**Contoh Node.js Express CRUD dengan MySQL**

Dalam tutorial ini kita akan melihat bagaimana melakukan contoh Node js Express CRUD dengan mysql. Kerangka kerja Node js Express dengan Database MySQL di kami akan memberi Anda contoh langkah demi langkah operasi CRUD di node js.

Jadi, mari kita mulai dan melihat contoh mentah di node.js atau cara membuat aplikasi CRUD (Create Read Update Delete) menggunakan Node.js dan MySQL.

Langkah 1: Buat Aplikasi Node.js untuk contoh mentah node.js

Langkah 2: Instal Prasyarat

Langkah 3: Buat Database, Tabel dan Koneksi untuk contoh node js crud dengan mysql

Langkah 4: Buat Rute CRUD

Langkah 5: Crete View Files untuk contoh mentah di node.js

Langkah 6: Jalankan index.js

[**Node.js**](https://nodejs.org/)  adalah open source, lingkungan runtime lintas platform untuk mengembangkan aplikasi sisi server dan jaringan. Anda harus memiliki pemahaman dasar tentang nodejs.

[**Express.js**](https://expressjs.com/)  adalah salah satu kerangka kerja web paling trending untuk node.js. Itu dibangun di atas modul http node.js, dan menambahkan dukungan untuk perutean, middleware, sistem tampilan, dll.

[**MySQL**](https://dev.mysql.com/downloads/installer/)  adalah sistem manajemen basis data relasional sumber terbuka yang dapat beroperasi di berbagai platform. Ini adalah kemudahan manajemen.

**Langkah 1: Buat Aplikasi Node.js untuk contoh mentah node.js**

Pada langkah ini kita akan membuat aplikasi node.js menggunakan perintah di bawah ini.

1. Mkdir node\_js\_crud\_operation (untuk mebuat folder node\_js\_crud\_operation
2. Cd node\_js\_crud\_operation (untuk masuk dalam forder node\_js\_crud\_operation didalam git bash)
3. npm init (untuk install package.json)

**Langkah 2: Instal template JavaScript Tertanam (ejs)**

Pada langkah ini kita akan menginstal ejs menggunakan perintah di bawah ini:

npm install ejs (akan muncul di dependency)

EJS atau Embedded Javascript Templateing adalah mesin templating yang digunakan oleh Node.js. Mesin templat membantu membuat templat HTML dengan kode minimal. Juga, dapat menyuntikkan data ke dalam template HTML di sisi klien dan menghasilkan HTML akhir. EJS adalah bahasa templating sederhana yang digunakan untuk menghasilkan markup HTML dengan JavaScript biasa. Ini juga membantu untuk menyematkan JavaScript ke halaman HTML.

**Langkah 3: Instal express.js di Aplikasi**

Sekarang instal Express js menggunakan perintah di bawah ini:

npm install express –save

Menyediakan middleware pesan flash Express.js yang berfungsi untuk rendering atau pengalihan.

**Memerlukan middleware express session atau sesi**[**cookie**](https://github.com/expressjs/cookie-session) **session  sebelum menerapkan middleware ini.**

**Langkah 4: Instal Ketergantungan**

Selanjutnya, kita perlu menginstal beberapa dependensi. salin perintah di bawah ini dan tempel di terminal Anda

# flash message:

npm install express-flash --save

#session like PHP:

npm install express-session –save

# to send PUT and DELETE requests:

npm install method-override –save

# driver to connect Node.js with MYSQL:

npm install mysql –save

**Langkah 5: Buat Database dan Tabel untuk contoh node js crud dengan mysql**

Sekarang buat database dan buat  tabel **pengguna  .**salin struktur tabel di bawah ini dan jalankan.

CREATE TABLE `users` (

`id` int(50) NOT NULL,

`name` varchar(100) DEFAULT NULL,

`email` varchar(100) DEFAULT NULL,

`position` varchar(100) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `users`

ADD PRIMARY KEY (`id`);

ALTER TABLE `users`

MODIFY `id` int(50) NOT NULL AUTO\_INCREMENT;

COMMIT;

Setelah membuat tabel dan database. kami menghubungkan database dan aplikasi. Di direktori proyek, buat folder bernama ' **lib'** . Di folder **lib** , buat file bernama ' **db.js'** .

**jalur: lib/db.js**

var mysql = require('mysql');

var connection = mysql.createConnection({

    host:'localhost',

    user:'root',

    password:'',

    database:'node\_js\_crud'

});

connection.connect(function(error){

    if(!!error) {

        console.log(error);

    } else {

        console.log('Database Connected Successfully..!!');

    }

});

module.exports = connection;

**Step 6 : Make Routes**

In this step first create **routes folder.**In routes folder create **users.js**file and copy below code.

var express = require('express');

var router = express.Router();

var dbConn  = require('../lib/db');

// display user page

router.get('/', function(req, res, next) {

    dbConn.query('SELECT \* FROM users ORDER BY id desc',function(err,rows)     {

        if(err) {

            req.flash('error', err);

            // render to views/users/index.ejs

            res.render('users',{data:''});

        } else {

            // render to views/users/index.ejs

            res.render('users',{data:rows});

        }

    });

});

// display add user page

router.get('/add', function(req, res, next) {

    // render to add.ejs

    res.render('users/add', {

        name: '',

        email: '',

        position:''

    })

})

// add a new user

router.post('/add', function(req, res, next) {

    let name = req.body.name;

    let email = req.body.email;

    let position = req.body.position;

    let errors = false;

    if(name.length === 0 || email.length === 0 || position === 0) {

        errors = true;

        // set flash message

        req.flash('error', "Please enter name and email and position");

        // render to add.ejs with flash message

        res.render('users/add', {

            name: name,

            email: email,

            position:position

        })

    }

    // if no error

    if(!errors) {

        var form\_data = {

            name: name,

            email: email,

            position:position

        }

        // insert query

        dbConn.query('INSERT INTO users SET ?', form\_data, function(err, result) {

            //if(err) throw err

            if (err) {

                req.flash('error', err)

                // render to add.ejs

                res.render('users/add', {

                    name: form\_data.name,

                    email: form\_data.email,

                    position:form\_data.position

                })

            } else {

                req.flash('success', 'User successfully added');

                res.redirect('/users');

            }

        })

    }

})

// display edit user page

router.get('/edit/(:id)', function(req, res, next) {

    let id = req.params.id;

    dbConn.query('SELECT \* FROM users WHERE id = ' + id, function(err, rows, fields) {

        if(err) throw err

        // if user not found

        if (rows.length <= 0) {

            req.flash('error', 'User not found with id = ' + id)

            res.redirect('/users')

        }

        // if user found

        else {

            // render to edit.ejs

            res.render('users/edit', {

                title: 'Edit User',

                id: rows[0].id,

                name: rows[0].name,

                email: rows[0].email,

                position: rows[0].position

            })

        }

    })

})

// update user data

router.post('/update/:id', function(req, res, next) {

    let id = req.params.id;

    let name = req.body.name;

    let email = req.body.email;

    let position = req.body.position;

    let errors = false;

    if(name.length === 0 || email.length === 0 || position.length === 0) {

        errors = true;

        // set flash message

        req.flash('error', "Please enter name and email and position");

        // render to add.ejs with flash message

        res.render('users/edit', {

            id: req.params.id,

            name: name,

            email: email,

            position:position

        })

    }

    // if no error

    if( !errors ) {

        var form\_data = {

            name: name,

            email: email,

            position:position

        }

        // update query

        dbConn.query('UPDATE users SET ? WHERE id = ' + id, form\_data, function(err, result) {

            //if(err) throw err

            if (err) {

                // set flash message

                req.flash('error', err)

                // render to edit.ejs

                res.render('users/edit', {

                    id: req.params.id,

                    name: form\_data.name,

                    email: form\_data.email,

                    position: form\_data.position

                })

            } else {

                req.flash('success', 'User successfully updated');

                res.redirect('/users');

            }

        })

    }

})

// delete user

router.get('/delete/(:id)', function(req, res, next) {

    let id = req.params.id;

    dbConn.query('DELETE FROM users WHERE id = ' + id, function(err, result) {

        //if(err) throw err

        if (err) {

            // set flash message

            req.flash('error', err)

            // redirect to user page

            res.redirect('/users')

        } else {

            // set flash message

            req.flash('success', 'User successfully deleted! ID = ' + id)

            // redirect to user page

            res.redirect('/users')

        }

    })

})

module.exports = router;

**Step 7 : Create View Files for crud xample in node.js**

We need to create view files for add, edit and view. We've set **EJS**for templating. Go to the **views**folder & create a folder called **users**. Under the users folder, make files named index.ejs, add.ejs and edit.ejs.

**Path : users/index.ejs**

<html lang="en">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <title>Users</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

</head>

<body class="container" style="margin-top: 50px;">

    <% if (messages.success) { %>

    <div class="alert alert-success" role="alert"><%- messages.success %></div>

    <% } %>

    <% if (messages.error) { %>

    <div class="alert alert-danger" role="alert"><%- messages.error %></div>

    <% } %>

    <div class="card">

        <div class="card-header">

            <ul class="nav nav-pills w-100">

                <li class="nav-pill active">

                    <a class="nav-link">Users</a>

                </li>

                <li class="nav-pill ml-auto">

                    <a class="nav-link active" href="/users/add">Add User</a>

                </li>

            </ul>

        </div>

    <div class="card-body">

        <% if(data.length) { %>

        <table class="table">

            <thead>

                <tr>

                    <th scope="col">#</th>

                    <th scope="col">Name</th>

                    <th scope="col">Email</th>

                    <th scope="col">Position</th>

                    <th width="200px">Action</th>

                </tr>

            </thead>

            <tbody>

            <% for(var i = 0; i< data.length; i++) { %>

                <tr>

                    <th scope="row"><%= (i+1) %></th>

                    <td><%= data[i].name%></td>

                    <td><%= data[i].email%></td>

                    <td><%= data[i].position%></td>

                    <td>

                        <a class="btn btn-success edit" href="../users/edit/<%=data[i].id%>">Edit</a>

                        <a class="btn btn-danger delete" onclick="return alert('Are you sure want to delete this record?')" href="../users/delete/<%=data[i].id%>">Delete</a>

                    </td>

                </tr>

            <% } %>

            </tbody>

        </table>

        <% } %>

        <!-- if result is empty -->

        <% if(!data.length) { %>

            <p class="text-center">No users found!</p>

        <% } %>

        </div>

    </div>

</body>

</html>

**Path : users/add.ejs**

<html lang="en">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <title>Add User</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

</head>

<body class="container" style="margin-top: 50px;">

    <% if (messages.error) { %>

    <div class="alert alert-danger" role="alert"><%- messages.error %></div>

    <% } %>

    <div class="card">

        <div class="card-header">

            Add Users

        </div>

        <div class="card-body">

            <form action="/users/add" method="post">

                <div class="form-group">

                    <label>Name:</label>

                    <input type="text" class="form-control" name="name" value="<%= name %>" autocomplete="off">

                </div>

                <div class="form-group">

                    <label>Email:</label>

                    <input type="text" class="form-control" name="email" value="<%= email %>" autocomplete="off">

                </div>

                <div class="form-group">

                    <label>Position:</label>

                    <input type="text" class="form-control" name="position" value="<%= position %>" autocomplete="off">

                </div>

                <div class="form-group">

                    <input type="submit" class="btn btn-info" value="Add"/>

                </div>

            </form>

        </div>

    </div>

</body>

</html>

**Path : users/edit.ejs**

<html lang="en">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <title>Edit User</title>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">

</head>

<body class="container" style="margin-top: 50px;">

    <% if (messages.error) { %>

    <div class="alert alert-danger" role="alert"><%- messages.error %></div>

    <% } %>

    <div class="card">

        <div class="card-header">

            Edit User

        </div>

        <div class="card-body">

            <form action="/users/update/<%= id %>" method="post">

                <div class="form-group">

                    <label>Name:</label>

                    <input type="text" class="form-control" name="name" value="<%= name %>" autocomplete="off">

                </div>

                <div class="form-group">

                    <label>Email:</label>

                    <input type="text" class="form-control" name="email" value="<%= email %>" autocomplete="off">

                </div>

                <div class="form-group">

                    <label>Position:</label>

                    <input type="text" class="form-control" name="position" value="<%= position %>" autocomplete="off">

                </div>

                <div class="form-group">

                    <input type="submit" class="btn btn-info" value="Update"/>

                </div>

            </form>

        </div>

    </div>

</body>

</html>

**Langkah 8 : Buat File index.js**

Pada langkah ini impor semua dependensi seperti express js, pesan flash, driver mysql dll.

var createError = require('http-errors');

var express = require('express');

var path = require('path');

var flash = require('express-flash');

var session = require('express-session');

var mysql = require('mysql');

var connection  = require('./lib/db');

var usersRouter = require('./routes/users');

var app = express();

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

app.use(express.json());

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

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use(session({

    cookie: { maxAge: 60000 },

    store: new session.MemoryStore,

    saveUninitialized: true,

    resave: 'true',

    secret: 'secret'

}))

app.use(flash());

app.use('/users', usersRouter);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

  next(createError(404));

});

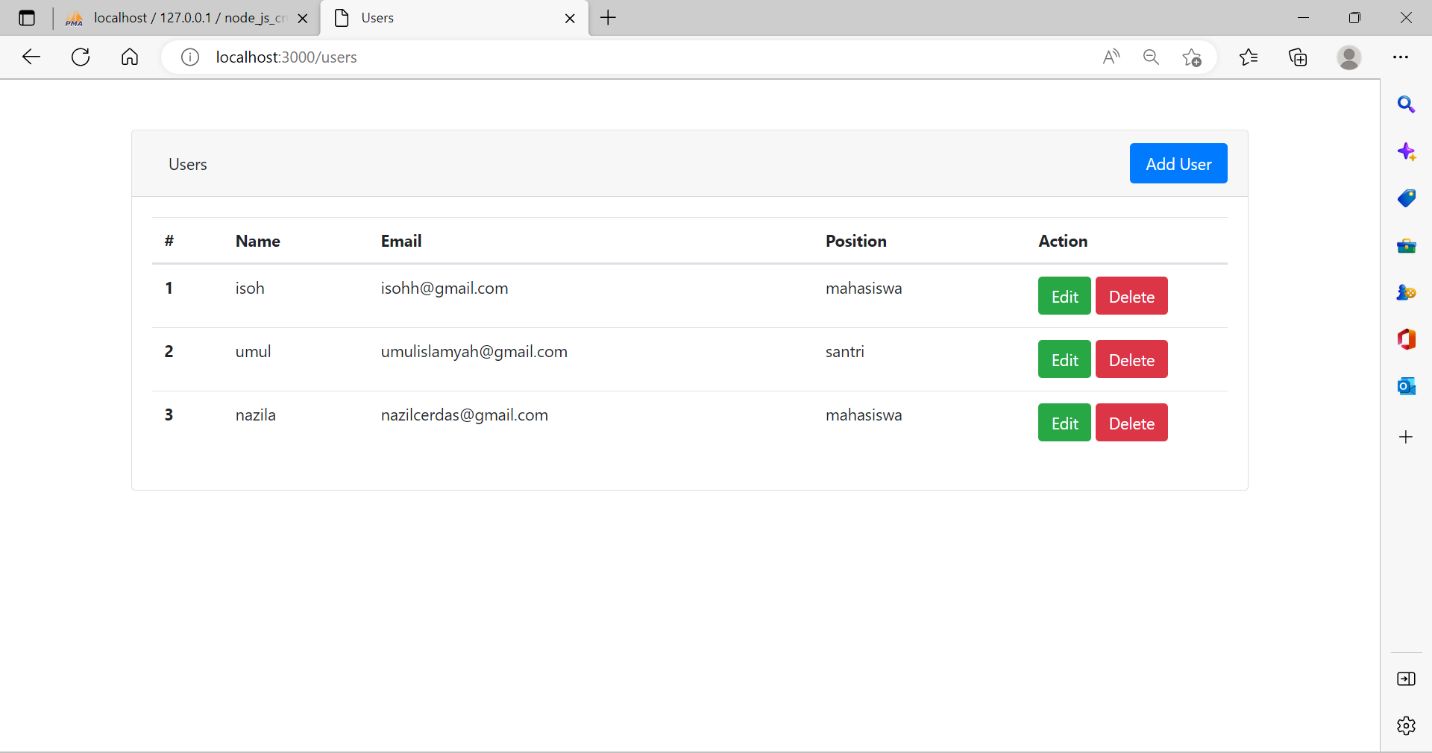
app.listen(3000);

**Langkah 9 : Jalankan file index.js**

Sekarang jalankan index.js menggunakan kode di bawah ini

node index.js

Di browser Anda buka di bawah URL:

[**http://localhost:3000/users**](http://localhost:3000/users)

Setelah menjalankan perintah ini Anda akan mendapatkan output seperti screenshot di bawah ini.