https://shorturl.at/kAHNV

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Modules in express**

**Middlewares**

**File Handling**

**MongoDB CRUD operations (Local and Cloud)**

**Hosting MongoDBCRUD application**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

==============================================================

Modules in Express

==============================================================

<>

login

- login.js

logout

- logout.js

- server.js

\*\*\*login.js\*\*\*

//import express module

let express = require('express')

//create router instance

let router = express.Router()

//create get request

router.get("/", (req, res) => {

res.send({ 'message': 'Welcome to Login module' })

})

//create one more get request

router.get("/login/:uname/:upwd", (req, res) => {

//here we are reading url parameters using params

let uname = req.params.uname

let upwd = req.params.upwd

if (uname == 'admin' && upwd == 'admin')

res.json({ 'login': 'success' })

else

res.json({ 'login': 'failed' })

})

//export router

module.exports = router

\*\*\*logout.js\*\*\*

//import express module

let express = require('express')

//create router instance

let router = express.Router()

//create get request

router.get("/", (req, res) => {

res.json({ 'message': 'Welcome to logout module' })

})

//create one more get request

//URL :- http://localhost:8080/logout/logout/?uname=admin&upwd=admin

router.get("/logout", (req, res) => {

//here we are reading get parameters using query

let uname = req.query.uname

let upwd = req.query.upwd

if (uname == 'admin' && upwd == 'admin')

res.send({ 'logout': 'Success' })

else

res.send({ 'logout': 'Failed' })

})

//export router

module.exports = router

\*\*\*server.js\*\*\*

//import modules

let express = require('express')

let login = require('./login/login')

let logout = require('./logout/logout')

//create rest object

let app = express()

//use modules

app.use('/login', login)

app.use('/logout', logout)

//assign port no

app.listen(8080)

console.log('Server listening port no 8080')

/\*

http://localhost:8080/login

http://localhost:8080/login/login/admin/admin

http://localhost:8080/logout

http://localhost:8080/logout/logout?uname=admin&upwd=admin

\*/

==============================================================

Middlewares

==============================================================

- Middlewares execute before routing in ReST api

- An Express application can use the following types of middleware:

- Application-level middleware

- Router-level middleware

- Error-handling middleware

- Built-in middleware

- Third-party middleware

Application-level middleware

- Bind application-level middleware to an instance of the app

object by using the app.use() and app.METHOD() functions,

where METHOD is the HTTP method of the request that the

middleware function handles

Router-level middleware

- Router-level middleware works in the same way as

application-level middleware, except it is bound

to an instance of express.Router().

Error-handling middleware

- Define error-handling middleware functions in the same way as other

middleware functions, except with four arguments instead of

three, specifically with the signature (err, req, res, next))

Built-in middleware

- Express has the following built-in middleware functions:

- express.static: serves static assets such as HTML files, images, and so on.

- express.json: parses incoming requests with JSON payloads. NOTE: Available with Express 4.16.0+

- express.urlencoded: parses incoming requests with URL-encoded payloads. NOTE: Available with Express 4.16.0+

Third-party middleware

- Use third-party middleware to add functionality to Express apps.

- Install the Node.js module for the required functionality,

then load it in your app at the application level or at the router level.

//initialyse project

//>npm init -y

//download express module

//>yarn add express --save

\*\*\*server.js\*\*\*

//import express module

let express = require('express')

//create rest object

let app = express()

//middleware

//Eg01

const myMiddleware = (req, res, next) => {

console.log("I am in middleware")

next()

}

//use middleware

app.use(myMiddleware)

//create rest api

app.get("/", (req, res) => {

console.log("I will execute after middleware")

})

//Eg02

const myMiddleware = (req, res, next) => {

let uname = req.query.uname

let upwd = req.query.upwd

if (uname == 'admin' && upwd == 'admin')

next()

else

res.json({ 'Authentication': 'Failed' })

}

//use middleware

app.use(myMiddleware)

//rest api

app.get("/", (req, res) => {

res.json({ 'Authentication': 'Success' })

})

//url localhost:8080/?uname=admin&upwd=admin

//Eg03

let my\_fun = require('./my\_fun')

app.use(my\_fun({ uname: 'admin', upwd: 'admin' }))

//rest api

app.get("/", (req, res) => {

res.send({ 'Authentication': 'Success' })

})

//assign port no

app.listen(8080)

console.log("Server listening port no 8080")

\*\*\*my\_fun.js\*\*\*

module.exports = (obj) => {

return (req, res, next) => {

let uname = obj.uname

let upwd = obj.upwd

if (uname == 'admin' && upwd == 'admin')

next()

else

res.json({ 'Authentication': 'Failed' })

}

}

==============================================================

File Operations:-

==============================================================

- 'fs' is the predefined module in NodeJS.

- fs stands for file system.

- fs module is used to work with files.

- fs is native module, so no need to download it.

- fs module is available with node engine.

There are two modes

- Synchronous mode

- Asynchronous mode

Reading data from files

- ‘readFile(-,-)’ is the predefined function in the fs module.

- This function is used to read data asynchronously.

- ‘readFileSync(-)’ is the predefined function in the fs module.

- This function is used to read data synchronously.

<>

* path.js

static

- sample.txt

read

- readSync.js

- readAsync.js

\*\*\*sample.txt\*\*

Welcome to fs module

\*\*\*path.js\*\*\*

module.exports = "../static/sample.txt"

\*\*\*readSync.js\*\*\*

//import fs module

let fs = require('fs')

//read data synchronously

/\*let data = fs.readFileSync(require('../path'),'utf-8')

console.log(data)\*/

let data = fs.readFileSync(require('../path'))

console.log(data.toString())

\*\*\*readAsync.js\*\*\*

//import fs module

let fs = require('fs')

//read data asynchronously

fs.readFile(require('../path'), "utf-8", (err, data) => {

if (err)

console.log('Error while reading data ', err.code)

else

console.log(data)

})

Writing Data to files

- 'writeFile(-,-,-)' is the predefined function in 'fs' module, used to write data to files asynchronously.

- 'writeFileSync(-,-)' is the predefined function in 'fs' module, used to write data to files synchronously.

<>

write

- writeSync.js

- writeAsync.js

\*\*\*writeAsync.js\*\*\*

//import fs module

let fs = require('fs')

//write data asynchronously

fs.writeFile(require('../path'), "hi", (err) => {

if (err)

console.log('Error while writing data')

else

console.log('Data write success')

})

\*\*\*writeSync.js\*\*\*

//import fs module

let fs = require('fs')

//write data synchronously

fs.writeFileSync(require('../path'),'Hello...!')

console.log('Write Success')

Appending data to files

- 'appendFile(-,-,-)' is the predefined function in the 'fs' module used to append data to files Asynchronously.

- 'appendFileSync(-,-)' is the predefined function in the fs module used to append data to files synchronously.

append

- appedSync.js

- appendAsync.js

\*\*\*appendAsync.js\*\*\*

//import fs module

let fs = require('fs')

let path = require('../path')

//append data asynchronously

fs.appendFile(path, 'Good Morning...!', (err) => {

if (err)

console.log('Error while appending data')

else

console.log('Append Success')

console.log(fs.readFileSync(path, 'utf-8'))

})

\*\*\*appedSync.js\*\*\*

//import fs module

let fs = require('fs')

let path = require('../path')

//append data synchronously

fs.appendFileSync(path, '\tWelcome')

console.log('Data appended\n', fs.readFileSync(path, 'utf-8'))

Open File

- 'open(-,-,-)' is the predefined function in fs module used to open files Asynchronously.

- 'openSync(-,-)' is the predefined function in the fs module used to open files Synchronously.

modes:-

r -> open file for reading

r+ -> read + write

rs -> read synchronous

w -> writing (create file if doesn't exist, truncate if exist.)

w+-> read + write

a -> append

a+ -> read and append

<>

open

- openSync.js

- openAsync.js

\*\*\*openSync.js\*\*\*

//

let fs = require('fs')

//

fs.openSync(require('../path'),"r+")

console.log('File Opened successfully')

\*\*\*openAsync.js\*\*\*

—Assignment—

Deleting Contents from file

- 'truncate(-,-,-)' is the predefined function in the fs module used to delete contents from a file.

- —Check for Sync—

<>

truncate

- truncate.js

\*\*\*truncate.js\*\*\*

//

let fs = require('fs')

//

fs.truncate(require('../path'), 5, (err) => {

if (err)

console.log('Error while deleting data')

else

console.log("Delete success")

})

Delete file

- unlink(-,-) is the predefined function used to delete file. [unlinkSync(-) ?]

delete

- delete.js

\*\*\*delete.js\*\*\*

//

let fs = require('fs')

//

fs.unlink(require('../path'), (err) => {

if (err)

console.log('Error while deleting file')

else

console.log('Delete success')

})

==============================================================

Interaction with MongoDB

==============================================================

- MongoDB is a lightweight NoSQL database.

- MongoDB follows client server architecture.

- MongoDB follows the 'mongodb' protocol.

- Mongoserver running on port no 27017.

- MongoDB supports JSON.

Installation of MongoDB

1. Download and install mongodb(community edition) (Prefer v5.x)

https://www.mongodb.com/try/download/community

\*\*\* For windows OS \*\*\*

2. Create directory structure

c:/data/db

3. set path for environment variables

-> computer

-> properties

-> advanced system settings

-> advanced tab

-> environment variables

-> click on path

-> edit

-> new

-> copy and paste path of installation of mongodb

where mongo.exe and mongod.exe located

For cloud database

1. Login to mongodb.com (atlas)

- Perform relevant network access and database access settings

2. Browse collections

3. Create database 'nodedb' with collection 'products'

- Add my own data

4. Insert documents

5. Click on databases from left panel

6. Click on connect

7. click on compass

8. Copy url and follow steps below url from web page

mongodb+srv://admin:admin@mdb.vtkja.mongodb.net/

MongoDB Queries (Local Database)

- Open command prompt

>mongo

- create and switch to database

>use nodedb;

- create collection

>db.createCollection("products")

- insert data

>db.products.insert({"p\_id":111, "p\_name":"P\_one", "p\_cost":10000})

- fetch data

>db.products.find()

- show databases

>show dbs

- delete database

>db.dropDatabase()

url :-> mongodb://localhost:27017

<>

fetch

- fetch.js

insert

- insert.js

update

- update.js

delete

- delete.js

- server.js

- url.js

- fetch.js file is used to fetch data from collection.

- insert.js file is used to insert data in collection.

- update.js file is used to update data from collection.

- delete.js file is used to delete data from collection.

- server.js file is used to collaborate the modules.

- url.js file is used to store url of mongodb

initialize project

>npm init -y

download modules

express

mongodb

body-parser

cors

>yarn add express mongodb body-parser cors --save

\*\*\*url.js\*\*\*

module.exports = 'mongodb://localhost:27017'

//module.exports = 'mongodb+srv://admin:admin@mdb.vtkja.mongodb.net/'

\*\*\*fetch.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.get("/", (req, res) => {

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection :- ', err)

else {

let db = conn.db('nodedb')

db.collection('products').find().toArray((err, array) => {

if (err)

console.log('Error:- ', err)

else {

console.log('Data sent')

res.send(array)

conn.close()

}

})

}

})

})

//export router

module.exports = router

\*\*\*insert.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.post("/", (req, res) => {

let obj = {

"p\_id": req.body.p\_id,

"p\_name": req.body.p\_name,

"p\_cost": req.body.p\_cost

}

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection')

else {

let db = conn.db('nodedb')

db.collection('products').insertOne(obj, (err) => {

if (err)

res.send({ 'insert': 'error' })

else {

console.log("Data inserted")

res.json({ 'insert': 'success' })

conn.close()

}

})

}

})

})

//export router

module.exports = router

\*\*\*update.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.put("/", (req, res) => {

let p\_id = req.body.p\_id

let obj = {

"p\_name": req.body.p\_name,

"p\_cost": req.body.p\_cost

}

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection')

else {

let db = conn.db('nodedb')

db.collection('products').updateOne({ p\_id }, { $set: obj }, (err, result) => {

if (err)

res.status(404).send({ 'update': 'error' })

else {

if (result.matchedCount != 0) {

console.log("Data updated ")

res.status(200).send({ 'update': 'success' })

}

else {

console.log("Data not updated ")

res.status(200).send({ 'update': 'Record not found' })

}

conn.close()

}

})

}

})

})

//export router

module.exports = router

\*\*\*delete.js\*\*\*

//import modules

const express = require('express')

let mongodb = require('mongodb')

//import url

let url = require('../url')

//create mongoclient

let mcl = mongodb.MongoClient

//create router instance

let router = express.Router()

//create rest api

router.delete("/", (req, res) => {

let obj = { "p\_id": req.body.p\_id }

//connect to mongodb

mcl.connect(url, (err, conn) => {

if (err)

console.log('Error in connection')

else {

let db = conn.db('nodedb')

db.collection('products').deleteOne(obj, (err, result) => {

if (err)

res.status(404).send({ 'delete': 'error' })

else {

if (result.deletedCount != 0) {

console.log("Data deleted ")

res.status(200).send({ 'delete': 'success' })

}

else {

console.log("Data not deleted ")

res.status(200).send({ 'delete': 'Record not found' })

}

conn.close()

}

})

}

})

})

//export router

module.exports = router

\*\*\*server.js\*\*\*

//import modules express body-parser cors

let express = require('express')

let bodyparser = require('body-parser')

let cors = require('cors')

//create rest object

let app = express()

//set JSON as MIME type

app.use(bodyparser.json())

//client is not sending form data -> encoding JSON

app.use(bodyparser.urlencoded({ extended: false }))

//enable CORS -> Cross Origin Resource Sharing -> communication among various ports

app.use(cors())

//create port

let port = process.env.PORT || 8080

//import fetch insert update delete modules

let fetch = require('./fetch/fetch')

let insert = require('./insert/insert')

let update = require('./update/update')

let remov = require('./delete/delete')

//use above modules

app.use('/fetch', fetch)

app.use('/insert', insert)

app.use('/update', update)

app.use('/delete', remov)

//assign port no

app.listen(port, () => {

console.log("Server listening port no:- ", port)

})

/\*

>node server

Test following URLs with postman

http://localhost:8080/fetch (get)

http://localhost:8080/insert (post)

http://localhost:8080/update (put)

http://localhost:8080/delete (delete)

body -> raw -> json

\*/

Hosting the application

1. create '.gitignore' file

>npx gitignore node

2. login to github.com and create repository

3. copy url

–copied url–-

4. initialise local repository

>git init

5. add files to repository

>git add .

6. check status

>git status

7. commit

>git commit -m "initial Commit"

8. add to remote repository

>git remote add origin –copied url–-

9. push code to repository

>git push -u origin master

Deploying nodejs on render.com

\*Login to render.com

1. goto render dashboard

2. click on new+

choose web service

3. Go down to public repository

4. paste url of github repository

5. Click on continue

6. Choose name for web service

7. leave region

8. Branch -> Master

9. root directory ./ -> path of server.js file

10. Runtime Node

11. Build command npm install

12. Start command node server

13. Click on create web service and wait

14. in command prompt of render will get port no

15. now your url on upper left part of page is ready

'<web\_service\_name>.onrender.com'

by9am.onrender.com

Deploying nodejs on cyclic.sh

\*Login cyclic.sh with github

1. Click on deploy now

2. Select Link your own

3. Search and select required repository

4. Click on connect

5. Wait to deploy it

6. after getting the message 'You're Live!'

ur url is ready

https://talented-worm-suit.cyclic.app/

Design HTML CSS JS frontend for above

\*\*\*script.js\*\*\*

let url = `https://talented-worm-suit.cyclic.app`

function LOAD() {

$.ajax({

url: url + '/fetch',

type: 'GET',

success: (posRes) => {

let x = ''

x = x + `

<table border = 1px

cellpadding = 10px

cellspacing = 10px

align = center>

<thead>

<tr>

<th>p\_id</th>

<th>p\_name</th>

<th>p\_cost</th>

</tr>

<thead>

<tbody>

`

for (let i = 0; i < posRes.length; i++) {

x = x + `

<tr>

<td>${posRes[i].p\_id}</td>

<td>${posRes[i].p\_name}</td>

<td>${posRes[i].p\_cost}</td>

</tr>

`

}

x = x + `</tbody>

</table>

`

document.getElementById("op").innerHTML = x

},

error: (errRes) => {

console.log(errRes)

}

})

}

LOAD()

//////////////////////////////////

$(document).ready(() => {

$('#getData').click((event) => {

event.preventDefault()

LOAD()

})

$('#send').click((event) => {

event.preventDefault()

let data = JSON.stringify({

"p\_id": parseInt(document.getElementById('p\_id').value),

"p\_name": document.getElementById('p\_name').value,

"p\_cost": parseInt(document.getElementById('p\_cost').value)

})

$.ajax({

url: url + "/insert",

type: 'POST',

contentType: "application/json; charset=utf-8",

dataType: "json",

data: data,

success: (posRes) => {

console.log(posRes)

LOAD()

},

error: (errRes) => {

console.log(errRes)

}

})

})

$('#update').click((event) => {

event.preventDefault()

let data = JSON.stringify({

"p\_id": parseInt(document.getElementById('p\_id').value),

"p\_name": document.getElementById('p\_name').value,

"p\_cost": parseInt(document.getElementById('p\_cost').value)

})

$.ajax({

url: url + '/update',

type: 'PUT',

contentType: "application/json; charset=utf-8",

dataType: "json",

data: data,

success: (posRes) => {

console.log(posRes)

LOAD()

},

error: (errRes) => {

console.log(errRes)

}

})

})

$('#delete').click((event) => {

event.preventDefault()

let data = JSON.stringify({

"p\_id": parseInt(document.getElementById('p\_id').value)

})

$.ajax({

url: url + "/delete",

type: 'DELETE',

contentType: "application/json; charset=utf-8",

dataType: "json",

data: data,

success: (posRes) => {

console.log(posRes)

LOAD()

},

error: (errRes) => {

console.log(errRes)

}

})

})

})

================================================================

================================================================