This objective of this lesson will be to connect to and retrieve information from the database.

For troubleshooting purposes, this operation will use the ‘morgan’ package.

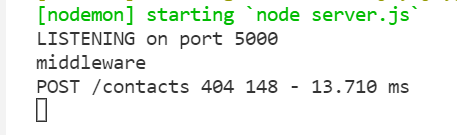
Additionally, the server will be prepared to parse the BODY of incoming HTTP messages through the installation and activation of *‘body-parser’.*

*Morgan:*

1. With the terminal, install the *morgan* package. “npm install morgan - - save”
2. In server.js import morgan const morganMonitor = require('morgan');
3. and deploy it by entering app.use(morganMonitor('tiny'));

As each Http message is received, its method will be printed in the terminal window.

Below is an example of what will be seen when a ‘post’ message is processed by the server.



body-parser: see the example code for server.js below:

The updated working ‘server.js’ is shown below

// file server.js

const express = require('express');   // import express

const cors = require('cors');      // import   cors

const allroutes = require('./routes/allroutes');

const API\_PORT = process.env.PORT || 5000;  // Run the server on Port 5000

const app = express();                // activate express as ‘app’

const bodyParser = require('body-parser'); // for parsing HTTP requests

const morganMonitor = require('morgan'); // for printing the HTTP method

app.use(cors());  // activate   CORS

app.use(morganMonitor('tiny'));  // activate the monitor

// activate the body-Parser to parse the BODY of Http: PUT statements

   app.use(bodyParser.urlencoded({

       extended: true,})

       );

   app.use(bodyParser.json());

// activate router for the api's

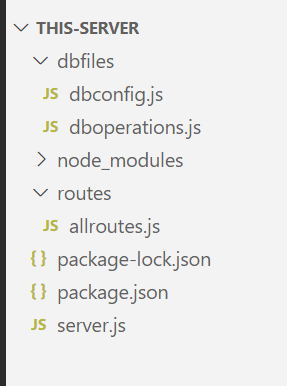
app.use(allroutes);

// activate the server

app.listen(API\_PORT, ()=> { console.log(`LISTENING on port ${API\_PORT}`) });

Database Connectivity:

1. Create a ‘dbfiles’ folder in the root (this-server) folder.
2. In the ‘dbfiles’ folder, create a file called ‘dboperations.js’ and ‘dbconfig.js’.



1. In ‘dbconfig.js’ enter your server configuration information. Below is an example of ‘dbconfig.js’:

// file dbconfig.js

const config = {

user: 'Fuoi',

    password: 'Fuoi',

    server: 'MSI',

    database: 'CustomerInfo',

    options: {

      trustServerCertificate: true,

      trustedconnection: false,

      enableArithAbort: true,

      instancename: 'SQLEXPRESS'

    },

    port: 1433

}

module.exports = config;

1. Create a database operations file which makes the database connection and forms the interface between the HTTP METHOD and the DATABASE QUERY

The database operation for the HTTP:GET route is defined in allroutes.js.

// file dboperations.js

const config = require('./dbconfig'); // access database configuration

const sql = require('mssql');

const getContacts = async() => {

     try {

        let pool = await sql.connect(config); // Log on to database

        let theseContacts = await pool.request().query(

            "SELECT \* FROM Contacts"

        );

        return theseContacts;

     }  // end try block

     catch(error) {

         console.log("DATABASE CONNECTION EROR IN     getThisContact  Error Posted Next Line");

         console.log(error);

     } // end catch

}  // end getContacts

module.exports = { getContacts}

1. In allroutes.js, import getContacts via dboperations.
2. In allroutes.js write the function which will respond to the HTTP:GET command and obtain the information from the database

// file allroutes.js

const express = require('express');

const router = express.Router();

// imports to implement api's

var bodyParser = require('express');

var cors = require('cors');

const app = express();  // to create an object of express

const dboperations = require('../dbfiles/dboperations')

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

app.use(bodyParser.json());

app.use(cors());

router.use((request, response, next) => {

  console.log('middleware');   // for authentication software etc

  next();

})

router.route('/contacts').get((request, response)=> {

  dboperations.getContacts().then(result => {

       console.log(result);

       response.json(result);

   })

})

router.route('/user').get((request, response)=> {

        console.log(" console    This is your first routed api call");

        response.send(" This is your first routed api call");

    })

    router.route('/test').get((request, response)=> {

      console.log(" console    This is a call to test");

      response.send(" This is a test");

  })

module.exports = router;    // export all of the routes

7) With POSTMAN or the BROWSER test both api’s to ensure client/server connection.

<http://localhost:5000/user>

and <http://localhost:5000/test>

and <http://localhost:5000/contacts>

With the last url all of the entries currently in the ‘contacts’ database will be listed in the terminal