Client-Server-Database

MONGODB Manipulation Using NODE JS

- **Install MongoDB Driver** to connect node js and mongodb
- Type command → C:\Users\admin>npm install mongodb

C:\WINDOWS\system32\cmd.exe

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

C:\Users\admin>npm install mongodb
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\admin\packa
ge.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.

+ mongodb@3.1.6
updated 1 package in 10.797s

C:\Users\admin>
```

Creating and closing a connection to a MongoDB database

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

var url = 'mongodb://localhost/EmployeeDB';

MongoClient.connect(url, function(err, db) {

    console.log("Connected");

        db.close();
    });
```

1. using the mongodb driver
2. Specify the connection url
3. Connecting to the database
4. Writing to the console log
5. Closing the database connection

- The first step is to include the mongoDB module, which is done through the require function.
- Next, we specify our connection string to the database. In the connect string, there are 3 key values which are passed.
  1. The first is 'mongodb' which specifies that we are connecting to a mongoDB database.
  2. The next is 'localhost' which means we are connecting to a database on the local machine.
  3. The next is 'EmployeeDB' which is the name of the database defined in our MongoDB database.
- The next step is to actually connect to our database. The connect function takes in our URL and has the facility to specify a callback function. It will be called when the connection is opened to the database.
- In the function, we are writing the string "Connection established" to the console to indicate that a successful connection was created.
- Finally, we are closing the connection using the db.close statement.

Create MongoDB database through Node JS

C:\Users\admin\ testdb_mongo.js

```
var MongoClient=require('mongodb').MongoClient;
console.log(MongoClient);
var url="mongodb://localhost:27017/employee_db";
MongoClient.connect(url,function(err,db) {
  if (err)          throw err;
  console.log("Database created!");
  db.close();});
```

- 27017 is the default port number for MongoDB.
- MongoDB waits until you have created a collection (table), with at least one document (record) before it actually creates the database (and collection).

Run the connectivity program by C:\Users\admin>node testdb_mongo.js

```
Command Prompt
C:\Users\Admin\nodejs>cd MyNodejsApp

C:\Users\Admin\nodejs\MyNodejsApp>node testdb_mongo.js
{ [Function: MongoClient]
  super_:
   { [Function: EventEmitter]
     once: [Function: once],
     EventEmitter: [Circular],
     usingDomains: false,
     defaultMaxListeners: [Getter/Setter],
     init: [Function],
     listenerCount: [Function] },
  connect:
   { [Function]
     MongoError: [Function: MongoError],
     MongoNetworkError: [Function: MongoNetworkError],
     MongoTimeoutError: [Function: MongoTimeoutError],
     MongoServerSelectionError: [Function: MongoServerSelectionError],
     MongoParseError: [Function: MongoParseError],
     MongoWriteConcernError: [Function: MongoWriteConcernError],
     MongoBulkWriteError: [Function: BulkWriteError],
     BulkWriteError: [Function: BulkWriteError],
     Admin: [Function: Admin],
     MongoClient: [Circular],
     Db:
      { [Function: Db]
     TWO_PWR_48_DBL_: 281474976710656,
     TWO_PWR_64_DBL_: 18446744073709552000,
     TWO_PWR_63_DBL_: 9223372036854776000,
     ZERO: [Timestamp],
     ONE: [Timestamp],
     NEG_ONE: [Timestamp],
     MAX_VALUE: [Timestamp],
     MIN_VALUE: [Timestamp],
     TWO_PWR_24_: [Timestamp],
     Timestamp: [Circular] },
   BSONRegExp: { [Function: BSONRegExp] BSONRegExp: [Circular] },
   Decimal128:
    { [Function: Decimal128] fromString: [Function], Decimal128: [Circular] },
   connect: [Circular],
   instrument: [Function] } }
(node:19568) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the Mo
ngoClient constructor.
Database created!

C:\Users\Admin\nodejs\MyNodejsApp>_
```

C:\Users\admin\testdb_coll.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
 if (err) throw err;
 var dbo = db.db("mydb");
 dbo.createCollection("customers", function(err, res) {  if (err) throw
 err;
   console.log("Collection created!");  db.close(); });
   });
```

Command prompt output:

C:\Users\Admin\nodejs\MyNodejsApp>node testdb_coll.js
(node:20880) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Collection created!

In VS code :

File Edit Selection View Go Run Terminal Help — testdb_coll.js - MyNodejsApp - Visual Studio Code

express.js | addapp.js | home.html | node-express-form.js | addform.js | node_cookie.js | Release Notes: 1.50.1 | testdb_mongo.js | testdb_coll.js X

testdb_coll.js > MongoClient.connect() callback > dbo.createCollection("customers") callback

```
1  var MongoClient = require('mongodb').MongoClient;
2  var url = "mongodb://localhost:27017/";
3  MongoClient.connect(url, function(err, db) {
4  if (err) throw err;
5  var dbo = db.db("mydb");
6  dbo.createCollection("customers", function(err, res)
7  {  if (err) throw err;
8  console.log("Collection created!");
9    db.close();   }); });
10
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE — Filter (e.g. text, !exclude)

```
C:\Program Files\nodejs\node.exe .\testdb_coll.js
(node:8852) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover
and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Collection created!
```

Inserting documents in a collection

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb://localhost/EmployeeDB';

MongoClient.connect(url, function(err, db) {

    db.collection('Employee').insertOne({

        Employeeid :4,
        EmployeeName: "NewEmployee"

    }
    );
    });
```

1 Use the insertOne method to insert a document

2 The document to insert in the collection

Inserting documents in a collection

C:\Users\admin\testdb_insert.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myobj = { CompanyID:"1",
                name: "VIT Infotech",
                address:  "VIT Road,Vellore" };
  dbo.collection("customers").insertOne(myobj, function(err, res) {
    if (err) throw err;
    console.log("1 document inserted");
    db.close();  });});});
```

The document to insert in the customers collection

insertOne method to insert 1 document

insertMany for array of documents to insert

C:\Users\Admin\nodejs\MyNodejsApp>node testdb_insert.js
(node:4396) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
1 document inserted

- Run the node program like above
- Run the below commands in mongo client

```
> use mydb
switched to db mydb
> db.customers.find().pretty()
{
        "_id" : ObjectId("5f8d7b318c85d345f03bb375"),
        "CompanyID" : "1",
        "name" : "VIT Infotech",
        "address" : "VIT Road,Vellore"
}
{
        "_id" : ObjectId("5f8d7be5f2a0ec28fcacbe0c"),
        "CompanyID" : "2",
        "name" : "MIT Infotech",
        "address" : "VIT Road,Vellore"
}
>
```

8

Querying for data in a MongoDB database

Using the find function to create a cursor of records

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb://localhost/EmployeeDB';

MongoClient.connect(url, function(err, db) {

  1 var cursor =db.collection('Employee').find( );

    cursor.each(function(err, doc) { 2

  3   console.log(doc);
    });
    });
```

Printing the results to the console

For each record in the cursor we are calling a function

Querying for data in a MongoDB database

C:\Users\admin\node_testdb_find.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var query = { name: “MIT Infotech" };
dbo.collection("customers").find(query).toArray(function(err,
result) {
    if (err) throw err;
    console.log(result);
    db.close();   });});});
```

9

Command Prompt - mongo

```
> db.customers.find().pretty();
{
        "_id" : ObjectId("5f8d7b318c85d345f03bb375"),
        "CompanyID" : "1",
        "name" : "VIT Infotech",
        "address" : "VIT Road,Vellore"
}
{
        "_id" : ObjectId("5f8d7be5f2a0ec28fcacbe0c"),
        "CompanyID" : "2",
        "name" : "MIT Infotech",
        "address" : "VIT Road,Vellore"
}
{
        "_id" : ObjectId("5f8dd08d2d5da4112c0a3ff5"),
        "CompanyID" : "2",
        "name" : "MIT Infotech",
        "address" : "VIT Road,Vellore"
}
>
```

Command Prompt

```
C:\Users\Admin\nodejs\MyNodejsApp>node node_testdb_find.js
(node:17072) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the Mo
ngoClient constructor.
[ { _id: 5f8d7be5f2a0ec28fcacbe0c,
    CompanyID: '2',
    name: 'MIT Infotech',
    address: 'VIT Road,Vellore' },
  { _id: 5f8dd08d2d5da4112c0a3ff5,
    CompanyID: '2',
    name: 'MIT Infotech',
    address: 'VIT Road,Vellore' } ]

C:\Users\Admin\nodejs\MyNodejsApp>
```

C:\Users\admin\node_testdb_update.js

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  var dbo = db.db("mydb");
  var myquery = { name: "MIT Infotech" };
  var newvalues = { $set: {name: "Mageshwari Industries", address:
                                  "Near Vellore Port " } };
dbo.collection("customers").updateOne(myquery, newvalues,
function(err, res) {
    if (err) throw err;
    console.log("1 document updated");
    db.close();  });});});
```

Command Prompt

```
C:\Users\Admin\nodejs\MyNodejsApp>node node_testdb_update.js
(node:14380) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the Mo
ngoClient constructor.
1 document updated

C:\Users\Admin\nodejs\MyNodejsApp>
```

Command Prompt - mongo

```
> use mydb
switched to db mydb
> db.customers.find().pretty()
{
        "_id" : ObjectId("5f8d7b318c85d345f03bb375"),
        "CompanyID" : "1",
        "name" : "VIT Infotech",
        "address" : "VIT Road,Vellore"
}
{
        "_id" : ObjectId("5f8d7be5f2a0ec28fcacbe0c"),
        "CompanyID" : "2",
        "name" : "Mageshwari Industries",
        "address" : "Near Vellore Port "
}
{
        "_id" : ObjectId("5f8dd08d2d5da4112c0a3ff5"),
        "CompanyID" : "2",
        "name" : "MIT Infotech",
        "address" : "VIT Road,Vellore"
}
>
```

MONGODB – Database NODE JS - Server Express JS – WebSite

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\packa
ge.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)})
```

16

node_register.html

```
<html>
<body>
<h1>Company Registration Form</h1>
   <form action="http://127.0.0.1:8080/process_post" method="P
OST">
      Company ID<input type="text" name="cid"><br/>
      Company Name<input type="text" name="cname"><br/>
      Address <input type="text" name="addr"><br/>
   <button>Register</button>
</form>

    </body> </html>
```

Run server program in node console

Command Prompt - node node-express-insert.js

```
C:\Users\Admin\nodejs\MyNodejsApp>node node-express-insert.js
Example app listening at http://:::8080//register
```

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

See the JSON formatted inserted document in server and client.

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 wil
l be removed in a future version. To use the new parser, pass option { useNewUrl
Parser: 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
```

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("5bd03b0cceec371f78bede66"),
        "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" : "VIT Road, Vellore"
}
{
        "_id" : ObjectId("5bd1697c6e6699109c3aa4e1"),
        "CompanyID" : "1234",
        "name" : "Marees TeleTech ",
        "address" : "Tirunelveli, Tamilnadu"
}
>
```

See the JSON formatted inserted document in mongo client

Using mongoose module- altenate way

- 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\packa
ge.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_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 + "/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\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("HelloWorld");
  });
- The app.use line will listen to requests from the browser and will return the text "Hello World" back to the browser.

Operation	Method	Resource	Example	Remarks
Read – List	GET	Collection	GET /customers	Lists objects (additional query string can be used for filtering and sorting)
Read	GET	Object	GET /customers/1234	Returns a single object (query stringmay be used to specify which fields)
Create	POST	Collection	POST /customers	Creates an object with the values specified in the body
Update	PUT	Object	PUT / customers/1234	Replaces the object with the one specified in the body
Update	PATCH	Object	PATCH /customers/1234	Modifies some properties of the object, as specified in the body
Delete	DELETE	Object	DELETE / customers/1234	Deletes the object

- **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 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,
  lastNameName: String
  });
- Once we have built our Schema, we need to create a model from it.
- var User = mongoose.model("User", nameSchema);

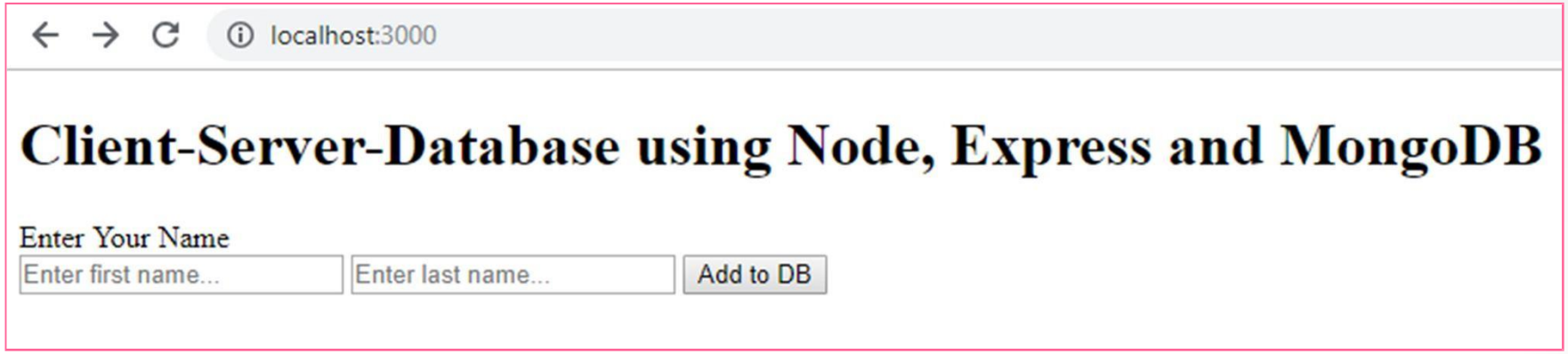
- **Creating a RESTful API**
- Now that we have a connection to our database, we need to create the mechanism by which data will be added to the database. This is done through our REST API. We will need to create an endpoint that will be used to send data to our server. Once the server receives this 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)

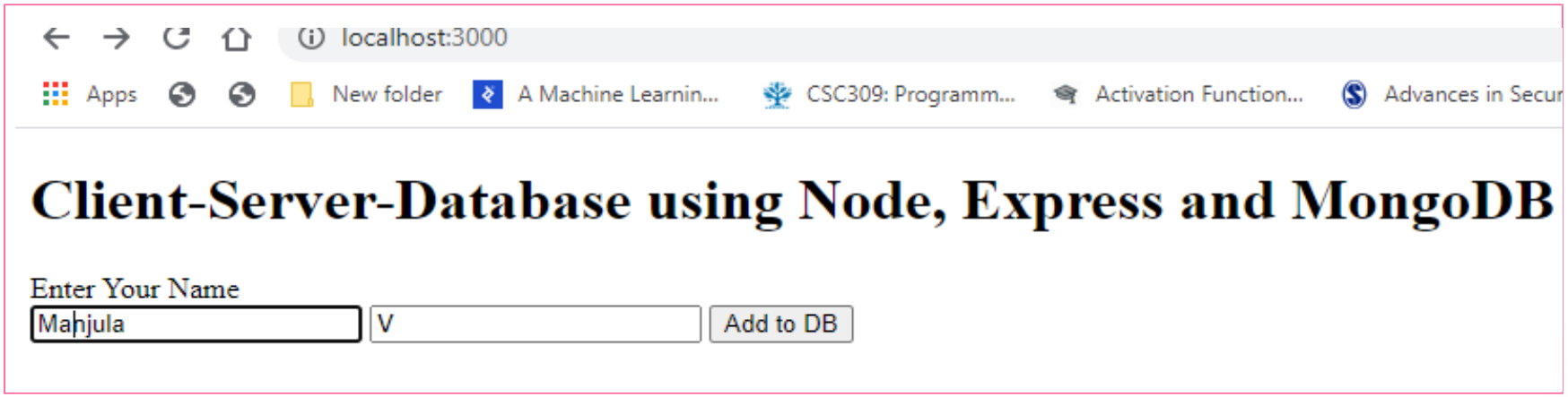
- The form in our .html file used a post method to call this endpoint
- app.post("/addname", (req, res) => {
  });
- **Express Middleware**
- To fill out the contents of our endpoint, we want to store the firstName and lastName entered by the user into the database. The values for firstName and lastName 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 }));

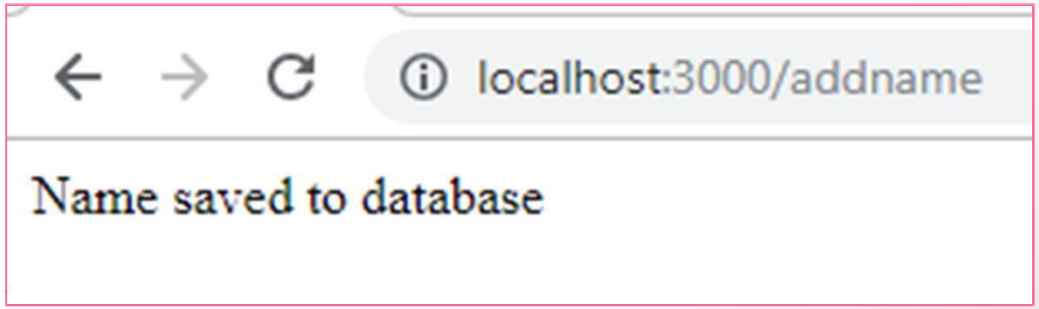
- **Saving data to database**
- Mongoose provides a save function that will take a JSON object and store it in the database. Our body-parser middleware, will convert the user's input into the JSON format for us.
- To save the data into the database, we need to create a new instance of our model that 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 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");
  });
  });

```
C:\Program Files\nodejs\node.exe .\node-express-mongoose.js
(node:18896) 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.
(node:18896) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
Server listening on port 3000
```







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

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