Build API with Node and MongoDB

|  |  |
| --- | --- |
| MongoDB is open source, NoSQL, document oriented database designed for scaling and performance. MongoDB can be easily interface with Node.js cause Mongo provides official native driver. | |
|  |  |
|  | Setting up MongoDB |
|  | Visit official [**MongoDB download**](https://www.mongodb.org/downloads) page to download MongoDB installer for specific operating system. They will detect your OS and offer you stable version to download. Once you have download it, just follow the setup wizard instructions to install it in your system. |
|  | To begin using the MongoDB, you need to specify the location where MongoDB will store your databases. That location will be a normal folder. Create any folder say **“mongoData”** at location easily accessible to you say on Desktop and open up terminal. |
|  | Open up terminal and run following command to start MongoDB Server at default port. |
|  | mongod --dbpath /Users/Keyur/Desktop/mongoData //for mac mongod --dbpath c:/Users/Keyur/Desktop/mongoData //for windows mongod --dbpath /home/Keyur/Desktop/mongoData //for Linux |
|  |  |
|  | Now open up another terminal and run following command. |
|  | mongo |
|  |  |
|  | To create new database, run following command. |
|  | use database\_name |
|  |  |
|  | Inside database, you create collections which actually like a table in database that will store your information. |
|  |  |
|  | Installing NPM Modules |
|  | MongoDB provide native driver for Node.js called **“mongodb”**which you can use to connect to MongoDB and perform various CRUD operation. There is another MongoDB recommended node module which is quite famous called **“mongoose”** and this one we are going to use. |
|  | Create project folder and start your project by using **“npm init”**cause its good practice. This wizard will ask you some question like name, version, test script etc and at the end you will have your **package.json** file ready. |
|  | To install mongoose run following command. |
|  | npm install --save mongoose |
|  |  |
|  | Here is my package.json. |
|  | package.json  {   "name": "nodeMongo",   "version": "1.0.0",   "description": "Node.js and MongoDb tutorial.",   "main": "Server.js",   "scripts": {     "test": "mocha"   },   "repository": {     "type": "git",     "url": "https://github.com/codeforgeek/Node-and-mongo-tutorial"   },   "keywords": [     "Node.js",     "mongoDb"   ],   "author": "Shahid Shaikh",   "license": "ISC",   "bugs": {     "url": "https://github.com/codeforgeek/Node-and-mongo-tutorial/issues"   },   "homepage": "https://github.com/codeforgeek/Node-and-mongo-tutorial",   "dependencies": {     "body-parser": "^1.13.3",     "express": "^4.13.3",     "mongoose": "^4.1.2"   } } |
|  | You can copy this and create your package file and run following command to install the modules. |
|  | npm install |
|  |  |
|  | Setting up Our Server |
|  | To set up our project Server, we are going to use[Express](https://codeforgeek.com/code/nodejs/express/) module. Here is our basic server. |
|  |  |
|  |  |
|  | Server.js  var express     =   require("express"); var app         =   express(); var bodyParser  =   require("body-parser"); var router      =   express.Router();  app.use(bodyParser.json()); app.use(bodyParser.urlencoded({"extended" : false}));  router.get("/",function(req,res){     res.json({"error" : false,"message" : "Hello World"}); });  app.use('/',router);  app.listen(3000); console.log("Listening to PORT 3000"); |
|  |  |
|  | Run the code using following command. |
|  | npm start |
|  |  |
|  | Creating Basic MongoDB Model |
|  | Mongoose allows you to create models ( OR Schema OR tables ) in your MongoDB database. To connect mongoDB database to our Node.js here is two-line of code. |
|  | var mongoose    =   require("mongoose"); mongoose.connect('mongodb://localhost:27017/demoDb'); */\** *\* MongoDB port is 27017 by default.* *\* Assuming you have created mongoDB database named "demoDb".* *\*/* |
|  |  |
|  | Unlike SQL queries, MongoDB uses JSON structure to create schema in model. So for example if you want to create simple table **user\_login** with two column say **email** and **password** you need to write following code |
|  | var mongoSchema =   mongoose.Schema; var userSchema  = {     "userEmail" : String,     "userPassword" : String }; mongoose.model('user\_login',userSchema); |
|  |  |
|  | Create separate folder called **“model”** and inside create file named **“mongo.js”** and add following line of code. |
|  | /model/mongo.js  var mongoose    =   require("mongoose"); mongoose.connect('mongodb://localhost:27017/demoDb'); *// create instance of Schema* var mongoSchema =   mongoose.Schema; *// create schema* var userSchema  = {     "userEmail" : String,     "userPassword" : String }; *// create model if not exists.* module.exports = mongoose.model('userLogin',userSchema); |
|  |  |
|  | In Server.js add following line. |
|  | var mongoOp     =   require("./models/mongo"); |
|  |  |
|  | Creating RESTful API’s using Node and MongoDB |
|  | Let’s develop simple RESTful engine using MongoDB. |
|  | So our Resource will be **“users”** and on that we are going to allow following CRUD operation. |
|  | * GET /users – Return all Users from MongoDB * POST /users – Add new user in MongoDB * GET /users/:id – Return User with matched ID * PUT /users/:id – Update users information * DELETE /users/:id – Delete particular user |
|  |  |
| 1. | GET /users – Return all Users from MongoDB |
|  | Server.js  var express     =   require("express"); var app         =   express(); var bodyParser  =   require("body-parser"); var mongoOp     =   require("./models/mongo"); var router      =   express.Router();  app.use(bodyParser.json()); app.use(bodyParser.urlencoded({"extended" : false}));  router.get("/",function(req,res){     res.json({"error" : false,"message" : "Hello World"}); });  *//route() will allow you to use same path for different HTTP operation.* *//So if you have same URL but with different HTTP OP such as POST,GET etc* *//Then use route() to remove redundant code.*  router.route("/users")     .get(function(req,res){         var response = {};         mongoOp.find({},function(err,data){         *// Mongo command to fetch all data from collection.*             if(err) {                 response = {"error" : true,"message" :"Error fetching data"};             } else {                 response = {"error" : false,"message" :data};             }             res.json(response);         });     });  app.use('/',router);  app.listen(3000); console.log("Listening to PORT 3000"); |
|  | To test the API, open up POSTMAN and hit **“localhost:3000/users”**. |
|  |  |
| 2. | POST /users – Add new user in MongoDB |
|  | Server.js  var express     =   require("express"); var app         =   express(); var bodyParser  =   require("body-parser"); var mongoOp     =   require("./models/mongo"); var router      =   express.Router();  app.use(bodyParser.json()); app.use(bodyParser.urlencoded({"extended" : false}));  router.get("/",function(req,res){     res.json({"error" : false,"message" : "Hello World"}); });  *//route() will allow you to use same path for different HTTP operation.* *//So if you have same URL but with different HTTP OP such as POST,GET etc* *//Then use route() to remove redundant code.*  router.route("/users")     .get(function(req,res){         ------------------------------------------------------     })     .post(function(req,res){         var db = new mongoOp();         var response = {};         *// fetch email and password from REST request.*         *// Add strict validation when you use this in Production.*         db.userEmail = req.body.email;          *// Hash the password using SHA1 algorithm.*         db.userPassword =  require('crypto')                           .createHash('sha1')                           .update(req.body.password)                           .digest('base64');         db.save(function(err){         *// save() will run insert() command of MongoDB.*         *// it will add new data in collection.*             if(err) {                 response = {"error" : true,"message" :"Error adding data"};             } else {                 response = {"error" : false,"message" :"Data added"};             }             res.json(response);         });     });  app.use('/',router);  app.listen(3000); console.log("Listening to PORT 3000"); |
|  |  |
|  | To add new user in MongoDB, open up POSTMAN, use POST as type and add following in input payload. |
|  | {     "email" : "keyur.555kn@gmail.com",     "password" : "ami" } |
|  |  |
| 3. | GET /users/:id – Return User with matched ID. |
|  | Server.js  var express     =   require("express"); var app         =   express(); var bodyParser  =   require("body-parser"); var mongoOp     =   require("./models/mongo"); var router      =   express.Router();  app.use(bodyParser.json()); app.use(bodyParser.urlencoded({"extended" : false}));  router.get("/",function(req,res){     res.json({"error" : false,"message" : "Hello World"}); });  *//route() will allow you to use same path for different HTTP operation.* *//So if you have same URL but with different HTTP OP such as POST,GET etc* *//Then use route() to remove redundant code.*  router.route("/users")     .get(function(req,res){         ------------------------------     })     .post(function(req,res){         ------------------------------     });  router.route("/users/:id")     .get(function(req,res){         var response = {};         mongoOp.findById(req.params.id,function(err,data){         *// This will run Mongo Query to fetch data based on ID.*             if(err) {                 response = {"error" : true,"message" :"Error fetching data"};             } else {                 response = {"error" : false,"message" :data};             }             res.json(response);         });     })  app.use('/',router);  app.listen(3000); console.log("Listening to PORT 3000"); |
|  | To check this route, hit our first API to get all users and copy any one ID. Then just paste the ID right after the URL and hit the request. |
|  |  |
| 4. | PUT /users/:id – Update users information. |
|  | Server.js  var express     =   require("express"); var app         =   express(); var bodyParser  =   require("body-parser"); var mongoOp     =   require("./models/mongo"); var router      =   express.Router();  app.use(bodyParser.json()); app.use(bodyParser.urlencoded({"extended" : false}));  router.get("/",function(req,res){     res.json({"error" : false,"message" : "Hello World"}); });  *//route() will allow you to use same path for different HTTP operation.* *//So if you have same URL but with different HTTP OP such as POST,GET etc* *//Then use route() to remove redundant code.*  router.route("/users")     .get(function(req,res){         -------------------------------     })     .post(function(req,res){         -------------------------------     });  router.route("/users/:id")     .get(function(req,res){         -------------------------------     })     .put(function(req,res){         var response = {};         *// first find out record exists or not*         *// if it does then update the record*         mongoOp.findById(req.params.id,function(err,data){             if(err) {                 response = {"error" : true,"message" :"Error fetching data"};             } else {             *// we got data from Mongo.*             *// change it accordingly.*                 if(req.body.userEmail !== undefined) {                     *// case where email needs to be updated.*                     data.userEmail = req.body.userEmail;                 }                 if(req.body.userPassword !== undefined) {                     *// case where password needs to be updated*                     data.userPassword =req.body.userPassword;                 }                 *// save the data*                 data.save(function(err){                     if(err) {                         response = {"error" : true,"message": "Error updating data"};                     } else {                         response = {"error" :false,"message" : "Data is updated for "+req.params.id};                     }                     res.json(response);                 })             }         });     })  app.use('/',router);  app.listen(3000); console.log("Listening to PORT 3000"); |
|  | In order to execute it, first get the ID of any user and the select type of request as PUT and pass data which you want to update in JSON format. |
|  | You can select that user detail to cross verify it. Just change the type to GET. |
|  |  |
| 5. | DELETE /users/:id – Delete particular user. |
|  | Server.js  var express     =   require("express"); var app         =   express(); var bodyParser  =   require("body-parser"); var mongoOp     =   require("./models/mongo"); var router      =   express.Router();  app.use(bodyParser.json()); app.use(bodyParser.urlencoded({"extended" : false}));  router.get("/",function(req,res){     res.json({"error" : false,"message" : "Hello World"}); });  router.route("/users")     .get(function(req,res){         ----------------------------------------     })     .post(function(req,res){         ----------------------------------------     });  router.route("/users/:id")     .get(function(req,res){         ----------------------------------------     })     .put(function(req,res){         ----------------------------------------     })     .delete(function(req,res){         var response = {};         *// find the data*         mongoOp.findById(req.params.id,function(err,data){             if(err) {                 response = {"error" : true,"message" :"Error fetching data"};             } else {                 *// data exists, remove it.*                 mongoOp.remove({\_id :req.params.id},function(err){                     if(err) {                         response = {"error" :true,"message" : "Error deleting data"};                     } else {                         response = {"error" :true,"message" : "Data associated with "+req.params.id+"is deleted"};                     }                     res.json(response);                 });             }         });     })  app.use('/',router);  app.listen(3000); console.log("Listening to PORT 3000"); |
|  | Just select the request type as DELETE and pass the ID which user you want to delete. |
|  |  |