  
    .then(notes => {  
      res.send(notes);  
    }).catch(err => {  
      res.status(500).send({  
        message: err.message || "Some error occurred while retrieving notes."  
      });  
    });  
};
```



```
// Find a single note with a noteId
exports.findOne = (req, res) => {
  Note.findById(req.params.noteId)
    .then(note => {
      if(!note) {
        return res.status(404).send({
          message: "Note not found with id " + req.params.noteId
        });
      }
      res.send(note);
    }).catch(err => {
      if(err.kind === 'ObjectId') {
        return res.status(404).send({
          message: "Note not found with id " + req.params.noteId
        });
      }
      return res.status(500).send({
        message: "Error retrieving note with id " + req.params.noteId
      });
    });
};
```

## Updating a Note

```
// Update a note identified by the noteId in the request
exports.update = (req, res) => {
  // Validate Request
  if(!req.body.content) {
    return res.status(400).send({
      message: "Note content can not be empty"
    });
  }

  // Find note and update it with the request body
  Note.findByIdAndUpdate(req.params.noteId, {
```



```
    }, {new: true})
    .then(note => {
      if(!note) {
        return res.status(404).send({
          message: "Note not found with id " + req.params.noteId
        });
      }
      res.send(note);
    }).catch(err => {
      if(err.kind === 'ObjectId') {
        return res.status(404).send({
          message: "Note not found with id " + req.params.noteId
        });
      }
      return res.status(500).send({
        message: "Error updating note with id " + req.params.noteId
      });
    });
  });
};
```

The `{new: true}` option in the `findByIdAndUpdate()` method is used to return the modified document to the `then()` function instead of the original.

## Deleting a Note

```
// Delete a note with the specified noteId in the request
exports.delete = (req, res) => {
  Note.findByIdAndRemove(req.params.noteId)
    .then(note => {
      if(!note) {
        return res.status(404).send({
          message: "Note not found with id " + req.params.noteId
        });
      }
      res.send({message: "Note deleted successfully!"});
    }).catch(err => {
```



```
        message: "Note not found with id " + req.params.noteId
    });
}
return res.status(500).send({
    message: "Could not delete note with id " + req.params.noteId
});
});
};
```

You can check out the documentation of all the methods that we used in the above APIs on Mongoose's official documentation -

- [Mongoose save\(\)](#)
- [Mongoose find\(\)](#)
- [Mongoose findById\(\)](#)
- [Mongoose findByIdAndUpdate\(\)](#)
- [Mongoose findByIdAndRemove\(\)](#)

## Testing our APIs

Let's now test all the APIs one by one using postman.

Creating a new Note using `POST /notes` API



POST http://localhost:3000/notes

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 [{"title": "My First Note", "content": "This is my first note in EasyNotes application"}]
```

Body Cookies Headers (6) Tests

Status: 200 OK Time: 24 ms

Pretty Raw Preview JSON

```
1 {
2   "_v": 0,
3   "updatedAt": "2017-05-26T12:23:32.340Z",
4   "createdAt": "2017-05-26T12:23:32.340Z",
5   "title": "My First Note",
6   "content": "This is my first note in EasyNotes application",
7   "_id": "59281e4447f72acb22fac8f8"
8 }
```

## Retrieving all Notes using GET /notes API

http://localhost:3000/ No Environment

GET http://localhost:3000/notes Params Send Save

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (6) Tests

Status: 200 OK Time: 27 ms

Pretty Raw Preview JSON

```
1 [
2   {
3     "_id": "59281e4447f72acb22fac8f8",
4     "updatedAt": "2017-05-26T12:23:32.340Z",
5     "createdAt": "2017-05-26T12:23:32.340Z",
6     "title": "My First Note",
7     "content": "This is my first note in EasyNotes application",
8     "_v": 0
9   },
10  {
11    "_id": "59281e9e47f72acb22fac8f9",
12    "updatedAt": "2017-05-26T12:25:02.179Z",
13    "createdAt": "2017-05-26T12:25:02.179Z",
14    "title": "My Second Note",
15    "content": "Creating my second note. I love EasyNotes.",
16    "_v": 0
17  }
18 ]
```

Search CalliCoder

Java Kotlin Golang Spring Boot Node.js JavaFX About



http://localhost:3000/ x +

GET http://localhost:3000/notes/59281e9e47f72acb22fac8f9 Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

Type No Auth

Body Cookies Headers (6) Tests Status: 200 OK Time: 37 ms

Pretty Raw Preview JSON

```
1 {
2   "_id": "59281e9e47f72acb22fac8f9",
3   "updatedAt": "2017-05-26T12:25:02.179Z",
4   "createdAt": "2017-05-26T12:25:02.179Z",
5   "title": "My Second Note",
6   "content": "Creating my second note. I love EasyNotes.",
7   "__v": 0
8 }
```

## Updating a Note using PUT /notes/:noteId API

http://localhost:3000/ x + No Environment

PUT http://localhost:3000/notes/59281e9e47f72acb22fac8f9 Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {"title": "Edited Title of Second Note", "content": "Edited Content of Second Note."}
```

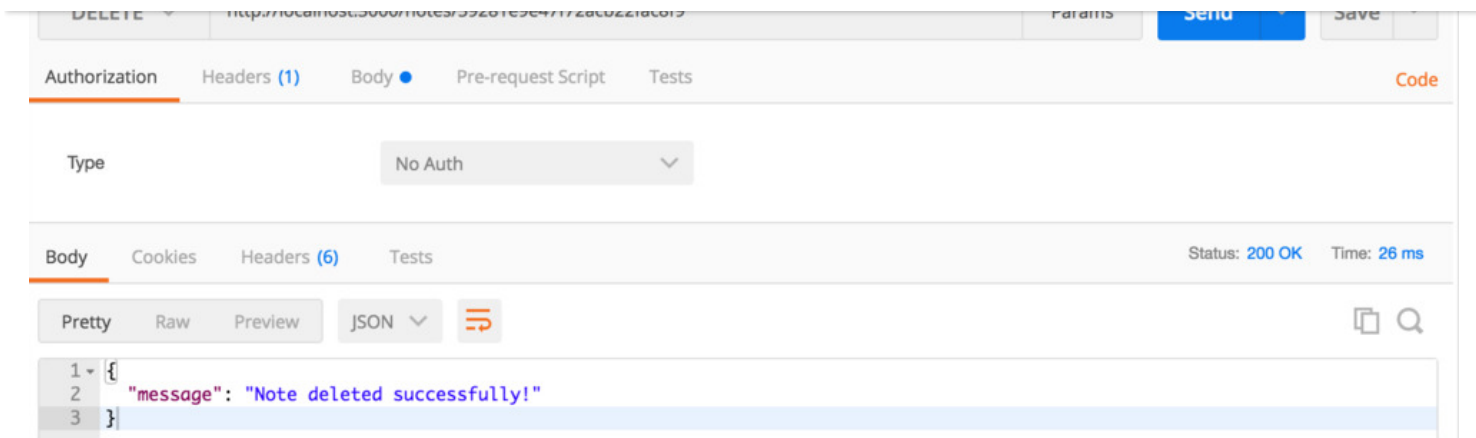
Body Cookies Headers (6) Tests Status: 200 OK Time: 61 ms

Pretty Raw Preview JSON

```
1 {
2   "_id": "59281e9e47f72acb22fac8f9",
3   "updatedAt": "2017-05-26T12:26:38.486Z",
4   "createdAt": "2017-05-26T12:25:02.179Z",
5   "title": "Edited Title of Second Note",
6   "content": "Edited Content of Second Note.",
7   "__v": 0
8 }
```

## Deleting a Note using DELETE /notes/:noteId API






## Conclusion

In this tutorial, We learned how to build rest apis in node.js using express framework and mongodb.


You can find the code for this tutorial in [my github repository](#). Please ask any questions that you might have in the comment section below.

Thanks for reading. See you in the next tutorial!



### Chrome for Enterprise

Keep Your Organization's Data Safe & Secure. Deploy Chrome Browser Today!



Liked the Article? Share it on Social media!


[Twitter](#)[Facebook](#)[Google+](#)[Linkedin](#)[Reddit](#)

65 Comments

CalliCoder

 Login

 Recommend 10

 Share

Sort by Newest



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name



Hi,

Your article is very nice and helpful. I'm using node.js with SQL database. All are working fine and now I want to deploy on IIS, I went through many articles but doesn't work for me. Please help me.

Thanks in advance !!

Meena Damwani

^ | v • Reply • Share ›


This comment is awaiting moderation. [Show comment.](#)

 **Shahbaz Mancho** → Shahbaz Mancho • 2 months ago

here is config

url: 'mongodb://localhost:27017/easy-notes'

^ | v • Reply • Share ›

 **Rajeev Kumar Singh** Mod → Shahbaz Mancho • 2 months ago

Hi,

Can you check if MongoDB running in your system?

^ | v • Reply • Share ›

 **Shahbaz Mancho** → Rajeev Kumar Singh • 2 months ago

Hi Kumar

yeah I just checked MongoDB, I made a stupid mistake, I thought mongo will run itself :P

^ | v • Reply • Share ›

 **Rajeev Kumar Singh** Mod → Shahbaz Mancho • 2 months ago

Cool. I've also updated the post, and removed `useMongoClient` from `mongoose.connect()` function. This will remove the warning that you're getting.

^ | v • Reply • Share ›

 **Shahbaz Mancho** → Rajeev Kumar Singh • 2 months ago

thankyou. yeah I see that warning and removed `useMongoClient`

^ | v • Reply • Share ›

 **LobsterYu** • 11 days ago

Thanks for sharing, this helped me a lot :)

^ | v • Reply • Share ›

 **Md. Amanul Huque Ony** • 17 days ago

Thanks for sharing this awesome tutorial. :)

^ | v • Reply • Share ›

 **harsha kushal** • 18 days ago

Thanks Rajeev, but now I am facing Schema not defined problem.

^ | v • Reply • Share ›

 **Rajeev Kumar Singh** Mod → harsha kushal • 18 days ago

Can you post the error stacktrace here?

^ | v • Reply • Share ›



^ | v • Reply • Share ›



**Rajeev Kumar Singh** Mod → harsha kushal • 18 days ago

Hey Harsha,

Please check that MongoDB running in your system and you're able to connect to it using `mongo` client.

^ | v • Reply • Share ›



**sheheryar nisar** • a month ago

Note does not get deleted, its just showing message "Note Deleted successfully"

After that when I get Notes, its showing me all notes included one I just deleted.

^ | v • Reply • Share ›



**Rajeev Kumar Singh** Mod → sheheryar nisar • a month ago

Hi,

I can't reproduce what you said. The delete API is working fine for me. Can you provide more details, like your Node version, Mongoose version. And can you cross check your delete API with the one in this article.

Regards,

Rajeev

^ | v • Reply • Share ›



**Monika** • a month ago

Hi

I m getting the below error.Please help me to solve this error

TypeError: Cannot read property 'content' of undefined

at exports.create.note.save.then.data.catch (D:\npm\app\controller\noteController.js:4:16)

at Layer.handle [as handle\_request] (D:\npm\node\_modules\express\lib\router\layer.js:95:5)

at next (D:\npm\node\_modules\express\lib\router\route.js:137:13)

at Route.dispatch (D:\npm\node\_modules\express\lib\router\route.js:112:3)

at Layer.handle [as handle\_request] (D:\npm\node\_modules\express\lib\router\layer.js:95:5)

at D:\npm\node\_modules\express\lib\router\index.js:281:22

at Function.process\_params (D:\npm\node\_modules\express\lib\router\index.js:335:12)

at next (D:\npm\node\_modules\express\lib\router\index.js:275:10)

at expressInit (D:\npm\node\_modules\express\lib\middleware\init.js:40:5)

at Layer.handle [as handle\_request] (D:\npm\node\_modules\express\lib\router\layer.js:95:5)

at trim\_prefix (D:\npm\node\_modules\express\lib\router\index.js:317:13)

at D:\npm\node\_modules\express\lib\router\index.js:284:7

at Function.process\_params (D:\npm\node\_modules\express\lib\router\index.js:335:12)

at next (D:\npm\node\_modules\express\lib\router\index.js:275:10)

at query (D:\npm\node\_modules\express\lib\middleware\query.js:45:5)

at Layer.handle [as handle\_request] (D:\npm\node\_modules\express\lib\router\layer.js:95:5)

^ | v • Reply • Share ›



**Rajeev Kumar Singh** Mod → Monika • a month ago

That error means that `req.body` is undefined in the `exports.create` method. Can you check the routes and verify that this method is being referred like this -

```
module.exports = (app) => {  
  const notes = require('../controllers/note.controller.js');  
  app.post('/notes', notes.create);  
}
```



^ | v • Reply • Share ›



**praneeth teja** • a month ago

`require('./app/routes/note.routes.js')(app);` .Hi everyone. I have a doubt. Should we not store this module in any var. Its giving error by writing this way.

^ | v • Reply • Share ›



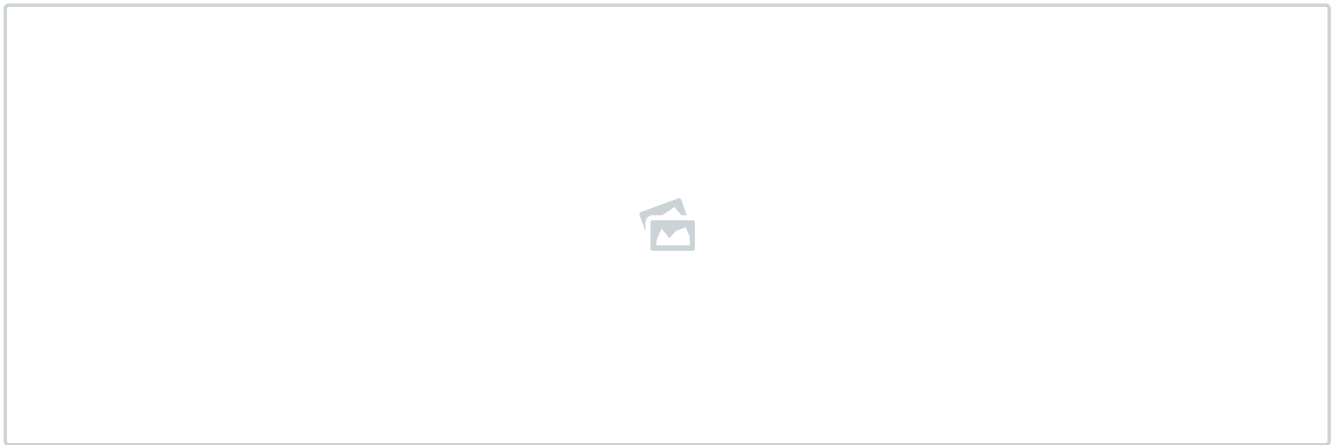
**Rajeev Kumar Singh** Mod ➔ praneeth teja • a month ago

What error are you getting? We can store the module in a `var`, but there is not point storing it when we're not gonna use that `var`.

^ | v • Reply • Share ›



**Akshaya Saravanan** • a month ago



Whenever I create a note and send it to the server through postman, I get an error saying the message content cannot be empty. I have attached the screenshots of the output. Please let me know the error.

^ | v • Reply • Share ›



**Rajeev Kumar Singh** Mod ➔ Akshaya Saravanan • a month ago

Request Content-Type should be `application/json`. Click on the `Text` drop down in Postman and select `JSON`. It should work after that.

^ | v • Reply • Share ›



**Winlight Solutions** • a month ago

The best and simplest tutorial for node, express mongo with ReSTful APIs! Wish that you could add " Create Angular Frontend" tutorial as a second part for this best tutorial and get "MEAN" completed. Thanks a million!

1 ^ | v • Reply • Share ›



**Srinivasarao chintala** • a month ago

Excellent tutorial, nicely explained. Thank you!!!!

^ | v • Reply • Share ›



**Seorang Kapiten** • 2 months ago

I get this error after configuring and connecting database section.

```
module.js:549
throw err;
^
```

Error: Cannot find module '/home/putrapc/webproject/node-easy-notes-app/config/server.js'



at startup (bootstrap\_node.js:188:16)  
at bootstrap\_node.js:609:3

how it could be?

^ | v • Reply • Share ›



**Seorang Kapiten** → Seorang Kapiten • 2 months ago

Sorry my fault its solved now

^ | v • Reply • Share ›



**Robert** • 2 months ago

Hi. Great tutorial. Any ideas why my update function creates a new object instead of updating one?

```
// Find note and update it with request body
Note.findByIdAndUpdate(
  req.params.noteId,
  {
    title: req.body.title || "Untitled Note",
    content: req.body.content
  },
  { new: true }
)
```

^ | v • Reply • Share ›



**Robert** → Robert • 2 months ago

nevermind :) I had create function in routes instead of update

^ | v • Reply • Share ›



**carlita** • 2 months ago

Hi! I am getting "ReferenceError: Note is not defined" when trying to retrieve all (verb get with no id). Like it cannot find the model. Do you know what this could be? Thanks!

^ | v • Reply • Share ›



**carlita** → carlita • 2 months ago

solved it, sorry! A case sensitive problem xD

3 ^ | v • Reply • Share ›



**marcus nixon** • 2 months ago

okay so i finished it all now im getting this when i try to run at the end



^ | v • Reply • Share ›



**Kleurbleur** → marcus nixon • 2 months ago

Be sure to actually start Mongo as well. It's not written in the tutorial but you have to start it yourself.

^ | v • Reply • Share ›



**marcus nixon** → Kleurbleur • 2 months ago



**Olanrewaju Olayinka Ahmed** → marcus nixon • 2 months ago

Also have the same issue on my machine. I have MongoDB installed. Please what can i do to solve this. Am using windows 10 OS.

^ | v • Reply • Share ›



**marcus nixon** → Olanrewaju Olayinka Ahmed • 2 months ago

Need to keep mongodb runing in another cmd

^ | v • Reply • Share ›



**Dan Phelps** → marcus nixon • 2 months ago

how do I keep monggodb running? thanks

^ | v • Reply • Share ›



**marcus nixon** → Dan Phelps • 2 months ago

just gotta keep it going in another cmd i think . have one open running that and the other for everything else

^ | v • Reply • Share ›



**Rajeev Kumar Singh** Mod → Olanrewaju Olayinka Ahmed • 2 months ago

Hi,

Can you check if you're able to connect to mongodb using `mongo` command?

^ | v • Reply • Share ›



**Rajeev Kumar Singh** Mod → marcus nixon • 2 months ago

Hi,

Do you have MongoDB installed and running in your system?

^ | v • Reply • Share ›



**marcus nixon** → Rajeev Kumar Singh • 2 months ago

Ahh yeh that was the problem didn't keep it running

^ | v • Reply • Share ›



Avatar

This comment was deleted.



**Rajeev Kumar Singh** Mod → Guest • 2 months ago

Remove the parentheses from `database.config.js`. Here is the content of the config file -

```
module.exports = {  
  url: 'mongodb://localhost:27017/easy-notes'  
}
```

^ | v • Reply • Share ›



**marcus nixon** → Rajeev Kumar Singh • 2 months ago

yeh was my bad missed the bottom bracket

^ | v • Reply • Share ›



**Kleurbleur** • 2 months ago

I followed the tut and ran into a problem while testing at the put and delete phases. So I downloaded your source



```
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
```

Cannot POST /notes/5ac3cb6cd378474738b947fd

```
</body>
</html>
^ | v • Reply • Share ›
```



**Rajeev Kumar Singh** Mod ➔ Kleurbleur • 2 months ago

Hi,

You're using POST method for updating a note. You should use PUT:

```
PUT /notes/:noteId
```

And for deleting a note, you should use DELETE:

```
DELETE /notes/:noteId
```

^ | v • Reply • Share ›



**Kleurbleur** ➔ Rajeev Kumar Singh • 2 months ago

God... guess I'm staring too long at the screen today. Works now as well in my code. Thanks!

^ | v • Reply • Share ›



**Roby Huzwandar** • 2 months ago

```
[nodemon] starting `node index.js`
/var/www/html/API2/app/models/pengurus.model.js:3
nama: string,
^
```

```
ReferenceError: string is not defined
at Object.<anonymous> (/var/www/html/API2/app/models/pengurus.model.js:3:11)
at Module._compile (module.js:652:30)
at Object.Module._extensions..js (module.js:663:10)
at Module.load (module.js:565:32)
at tryModuleLoad (module.js:505:12)
at Function.Module._load (module.js:497:3)
at Module.require (module.js:596:17)
at require (internal/module.js:11:18)
at Object.<anonymous> (/var/www/html/API2/app/controllers/pengurus.controller.js:1:78)
at Module._compile (module.js:652:30)
at Object.Module._extensions..js (module.js:663:10)
at Module.load (module.js:565:32)
at tryModuleLoad (module.js:505:12)
```



[nodemon] app crashed - waiting for file changes before starting...

I have error

^ | v • Reply • Share ›



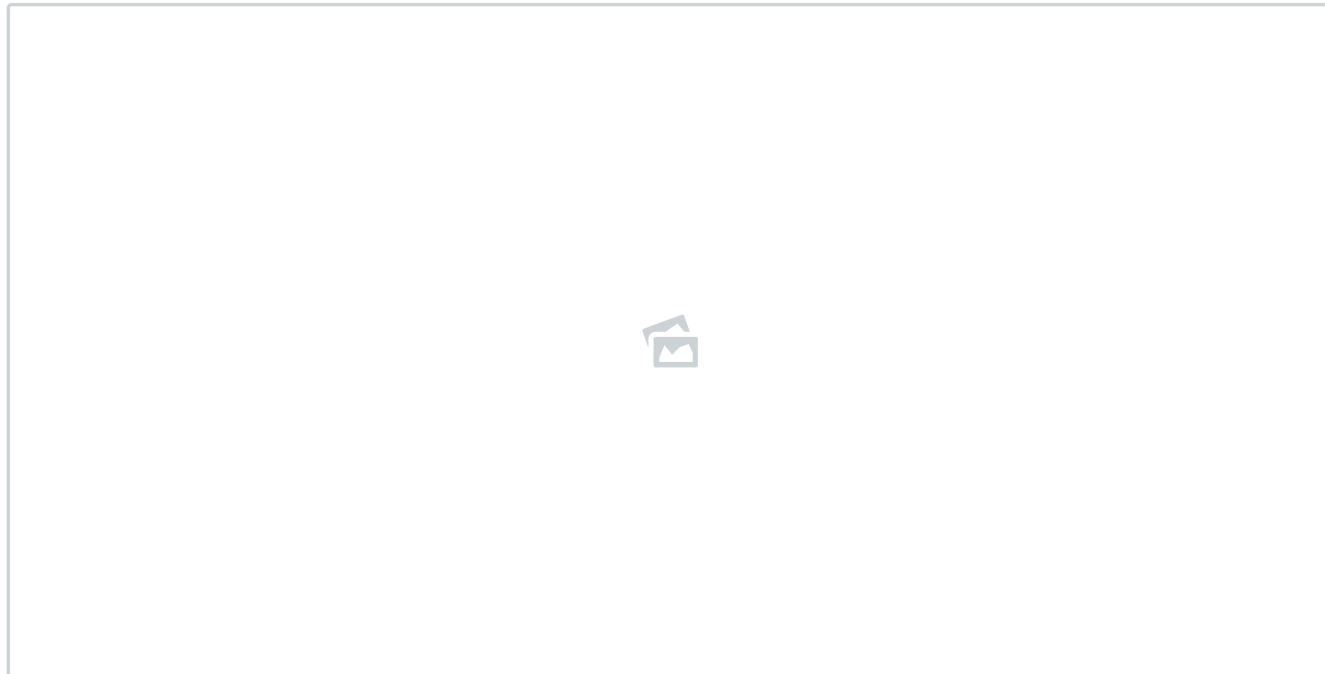
**Immanuel Kant** • 2 months ago

I have to say, this is the best tutorial I have learned to building Restful API. Thank you!

2 ^ | v • Reply • Share ›



**Nileshe** • 3 months ago



I am not getting any data after hitting my URL. Also its not throwing error.

Could you please elaborate do I need to create collection in MongoDB.

^ | v • Reply • Share ›



**Loïs Botta** • 3 months ago

Thank you for this tuto, it was very instructive and easy !

^ | v • Reply • Share ›



**prabha** • 3 months ago

Superb! clearly explained

^ | v • Reply • Share ›

[Load more comments](#)

#### ALSO ON CALLICODER

### Hibernate Many to Many Mapping Example with Spring Boot and JPA

11 comments • 6 months ago



**Raieev Kumar Singh** — Hi .Iosenh To add extra columns to

<https://www.calliCoder.com/node-js-express-mongodb-restful-crud-api-tutorial/>

### Full Stack App with Spring Boot, Spring Security, JWT, MySQL and React - Part 4

16 comments • 3 months ago



**Raieev Kumar Singh** — Hi Follow these steps to use





## Java ArrayList: The Complete Reference

2 comments • a month ago



**Rajeev Kumar Singh** — Thanks man! I'm glad that you found the article helpful.

## Kotlin Inheritance, Overriding Methods, and Properties

2 comments • 6 months ago



**Rajeev Kumar Singh** — Yes! Thanks for pointing it out. I've corrected the code now.

[Subscribe](#) [Add Disqus to your site](#) [Add Disqus](#) [Add Disqus' Privacy Policy](#) [Privacy Policy](#) [Privacy Policy](#)



[About](#) [Privacy](#) [Sitemap](#)



Copyright © CalliCoder 2017