Upload/store images in MySQL using Node.js, Express & Multer

Last modified: August 17, 2021 (https://www.bezkoder.com/node-js-upload-image-mysql/) bezkoder (https://www.bezkoder.com/author/bezkoder/) Node.js (https://www.bezkoder.com/category/node-js/)

In this tutorial, I will show you how to upload/store image in MySQL database using Node.js with the help of Multer and Express.

Related Posts:
– How to upload multiple files in Node.js (https://bezkoder.com/node-js-upload-multiple-files/)
– Upload & resize multiple images in Node.js using Express, Multer, Sharp (https://bezkoder.com/node-js-upload-resize-multiple-images/)
– How to upload/store images in MongoDB using Node.js, Express & Multer (https://bezkoder.com/node-js-upload-store-images-mongodb/)

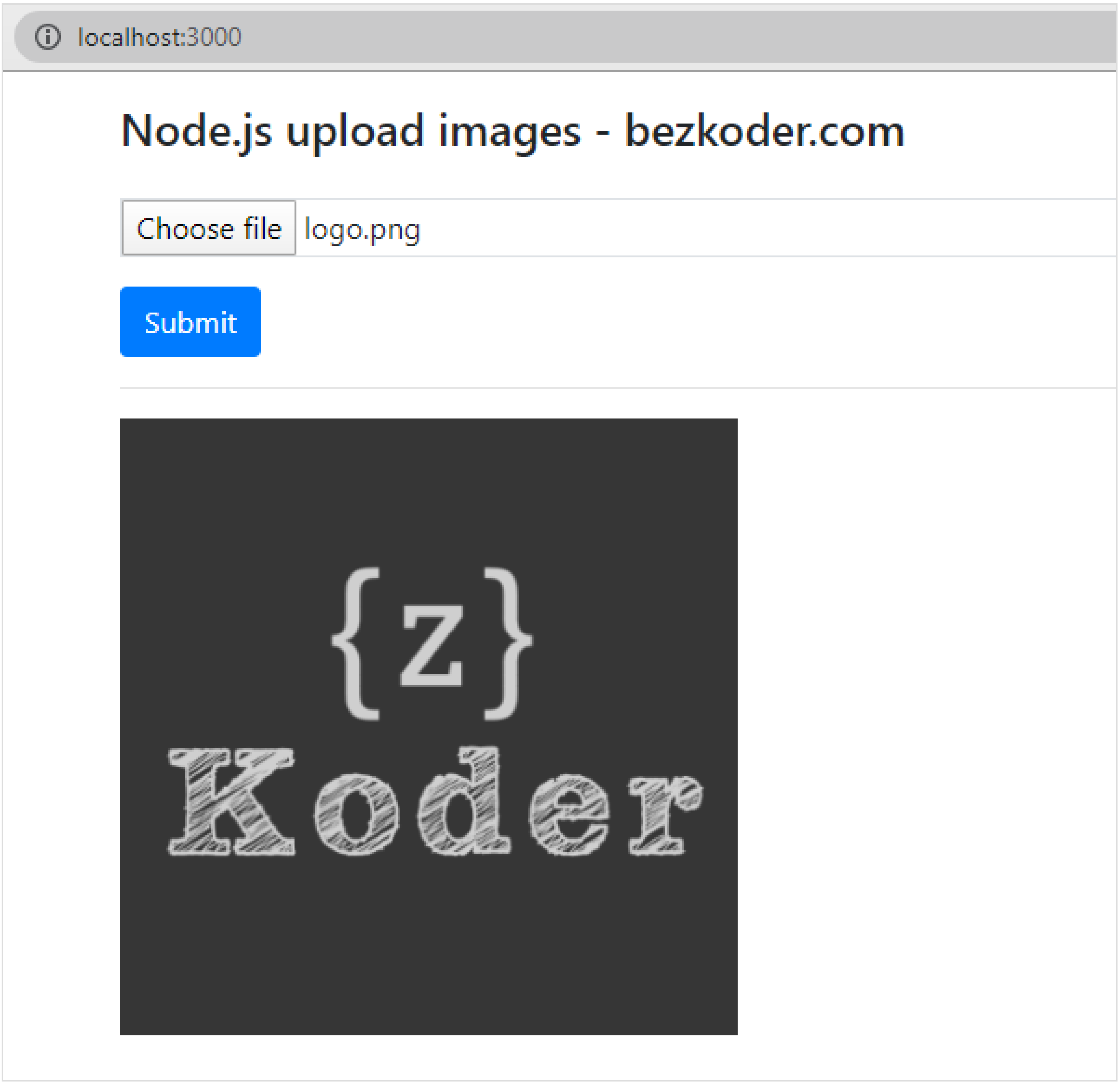
More Practice:
– Node.js Rest APIs example with Express, Sequelize & MySQL (https://bezkoder.com/node-js-express-sequelize-mysql/)
– Node.js & JWT – Token Based Authentication & Authorization example (https://bezkoder.com/node-js-jwt-authentication-mysql/)
– Dockerize Node.js Express and MySQL example – Docker Compose (https://www.bezkoder.com/docker-compose-nodejs-mysql/)

Contents [hide]

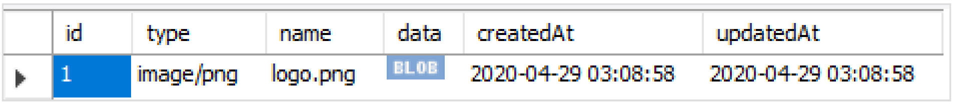
Source Code

Node.js upload/store image in MySQL overview

We're gonna make a Node.js application like this:

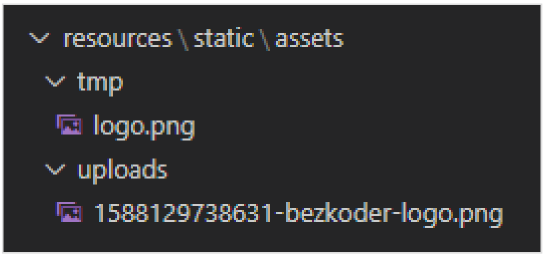


Click on **Submit** button, the file will be uploaded to MySQL database:



id	type	name	data	createdAt	updatedAt
1	image/png	logo.png	BLOB	2020-04-29 03:08:58	2020-04-29 03:08:58

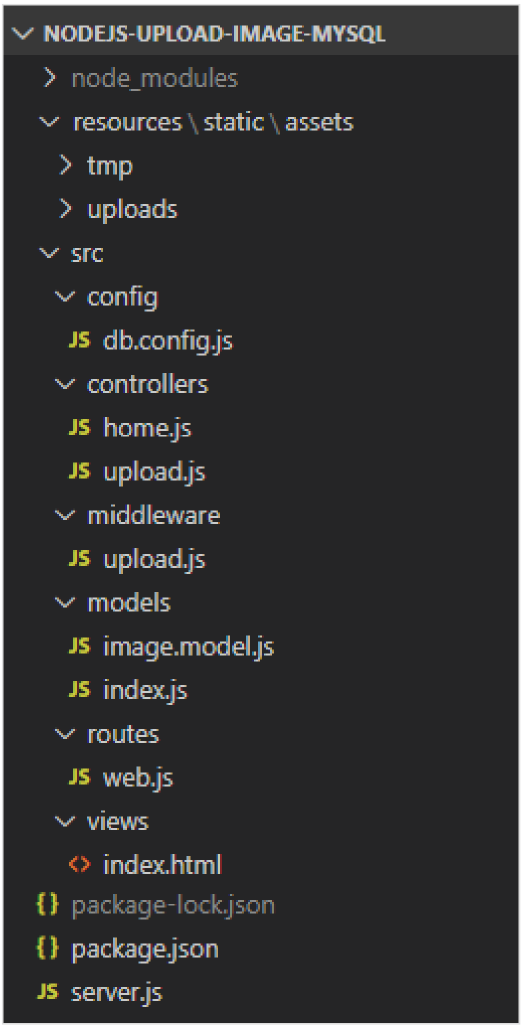
We also store the uploaded image in **upload** folders before saving its data to MySQL.
Then to check MySQL image data, we save the result data in **tmp** folder.



```
resources \ static \ assets
  tmp
    logo.png
  uploads
    1588129738631-bezkoder-logo.png
```

Project Structure

Let’s look at our project directory tree:



```
NODEJS-UPLOAD-IMAGE-MYSQL
  node_modules
  resources \ static \ assets
    tmp
    uploads
  src
    config
      db.config.js
    controllers
      home.js
      upload.js
    middleware
      upload.js
    models
      image.model.js
      index.js
    routes
      web.js
    views
      index.html
  package-lock.json
  package.json
  server.js
```

– `db.config.js` exports configuring parameters for MySQL connection & Sequelize.
– `models/index.js` : uses configuration above to initialize Sequelize, `models/image.model.js` for Sequelize model Image.
– `views/index.html` : contains HTML form for user to upload images.
– `routes/web.js` : defines routes for endpoints that is called from views, use controllers to handle requests.
– `controllers` :

- `home.js` returns `views/index.html`
- `upload.js` handles upload & store images with middleware function.

– `middleware/upload.js` : initializes Multer Storage engine and defines middleware function.
– `server.js` : initializes routes, runs Express app.

Setup Node.js modules

Open command prompt, change current directory to the root folder of our project.
Install Express, Multer, Sequelize, Mysql2 with the following command:

```
npm install express multer sequelize mysql2
```

The package.json file will look like this:

```
{
  "name": "upload-multiple-files-mysql",
  "version": "1.0.0",
  "description": "Node.js upload images to MySQL database",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "node",
    "upload",
    "image",
    "mysql"
  ],
  "author": "bezkoder",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
    "multer": "^1.4.2",
    "mysql2": "^2.1.0",
    "sequelize": "^5.21.7"
  }
}
```

Configure MySQL database & Sequelize

In the **src** folder, we create a separate **config** folder for configuration with *db.config.js* file like this:

```
module.exports = {
  HOST: "localhost",
  USER: "root",
  PASSWORD: "123456",
  DB: "testdb",
  dialect: "mysql",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000
  }
};
```

First five parameters are for MySQL connection.
`pool` is optional, it will be used for Sequelize connection pool configuration:

- `max` : maximum number of connection in pool

- `min` : minimum number of connection in pool
- `idle` : maximum time, in milliseconds, that a connection can be idle before being released
- `acquire` : maximum time, in milliseconds, that pool will try to get connection before throwing error

For more details, please visit API Reference for the Sequelize constructor (https://sequelize.org/master/class/lib/sequelize.js~Sequelize.html#instance-constructor-constructor).

Initialize Sequelize

Now we initialize Sequelize in **src**/**models** folder.

Create **src**/**models**/*index.js* with the following code:

```
const dbConfig = require("../config/db.config.js");

const Sequelize = require("sequelize");
const sequelize = new Sequelize(dbConfig.DB, dbConfig.USER, dbConfig.PASSWORD, {
  host: dbConfig.HOST,
  dialect: dbConfig.dialect,
  operatorsAliases: false,

  pool: {
    max: dbConfig.pool.max,
    min: dbConfig.pool.min,
    acquire: dbConfig.pool.acquire,
    idle: dbConfig.pool.idle,
  },
});

const db = {};

db.Sequelize = Sequelize;
db.sequelize = sequelize;

db.images = require("./image.model.js")(sequelize, Sequelize);

module.exports = db;
```

We're gonna define `Image` model in the next step.

Define the Sequelize Model

In *models* folder, create *image.model.js* file like this:

```
module.exports = (sequelize, DataTypes) => {
  const Image = sequelize.define("image", {
    type: {
      type: DataTypes.STRING,
    },
    name: {
      type: DataTypes.STRING,
    },
    data: {
      type: DataTypes.BLOB("long"),
    },
  });

  return Image;
};
```

This Sequelize Model represents **images** table in MySQL database. These columns will be generated automatically: *id*, *type*, *name*, *data*, *createdAt*, *updatedAt*.

The `data` field has `BLOB` type. A BLOB is binary large object that can hold a variable amount of data.

Create middleware for uploading & storing image

Inside **middleware** folder, create *upload.js* file with the following code:

```
const multer = require("multer");

const imageFilter = (req, file, cb) => {
  if (file.mimetype.startsWith("image")) {
    cb(null, true);
  } else {
    cb("Please upload only images.", false);
  }
};

var storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, __basedir + "/resources/static/assets/uploads/");
  },
  filename: (req, file, cb) => {
    cb(null, `${Date.now()}-bezkoder-${file.originalname}`);
  },
});

var uploadFile = multer({ storage: storage, fileFilter: imageFilter });
module.exports = uploadFile;
```

In the code above, we've done these steps:
– First, we import `multer` module.
– Next, we configure `multer` to use Disk Storage engine.
– We also define a filter to only allow images to pass.

You can see that we have two options here:
– `destination` determines folder to store the uploaded files.
– `filename` determines the name of the file inside the destination folder.
– We add the `[timestamp]-bezkoder-` prefix to the file's original name to make sure that the duplicates never occur.

Create Controller for uploading Images

controllers/upload.js

```
const fs = require("fs");

const db = require("../models");
const Image = db.images;

const uploadFiles = async (req, res) => {
  try {
    console.log(req.file);

    if (req.file == undefined) {
      return res.send(`You must select a file.`);
    }

    Image.create({
      type: req.file.mimetype,
      name: req.file.originalname,
      data: fs.readFileSync(
        __basedir + "/resources/static/assets/uploads/" + req.file.filename
      ),
    }).then((image) => {
      fs.writeFileSync(
        __basedir + "/resources/static/assets/tmp/" + image.name,
        image.data
      );

      return res.send(`File has been uploaded.`);
    });
  } catch (error) {
    console.log(error);
    return res.send(`Error when trying upload images: ${error}`);
  }
};

module.exports = {
  uploadFiles,
};
```

Now look at the `uploadFiles` function:
– First we get and check file upload from `req.file`.
– Next we use Sequelize model `create()` method to save an Image object (type, name, data) to MySQL database.
`data` is gotten from **uploads** folder (that middleware function stored the image).
– If the process is successful, we save write the image data to **tmp** folder.
– To read/write data, we use `fs.readFileSync('/path/to/file')` and `fs.writeFileSync('/path/to/file', image.data)` functions of Node.js `fs` module.

Create Controller for the view

controllers/home.js

```
const path = require("path");

const home = (req, res) => {
  return res.sendFile(path.join(`${__dirname}/../views/index.html`));
};

module.exports = {
  getHome: home
};
```

Create View for uploading image

In **views** folder, create *index.html* file with the HTML and Javascript code as below:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Node.js upload images</title>
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    />
    <style>
      div.preview-images > img {
        width: 30%;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-sm-8 mt-3">
          <h4>Node.js upload images - bezkoder.com</h4>

          <form
            class="mt-4"
            action="/upload"
            method="POST"
            enctype="multipart/form-data"
          >
            <div class="form-group">
              <input
                type="file"
                name="file"
                id="input-files"
                class="form-control-file border"
              />
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
          </form>
        </div>
      </div>
      <hr />
      <div class="row">
        <div class="col-sm-12">
          <div class="preview-images"></div>
        </div>
      </div>
    </div>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.m
    <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
    <script>
      $(document).ready(function() {
        let imagesPreview = function(input, placeToInsertImagePreview) {
          if (input.files) {
            let filesAmount = input.files.length;
            for (i = 0; i < filesAmount; i++) {
              let reader = new FileReader();
              reader.onload = function(event) {
                $($.parseHTML("<img>"))
                  .attr("src", event.target.result)
                  .appendTo(placeToInsertImagePreview);
              };
              reader.readAsDataURL(input.files[i]);
            }
          }
        };
        $("#input-files").on("change", function() {
          imagesPreview(this, "div.preview-images");
        });
      });
    </script>
  </body>
</html>
```

For HTML part, we create a form with following elements:

- `action="/upload"`
- `method="POST"`
- `enctype="multipart/form-data"`

You also need to notice the `input` tag with the `name="file"` attribute that we use in the middleware.

The jQuery script shows preview of the chosen images.
We also use Bootstrap to make the UI more comfortable to read.

Define routes

In **routes** folder, define routes in *web.js* with Express Router.

```
const express = require("express");
const router = express.Router();
const homeController = require("../controllers/home");
const uploadController = require("../controllers/upload");
const upload = require("../middleware/upload");

let routes = (app) => {
  router.get("/", homeController.getHome);

  router.post("/upload", upload.single("file"), uploadController.uploadFiles);

  return app.use("/", router);
};

module.exports = routes;
```

There are 2 routes:
– GET: Home page for the upload form.
– POST `"/upload"` to call the upload controller. This is for `action="/upload"` in the view.

The `single() (https://www.npmjs.com/package/multer#singlefieldname)` function with the parameter is the name of `input` tag (in html view: `<input type="file" name="file">`) will store the single file in `req.file` .

Create Express app server

Finally, we create an Express server.

server.js

```
const express = require("express");
const app = express();
const db = require("./src/models");
const initRoutes = require("./src/routes/web");

global.__basedir = __dirname;

app.use(express.urlencoded({ extended: true }));
initRoutes(app);

db.sequelize.sync();
// db.sequelize.sync({ force: true }).then(() => {
//   console.log("Drop and re-sync db.");
// });

let port = 3000;
app.listen(port, () => {
  console.log(`Running at localhost:${port}`);
});
```

In the code above, we call Sequelize `sync()` method.

```
db.sequelize.sync();
```

In development, you may need to drop existing tables and re-sync database. Just use `force: true` as following code:

```
db.sequelize.sync({ force: true }).then(() => {
  console.log("Drop and re-sync db.");
});
```

Run & Check result

First we need to create **tmp** and **uploads** folder with the path `resources/static/assets`.
On the project root folder, run this command: `node src/server.js`

The console shows:

```
Running at localhost:3000
```

Open your browser with url: `http://localhost:3000`.

Conclusion

Today we've learned how to upload and store image in MySQL database using **express**, **multer** & **sequelize** modules. We also built a Rest API with routes for showing the upload form and uploading images with middleware.

Happy learning! See you again.

Further Reading

- https://www.npmjs.com/package/express (https://www.npmjs.com/package/express)
- https://www.npmjs.com/package/multer (https://www.npmjs.com/package/multer)
- https://sequelize.org/v3/api/model/ (https://sequelize.org/v3/api/model/)

– Upload & resize multiple images in Node.js using Express, Multer, Sharp (https://bezkoder.com/node-js-upload-resize-multiple-images/)
– How to upload/store images in MongoDB using Node.js, Express & Multer (https://bezkoder.com/node-js-upload-store-images-mongodb/)
– Dockerize Node.js Express and MySQL example – Docker Compose (https://www.bezkoder.com/docker-compose-nodejs-mysql/)

Source Code

You can find the complete source code for this example on Github (https://github.com/bezkoder/nodejs-upload-image-mysql).

express (https://www.bezkoder.com/tag/express/) file (https://www.bezkoder.com/tag/file/)
multer (https://www.bezkoder.com/tag/multer/)
multipart file (https://www.bezkoder.com/tag/multipart-file/) mysql (https://www.bezkoder.com/tag/mysql/)
node.js (https://www.bezkoder.com/tag/node-js/) sequelize (https://www.bezkoder.com/tag/sequelize/)
upload (https://www.bezkoder.com/tag/upload/)

37 thoughts to "Upload/store images in MySQL using Node.js, Express & Multer"

Encrypter
May 21, 2020 at 8:37 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-2719)

Thanks a lot!
How do we retrieve the image/file in front?

REPLY

Trần Việt Anh
May 21, 2020 at 12:00 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-2727)

Hi thanks a lot for the tutorial. Can you also show instruction on how to make upload file request from Angular Client using Angular Service? Thanks a lot for your help.

REPLY

Trần Việt Anh
May 21, 2020 at 2:54 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-2732)

Actually I figured it out myself, but I'm struggling on downloading the file. Can you make a tutorial on downloading file from Blob data ?

REPLY

bezkoder
May 22, 2020 at 4:01 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-2739)

Hi, I will write the tutorial when having time 🙂

REPLY

Nur
May 24, 2020 at 8:46 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-2882)

Thank you for this article, it's explained really good.
A question regarding the association, whenever I want to connect the entity of Images with another entity, example:

User and Image entities, the column which is supposed to say user_id at Image entity won't appear.
It does appear without the the imgUpload controller, I'm guessing it is because the controller?

Any solution to this?

REPLY

bezkoder
May 25, 2020 at 2:03 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-2889)

Hi, I don't really understand your question. But if you want to implement associations between *user* and *image* entities. It could be One-to-Many and I have a tutorial for that:
Sequelize One-to-Many Association example with Node.js & MySQL (https://bezkoder.com/sequelize-associate-one-to-many/)

REPLY

Nur
May 26, 2020 at 7:29 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-2939)

Thank you from the bottom of my heart! You've been such a great help, I love your articles, beautifully explained.

REPLY

shall
June 23, 2020 at 3:53 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-3409)

I have loved all the tutorials you put out so far. Much appreciated. Just a quick question what is the reason for having a temp and a uploads folder when their both holding the same files?

REPLY

bezkoder
June 23, 2020 at 11:09 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-3424)

Hi, just for checking the upload file:)

REPLY

kunle oje
June 26, 2020 at 2:10 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-3458)

Hi, thanks for this. Can you show how to upload the image to cloudinary and save the response in MySQL database using same?

X

REPLY

boy
July 25, 2020 at 9:58 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-3956)

Hi I'm having trouble when coming to testing everything.
I get the error Error: Cannot find module './src/models'.
Does anyone know how to fix it,
Thanks.

REPLY

Robert Middleswarth
August 6, 2020 at 12:11 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4213)

Were you able to figure out what is going on?

Thanks
Robert

REPLY

Robert Middleswarth
August 6, 2020 at 1:36 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4214)

I figuired it out. Server.js is suppose to be in the root folder of the site not the src folder. So move it one folder up and change the start command to node server.js.

Thanks
Robert

REPLY

exe
August 1, 2020 at 12:28 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4131)

I can't see where is the table name

REPLY

bezkoder
August 2, 2020 at 4:35 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4149)

Hi, we defined the table in *image.model.js*:

```
sequelize.define("image", { ... });
```

REPLY

ARNAUD
August 15, 2020 at 11:07 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4320)

Hi! Thanks for it !
I would like to use REACT for the front end part and i must say i am struggling to figure it out…i'am new to it. Thanks a lot !

REPLY

pak
August 17, 2020 at 11:26 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4347)

great work

REPLY

Joaquim Ferrer
September 25, 2020 at 3:10 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4848)

Hey! There's one thing I don't understand. Are the images stored in the upload folder or at the db?

REPLY

bezkoder
September 26, 2020 at 12:15 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4851)

Hi, currently both of them, you can delete the images in uploads folder right after it is written to database.

REPLY

Joaquim Ferrer

September 26, 2020 at 4:21 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4861)

That’s what I thought. Thanks!

REPLY

Qua Nguyen
September 29, 2020 at 2:25 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4881)

how do you display the images stored in mysql db?

REPLY

dev
October 4, 2020 at 7:50 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-4921)

i have the same problem how do we render the image from db or using path of uploads folder

REPLY

erot
December 9, 2020 at 2:57 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-5701)

I truly love your site.. Good tutorials and explanation.

REPLY

CBcoder
December 9, 2020 at 10:16 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-5705)

bezkoder thank you for these tutorials! You have answered a lot of questions when it comes to me learning how to use stacks.

If I were to add another page or component that only a “mod” and higher should have access to, what would be the best way to go about that addition?

REPLY

bezkoder
December 10, 2020 at 7:04 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-5710)

Hi, you can read this tutorial:
Node.js Express: JWT example | Token Based Authentication & Authorization (https://bezkoder.com/node-js-jwt-authentication-mysql/)

REPLY

Adam
March 23, 2021 at 3:55 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-7715)

hello bezkoder you are really awesome, this helping a lot while doing my project.
I have a question is if I want to change database table to another I created, and delete createdAt, updatedAt columns.
I read code many times, but I can't figure it out.
Can you help me to point out where the config?
Thanks a lot

REPLY

Adam
March 23, 2021 at 4:17 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-7716)

I found the table name here sequelize.define("image", { … });
but why it will plus 's' whiling complete create the table

REPLY

aydin
April 5, 2021 at 12:21 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-7918)

hi how can I upload and download file using react and node,js and mysql but files save in project specified directory and it path save in mysql table?

REPLY

temeka
April 12, 2021 at 9:05 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-8059)

This is my first time to read your tutorial and I am in fact pleasant.

REPLY

menna ahmed
May 8, 2021 at 6:02 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-8445)

Thank you so much this tutorial was so helpful.
I have an issue while trying to connect to a pre-exisiting database, that the id field is inserted with null value, I tried that following while configuring the field in image.mode.js :

```
id: {
type: DataTypes.INTEGER,
primaryKey: true,
autoIncrement: true,
omitNull: true,
allowNull: false,
}
```

but it's still inserting null value, can you please help?

REPLY

Gary
May 8, 2021 at 11:38 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-8450)

Hi, I am trying to change script code in the HTML to change the image dimensions before displaying them on the screen. Any insight on how to go about doing that?

REPLY

FrankWip
July 5, 2021 at 2:00 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-9427)

Awesome tutorial!

REPLY

X

Edwardundum
July 29, 2021 at 12:08 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-10206)

Thanks!

REPLY

Keith2VAL
September 13, 2021 at 2:38 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-12155)

Great tutorial!

REPLY

Ronni1eCinly
September 14, 2021 at 8:08 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-12223)

Thank you

REPLY

Randal
September 15, 2021 at 2:28 am (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-12245)

Great tutorial!

REPLY

David23pooth
September 17, 2021 at 2:39 pm (https://www.bezkoder.com/node-js-upload-image-mysql/#comment-12367)

Thanks so much

REPLY

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

X

Name *

Email *

Website

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

Dart/Flutter – Encode/Decode Image to/from Base64 String (https://www.bezkoder.com/dart-base64-image/)

React Redux CRUD example with API calls (https://www.bezkoder.com/react-redux-crud-example/)

Search...

X

FOLLOW US

X

(https://facebook.com/bezkoder)

(https://www.youtube.com/channel/UCp0mx9RH0JxaFsmMvKXA86Q)

(https://github.com/bezkoder)

TOOLS

Json Formatter (https://www.bezkoder.com/json-formatter/)

X