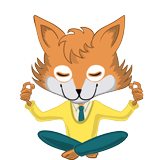
**How to Create REST API with Node, Express, and Mongoose**

[](https://hostadvice.com/author/sophija/)

* Node.js, Express.js, Mongoose.js and MongoDB are the great tools for building easy and fast REST API.
* Node.js is a packaged compilation of Google’s V8 JavaScript engine.
* Express.js is javascript web server that has complete function of web development including REST API.

Before we begin, are mandatory to be installed on your device

* [Node.js](https://hostadvice.com/hosting-services/nodejs/)
* [MongoDB](https://hostadvice.com/hosting-services/mongodb/)
* Editor or IDE

**Here we are using Terminal and Atom as the text editor. For Windows user, you can work using Node.js command line.**

**Create Express.js Project and Install Required Modules**

Lunch your terminal or node command line, go to your projects folder.

Then, using this command install express generator.

$ sudo npm install express-generator -g

Create an Express.js app using this command.

$ express node-rest-api -e

Next, go to newly created project folder then install node modules.

$ cd node-rest-api && npm install

Run the server using the line below.

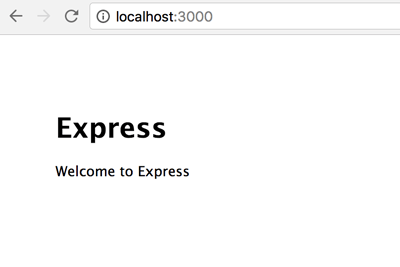
$ npm start

You will get this log in the terminal.

> node-rest-api@0.0.0 start /Users/didin/Documents/NodeApps/node-rest-api

> node ./bin/www

Go to your browser and type **localhost:3000**, Your server is ready if you get the result below



**Add Mongoose.js Module as ORM for MongoDB**

Presse CTRL+C to Stop node then type the command to install Mongoose.js module.

$ npm install mongoose --save

then type this command to start MongoDB server.

$ mongod

Now, using an IDE or a  text editor, open and edit app.js in the root of project folder.  
Declare mongoose in require sections.

var mongoose = require('mongoose');

Use mongoose with native Node Promise.

mongoose.Promise = global.Promise;

Then, Create a connection to MongoDB.

mongoose.connect('mongodb://localhost/product')

.**then**(() => console.log('connection successful'))

.**catch**((err) => console.error(err));

Run the node app using this

$ npm start

if you get this message on your terminal, the connection to MongoDB is successful.

> node-rest-api@0.0.0 start /Users/didin/Documents/NodeApps/node-rest-api

> node ./bin/www

connection successful

**Create Product Mongoose Model**

CTRL+C in the terminal to stop node server, then create models directory and javascript file as the model.

$ mkdir models

$ touch models/Product.js

In Product.js file the add the following lines of codes.

**var** mongoose = require('mongoose');

**var** ProductSchema = new mongoose.Schema({

prod\_name: String,

prod\_desc: String,

prod\_price: Number,

updated\_at: { **type**: Date, **default**: Date.now },

});

**module**.exports = mongoose.model('Product', ProductSchema);

**Create Routes for REST API endpoint**

|  |  |  |
| --- | --- | --- |
| **Method** | **Endpoints** | **Notes** |
| GET | /product | Get all products |
| GET | /product/:id | Get single product |
| POST | /product | Add product |
| PUT | /product/:id | Update product |
| DELETE | /product/:id | Delete product |

we will build the following functions for Our REST API .

First, add the javascript file to routes folder.

$ touch routes/products.js

in routes/products.js  add this lines of codes.

**var** express = require('express');

**var** router = express.Router();

**var** mongoose = require('mongoose');

**var** Product = require('../models/Product.js');

/\* GET ALL PRODUCTS \*/

router.get('/', **function**(req, res, next) {

Product.find(**function** (err, products) {

**if** (err) **return** next(err);

res.json(products);

});

});

/\* GET SINGLE PRODUCT BY ID \*/

router.get('/:id', **function**(req, res, next) {

Product.findById(req.params.id, **function** (err, post) {

**if** (err) **return** next(err);

res.json(post);

});

});

/\* SAVE PRODUCT \*/

router.post('/', **function**(req, res, next) {

Product.create(req.body, **function** (err, post) {

**if** (err) **return** next(err);

res.json(post);

});

});

/\* UPDATE PRODUCT \*/

router.put('/:id', **function**(req, res, next) {

Product.findByIdAndUpdate(req.params.id, req.body, **function** (err, post) {

**if** (err) **return** next(err);

res.json(post);

});

});

/\* DELETE PRODUCT \*/

router.delete('/:id', **function**(req, res, next) {

Product.findByIdAndRemove(req.params.id, req.body, **function** (err, post) {

**if** (err) **return** next(err);

res.json(post);

});

});

**module**.exports = router;

Next, In app.js add product route as require after users require.

**var** products = require('./routes/products');

Then add use after use of users.

app.use('/products', products);

**Test REST API Endpoints**

There are so many tools for testing REST API, we are testing using CURL from the terminal.

We start with Add/Save product data first.

$ curl -i -X POST -H "Content-Type: application/json" -d '{ "prod\_name":"HP laptop ","prod\_desc":"the new hp","prod\_price": 999 }' localhost:3000/products

If you get this response l then the product is saved successfully.

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: application/json; charset=utf-8

Content-Length: 185

ETag: W/"b9-ymiFjoNdt5wABoii1CiYbg"

Date: Sun, 19 Feb 2017 *03*:30:35 GMT

Connection: keep-alive

{**"\_\_v":**0,**"prod\_name":**"HP laptop ",**"prod\_desc":**"the new hp ",**"prod\_price":**999 ,**"\_id":**"1",**"updated\_at":**"2018-08-07T03:30:34.415Z"}

You can the same to save more records.

Now, we are going to get all products data by typing this command.

$ curl -i -H "Accept: application/json" localhost:3000/products

The response will be displayed  in JSON format like the following.

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: application/json; charset=utf-8

Content-Length: 344

ETag: W/"158-V1WRYZrNC8yW7HFEfOSSew"

Date: Sun, 19 Feb 2017 *03*:34:26 GMT

Connection: keep-alive

[{"\_id":"1","prod\_name":"HP laptop","prod\_desc":"the new hp ","prod\_price":999,"\_\_v":0,"updated\_at":"2018-08-07T03:30:34.415Z"},{"\_id":"2","prod\_name":"Sony Xperia","prod\_desc":" Sony phone"," prod\_price":600,"\_\_v":0,"updated\_at":"2018-08-05T03:33:24.941Z"}]

Next, to get one product by id use this command.

$ curl -i -H "Accept: application/json" localhost:3000/products/2

Here is the response.

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: application/json; charset=utf-8

Content-Length: 156

ETag: W/"9c-NYW3p4BkPVbiNf05Ezj+zA"

Date: Sun, 19 Feb 2017 *03*:45:48 GMT

Connection: keep-alive

{**"\_id":**"2",**"prod\_name":**"Sony Xperia",**"prod\_desc":**"Sony phone ",**"prod\_price":**600,**"\_\_v":**0,**"updated\_at":**"2018-08-05T03:33:24.941Z"}]

Next, we are going to update a product by id using this command.

$ curl -i -X PUT -H "Content-Type: application/json" -d '{"prod\_desc":"the old hp"}' localhost:3000/products/1

the response should be like this.

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: application/json; charset=utf-8

Content-Length: 185

ETag: W/"b9-k9Wipgusc9JVZAMyHgjVXw"

Date: Sun, 19 Feb 2017 *03*:38:24 GMT

Connection: keep-alive

{**"\_id":**"1",**"prod\_name":**"HP laptop",**"prod\_desc":**"the new hp ",**"prod\_price":**999,**"\_\_v":**0,**"updated\_at":**"2018-08-07T06:30:34.415Z"}

Finally, we are going to delete a product by id like the following.

$ curl -i -X DELETE localhost:3000/products/1

The response will look like this.

HTTP/1.1 200 OK

X-Powered-By: Express

Content-Type: application/json; charset=utf-8

Content-Length: 155

ETag: W/"9b-pP1KXaQhyqcMkvBlLa6pFQ"

Date: Sun, 19 Feb 2017 *03*:41:54 GMT

Connection: keep-alive

{**"\_id":**"1",**"prod\_name":**"HP laptop",**"prod\_desc":**"the new hp ",**"prod\_price":**999,**"\_\_v":**0,**"updated\_at":**"2018-08-07T06:30:34.415Z"}

**Quick REST API Creation using Node-Restful Library**

One of quicker and elegant method  for creating REST API is the Node-Restful library that you can find on[Github](https://github.com/baugarten/node-restful). You can add the library using npm command.

$ npm install node-restful --save

In app.js on the root of the project. Add require line for Node-Restful and Method-Override.

**var** restful = require('node-restful');

**var** methodOverride = require('method-override');

Change “bodyParse extended” value to true then add this lines.

app.use(bodyParser.urlencoded({'extended':'true'}));

app.use(bodyParser.json({**type**:'application/vnd.api+json'}));

app.use(methodOverride());

We will create mongoose model that can ve use together with Node-Restful library. Here we are creating “category” model.

**var** Category = app.resource = restful.model('category', mongoose.Schema({

cat\_name: String,

}))

.methods(['get', 'post', 'put', 'delete']);

Category.register(app, '/category');

Run again the app and test it using previous way for testing REST API.