

MONGODB –Database

NODE JS - Server

Express JS – Web Site

Install mongodb driver, express and body-parser

```
C:\Users\admin>npm install body-parser --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\admin\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\admin\package.json'
npm WARN admin No description
npm WARN admin No repository field.
npm WARN admin No README data
npm WARN admin No license field.

+ body-parser@1.18.3
updated 1 package and audited 376 packages in 2.789s
found 0 vulnerabilities
```

C:\Users\admin\node_express_insert.js

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

```
var app = express();
```

```
var bodyParser = require('body-parser');
```

```
var urlencodedParser = bodyParser.urlencoded({ extended: true });
```

```
var MongoClient = require('mongodb').MongoClient;
```

```
app.use(express.static(__dirname + '/../public'));
```

```
app.get('/register', function (req, res) {
```

```
    res.sendFile(__dirname + "/" + "node_register.html");
```

```
app.post('/process_post', urlencodedParser, function (req, res) {
```

```
    // Prepare output in JSON format
```

```
    response = {      CompanyID:req.body.cid,      name:req.body.cname,  
    address:req.body.addr  };
```

```

MongoClient.connect('mongodb://localhost:27017/', function(err, db)
{
    if (err) throw err;
    console.log("Connected to Database");
    var dbo=db.db("mydb");
    //insert document in mongodb
    dbo.collection('customers').insert(response, function(err, result)
    {
        if (err) throw err;
console.log("1 document inserted in your mongodb database" ); });});
console.log(response); // display in node console window
res.end(JSON.stringify(response));}) // display in browser window
var server = app.listen(8080, function () {
    var host = server.address().address
    var port = server.address().port

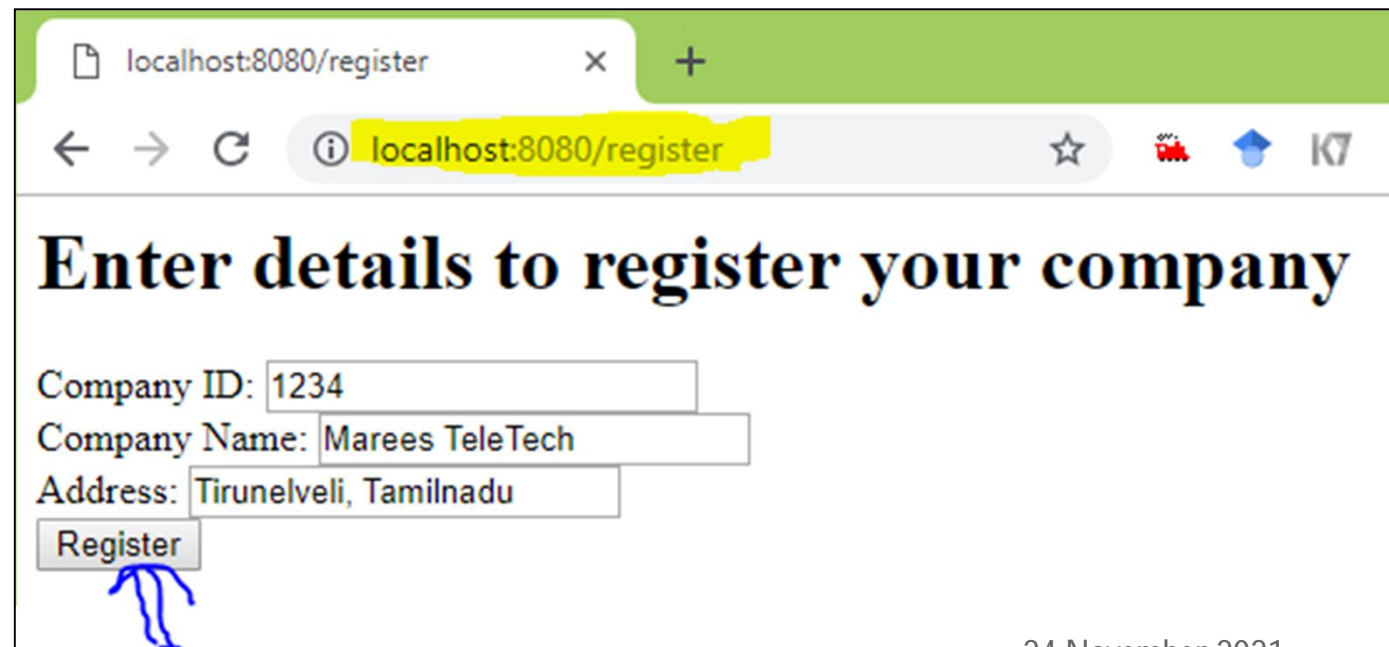
    console.log("Example app listening at http://%s:%s/register", host,
port));

```

Run server program in node console

```
C:\WINDOWS\system32\cmd.exe - node marees_node_express_insert.js  
Microsoft Windows [Version 6.3.9600]  
<c> 2013 Microsoft Corporation. All rights reserved.  
  
C:\Users\admin>node marees_node_express_insert.js  
Example app listening at http://:::8080/register
```

Request the server : Run client in web browser by
localhost:8080/register



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/register'. The page title is 'Enter details to register your company'. The form contains three input fields: 'Company ID' with the value '1234', 'Company Name' with the value 'Marees TeleTech', and 'Address' with the value 'Tirunelveli, Tamilnadu'. Below the input fields is a 'Register' button. A blue arrow points to the 'Register' button.

```
C:\WINDOWS\system32\cmd.exe - node marees_node_express_insert.js

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\admin>node marees_node_express_insert.js
Example app listening at http://:::8080/register
{ CompanyID: '1234',
  name: 'Marees TeleTech ',
  address: 'Tirunelveli, Tamilnadu' }
(node:4252) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
Connected to Database
(node:4252) DeprecationWarning: collection.insert is deprecated. Use insertOne, insertMany or bulkWrite instead.
1 document inserted in your mongodb database
```

See the JSON formatted inserted document in server and client.

```
localhost:8080/process_post

localhost:8080/process_post

{"CompanyID":"1234","name":"Marees TeleTech ","address":"Tirunelveli, Tamilnadu"}
```

```
C:\WINDOWS\system32\cmd.exe - mongo
> db.customers.find().pretty();
{
  "_id" : ObjectId("5bd03b0cceeec371f78bede66"),
  "name" : "Company Inc",
  "address" : "Highway 37"
}
{
  "_id" : ObjectId("5bd03c00f68d6d19d854898a"),
  "CompanyID" : "1",
  "name" : "Mareeswari Industries",
  "address" : "Near Vellore Port "
}
{
  "_id" : ObjectId("5bd045a02359901b806cbfba"),
  "CompanyID" : "1",
  "name" : "Marees Infotech",
  "address" : "UIT Road, Vellore"
}
{
  "_id" : ObjectId("5bd1697c6e6699109c3aa4e1"),
  "CompanyID" : "1234",
  "name" : "Marees TeleTech ",
  "address" : "Tirunelveli, Tamilnadu"
}
>
```

See the JSON formatted inserted document in mongo client

Join Collections

- MongoDB is not a relational database, but you can perform a left outer join by using the **\$lookup** stage.
- The **\$lookup** stage lets you specify which collection you want to join with the current collection, and which fields that should match.

{ \$lookup:

{

from: <collection to join> ,

localField: <field from the input documents> ,

foreignField: <field from the documents of the "from" collection> ,

as: <output array field> } }

- orders

```
[  
  { _id: 1, product_id: 154, status: 1 }  
]
```

- products

```
[  
  { _id: 154, name: 'Chocolate Heaven' },  
  { _id: 155, name: 'Tasty Lemons' },  
  { _id: 156, name: 'Vanilla Dreams' }  
]
```

```
var MongoClient = require('mongodb').MongoClient;  
var url = "mongodb://127.0.0.1:27017/";
```

```
MongoClient.connect(url, function(err, db) {  
  if (err) throw err;  
  var dbo = db.db("mydb");  
  dbo.collection('orders').aggregate([  
    {  
      $lookup:  
      {  
        from: 'products',  
        localField: 'product_id',  
        foreignField: '_id',  
        as: 'orderdetails'  
      }  
    }  
  ]).toArray(function(err, res) {  
    if (err) throw err;  
    console.log(JSON.stringify(res));  
    db.close();  
  });  
});
```

Appendix

Using mongoose module

- Installation in NPM (express, body-parser, mongoose)

```
C:\Users\admin>npm install body-parser --save
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\admin\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\admin\package.json'
npm WARN admin No description
npm WARN admin No repository field.
npm WARN admin No README data
npm WARN admin No license field.

+ body-parser@1.18.3
updated 1 package and audited 376 packages in 2.789s
found 0 vulnerabilities
```

C:\Users\admin\marees_node_express_db.js

```
var express = require("express");  
var app = express();  
var port = 3000;  
var bodyParser = require('body-parser');  
app.use(bodyParser.json());  
app.use(bodyParser.urlencoded({ extended: true }));  
var mongoose = require("mongoose");  
mongoose.Promise = global.Promise;  
mongoose.connect("mongodb://localhost:27017/");  
var nameSchema = new mongoose.Schema({  
  firstName: String,  lastName: String});
```

```
var User = mongoose.model("User", nameSchema);
app.get("/", (req, res) => {
  res.sendFile(__dirname + "/marees_index.html");});
app.post("/addname", (req, res) => {
  var myData = new User(req.body);
  myData.save()
    .then(item => {
      res.send("Name saved to database");    })
    .catch(err => {
      res.status(400).send("Unable to save to database");  });});
app.listen(port, () => {
  console.log("Server listening on port " + port);});
```

C:\Users\admin\marees_index.html

```
<!DOCTYPE html> <html> <head>
<title>Node and MongoDB</title> </head> <body>
<h1>Client-Server-Database using Node, Express and
MongoDB</h1>
<form method="post" action="/addname">
<label>Enter Your Name</label><br>
<input type="text" name="firstName" placeholder="Enter first
name..." required>
<input type="text" name="lastName" placeholder="Enter last
name..." required>
<input type="submit" value="Add to DB">
</form> </body></html>
```

Explanation

- The **MEAN stack** is used to describe development using MongoDB, Express.js, Angular.js and Node.js. In this program I will show you how to use Express.js, Node.js and MongoDB.js.
- A **RESTful API** is an application program interface that uses HTTP requests to GET, PUT, POST and DELETE data. We will be **using** an API to define when **we add data to our database** and when we **read** from the database.
- ```
app.get("/", (req, res) => {
 res.send("Hello World");
});
```
- The `app.use` line will listen to requests from the browser and will return the text "Hello World" back to the browser.



- **Displaying our Website to Users**

- Now we want to display our html file that we created. To do this we will need to change the app.use line in our application(.js) file.

- ```
app.use("/", (req, res) => {  
  res.sendFile(__dirname + "/index.html");  
});
```

- **Connecting to the Database**

- Now that we have the mongoose module installed, we need to connect to the database in our application(.js) file. MongoDB, by default, runs on port 27017. You connect to the database by telling it the location of the database and the name of the database.

- ```
var mongoose = require("mongoose");
mongoose.Promise =
global.Promise;mongoose.connect("mongodb://localhost:27017/");
```

- **Creating a Database Schema**

- Once the user enters data in the input field and clicks the add button, we want the contents of the input field to be stored in the database. In order to know the format of the data in the database, we need to have a Schema.
- ```
var nameSchema = new mongoose.Schema({  
  firstName: String,  
  lastName: String  
});
```
- Once we have built our Schema, we need to create a model from it.
- ```
var User = mongoose.model("User", nameSchema);
```

- **Creating a RESTful API**

- Now that we have a connection to our database, we need to create the mechanism by which data will be added to the database. This is done through our REST API. We will need to create an endpoint that will be used to send data to our server. Once the server receives this data then it will store the data in the database.
- An endpoint is a route that our server will be listening to get data from the browser. We already have one route that we have created already in the application and that is the route that is listening at the endpoint "/" which is the homepage of our application.
- **HTML Verbs in a REST API**
- The communication between the client(the browser) and the server is done through an HTTP verb. GET (read), PUT (update), POST (create), and DELETE (delete)

- The form in our .html file used a post method to call this endpoint
- `app.post("/addname", (req, res) => {  
});`
- **Express Middleware**
- To fill out the contents of our endpoint, we want to store the firstName and lastName entered by the user into the database. The values for firstName and lastName are in the body of the request that we send to the server. We want to capture that data, convert it to JSON and store it into the database.
- Express.js version 4 removed all middleware. To parse the data in the body we will need to add middleware into our application to provide this functionality. We will be using the body-parser module.
- `var bodyParser = require('body-parser');  
app.use(bodyParser.json());  
app.use(bodyParser.urlencoded({ extended: true }));`

- **Saving data to database**

- Mongoose provides a save function that will take a JSON object and store it in the database. Our body-parser middleware, will convert the user's input into the JSON format for us.
- To save the data into the database, we need to create a new instance of our model that we created early. We will pass into this instance the user's input. Once we have it then we just need to enter the command "save".
- Mongoose will return a promise on a save to the database. A promise is what is returned when the save to the database completes. This save will either finish successfully or it will fail. A promise provides two methods that will handle both of these scenarios.
- If this save to the database was successful it will return to the .then segment of the promise. In this case we want to send text back the user to let them know the data was saved to the database.

- If it fails it will return to the `.catch` segment of the promise. In this case, we want to send text back to the user telling them the data was not saved to the database. It is best practice to also change the `statusCode` that is returned from the default 200 to a 400. A 400 `statusCode` signifies that the operation failed.
- ```
app.post("/addname", (req, res) => {  
  var myData = new User(req.body);  
  myData.save()  
    .then(item => {  
      res.send("item saved to database");  
    })  
    .catch(err => {  
      res.status(400).send("unable to save to database");  
    });  
});
```

- **Testing our code**
- Make sure you have mongo running.
- <https://codeburst.io/hitchhikers-guide-to-back-end-development-with-examples-3f97c70e0073>

```
C:\Users\admin>node marees_node_express_db.js
(node:8432) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
Server listening on port 3000
```

← → ↻ ⓘ localhost:3000

Client-Server-Database using Node, Express and MongoDB

Enter Your Name

← → ↻ ⓘ localhost:3000

Client-Server-Database using Node, Express and MongoDB

Enter Your Name

← → ↻ ⓘ localhost:3000/addname

Name saved to database

References

- MEAN Stack Program (MEAN is an acronym for MongoDB, ExpressJS, AngularJS and Node.js)
- <https://www.djamware.com/post/5b00bb9180aca726dee1fd6d/mean-stack-angular-6-crud-web-application>