https://shorturl.at/kAHNV

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

**MongoDB Advanced operations**

**MySql CRUD operations**

**TokenAuthentication System**

**Connection pooling**

**Multer module**

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

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

MongoDB Advanced Operations

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

Find particular record/s ?

db.products.find(

{

$where: function () {

return (

this.p\_name == "p\_five"

)

}

}

)

Sorting mongodatabase (Note:- only display)

>db.products.find().sort({p\_name:1})

where p\_name is sort key

1 -> ascending order sort

-1 -> descending order sort

pretty() method

>db.products.find().sort({p\_name:1}).pretty()

Limiting Records

>db.products.find().sort({p\_name:-1}).limit(2)

where we can access only two records

MongoDB task

Create a mongodb database namely college

create collection Student

insert atleast 20 documents

Student Name

Branch

percentage

gender

1. Find topper of CS branch

2. Find College topper

3. Find Topper from girls

4. Find Topper from boys

5. Sort according to percentage

6. Find Top 3 students

7. Find slow learners (less percentage)

{"name": ,"branch": , "percentage": , "gender": }

Sample for insert many

>use college

>db.createCollection("students")

>db.students.insertMany([{},{},{}])

take screenshots of each result

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

MySQL CRUD operations

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

- mysql is a third party module.

- mysql module is used to interact with mysql database.

<>

config

- db\_properties.js

- db\_connection.js

fetch

- fetch.js

insert

- insert.js

update

- update.js

delete

- delete.js

- server.js

- db\_properties

- db\_connection

- fetch.js

- insert.js

- update.js

- delete.js

- server.js

- db\_properties.js file is used to maintain database properties.

- db\_connection.js file is used to create a mysql connection object.

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

- insert.js file is used to insert data in a table.

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

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

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

MySQL Queries

Download and install MySQL database

default password :- root

- Create schema

> create schema nodedb;

- switch to schema

> use nodedb;

- create table

> create table products(p\_id int, p\_name varchar(20), p\_cost int);

- insert data

> insert into products values(111, 'p\_one', 10000);

- fetch data

> select \* from products;

- update data

> update products set p\_name = 'P\_ONE', p\_cost = 11111 where p\_id = 111;

- delete data

> delete from products where p\_id = 111;

- show databases

> show databases;

- delete database

> drop schema nodedb;

Database properties

host : localhost

user : root

password : root

database : nodedb

download following modules

- express, mysql, body-parser, cors

- create server.js

- >npm init -y

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

\*\*\*db\_properties.js\*\*\*

module.exports = {

"host": "localhost",

"user": "root",

"password": "root",

"database": "nodedb"

}

\*\*\*db\_connection.js\*\*\*

//import mysql module

let mysql = require('mysql')

//import database properties

let dbproperties = require('./db\_properties')

//export the connection

module.exports = {

getConnection: () => {

return mysql.createConnection(dbproperties)

}

}

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

//import express module

let express = require('express')

//create router instance

let router = express.Router()

//import database connection

let conn = require('../config/db\_connection')

//get connection object

let connection = conn.getConnection()

//connect to database

connection.connect()

//create rest api

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

connection.query(`select \* from products`, (err, recordsArray) => {

if (err)

res.send({ 'message': 'error ' + err })

else {

console.log('Data sent')

res.send(recordsArray)

}

})

})

//export router

module.exports = router

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

//import express module

let express = require('express')

//create router instance

let router = express.Router()

//import database connection

let conn = require('../config/db\_connection')

//get connection object

let connection = conn.getConnection()

//connect to database

connection.connect()

//create rest api

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

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

let p\_name = req.body.p\_name

let p\_cost = req.body.p\_cost

connection.query(`insert into products values (${p\_id}, '${p\_name}', ${p\_cost})`,

(err, result) => {

if (err)

res.status(400).send({ 'insert': 'error ' + err })

else {

if (result.affectedRows != 0) {

console.log('Data inserted')

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

}

else {

console.log('NOT inserted')

res.status(200).send({ 'insert': 'failed' })

}

}

})

})

//export router

module.exports = router

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

//import express module

let express = require('express')

//create router instance

let router = express.Router()

//import database connection

let conn = require('../config/db\_connection')

//get connection object

let connection = conn.getConnection()

//connect to database

connection.connect()

//create rest api

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

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

let p\_name = req.body.p\_name

let p\_cost = req.body.p\_cost

connection.query(`update products set p\_name = '${p\_name}', p\_cost = ${p\_cost} where p\_id = ${p\_id}`,

(err, result) => {

if (err)

res.status(400).send({ 'update': 'error ' + err })

else {

if (result.affectedRows != 0) {

console.log('Data updated')

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

}

else {

console.log('NOT updated')

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

}

}

})

})

//export router

module.exports = router

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

//import express module

let express = require('express')

//create router instance

let router = express.Router()

//import database connection

let conn = require('../config/db\_connection')

//get connection object

let connection = conn.getConnection()

//connect to database

connection.connect()

//create rest api

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

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

connection.query(`delete from products where p\_id = ${p\_id}`,

(err, result) => {

if (err)

res.status(400).send({ 'delete': 'error ' + err })

else {

if (result.affectedRows != 0) {

console.log('Data deleted')

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

}

else {

console.log('NOT deleted')

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

}

}

})

})

//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

\*/

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

Token Based authentication system (Using MySQL)(assignment-> try with MongoDB)

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

Step 1:-

Create database for authentication system

>use nodedb;

>create table login\_details(uname varchar(20), upwd varchar(20));

>insert into login\_details values("admin","admin")

>select \* from login\_details;

database properties

host:localhost

user:root

password:root

database:nodedb

Step 2 :-

Create following directory structure

<>

config

- db\_properties.js

- db\_connection.js

token

- token.js -> to generate token

login

- login.js -> implement login module

- server.js

Setp 3:-

initialyse project and download modules

Download following modules

1)express -> create ReST api

2)mysql -> interact with MySql database

3)body-parser -> post parameters from express / set MIME type

4)jwt-simple -> for generating tokens

>npm init -y

>yarn add express mysql body-parser jwt-simple --save

\*\*\*db\_properties.js\*\*\*

module.exports = {

"host": "localhost",

"user": "root",

"password": "root",

"database": "nodedb"

}

\*\*\*db\_connection.js\*\*\*

//import mysql module

let mysql = require('mysql')

//import database properties

let dbproperties = require('./db\_properties')

//export the connection

module.exports = {

getConnection: () => {

return mysql.createConnection(dbproperties)

}

}

\*\*\*token.js\*\*\*

//import jwt-simple

let jwt = require('jwt-simple')

module.exports = (obj, enc\_key) => {

return jwt.encode(obj, enc\_key)

}

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

//import express module

let express = require('express')

//create router instance

let router = express.Router()

//import db\_connection

let conn = require('../config/db\_connection')

//get connection object

let connection = conn.getConnection()

//connect to database

connection.connect()

//import token

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

//create rest api

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

let uname = req.body.uname

let upwd = req.body.upwd

//compare with database

connection.query(`select \* from login\_details where uname = '${uname}' and upwd = '${upwd}'`,

(err, recordsArray) => {

if (recordsArray.length != 0) {

let obj = { uname, upwd }

let my\_token = token(obj, JSON.stringify(new Date()))

res.send({ 'auth': 'success', token: my\_token })

}

else

res.send({ 'auth': 'failed' })

})

})

//export router

module.exports = router

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

//import modules

let express = require('express')

let bodyparser = require('body-parser')

//create rest object

let app = express()

//set JSON as MIME type

app.use(bodyparser.json())

//client encoding form data to json

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

//import login module

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

//use login module

app.use('/login', login)

//assign port no

app.listen(8080)

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

/\*

>node server

http://localhost:8080/login

POST

body -> raw -> JSON

\*/

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

Connection pooling:-

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

- we can get mysql connection objects through the pool.

- pool contains a set of predefined connection objects.

- If we get connection object from the pool automatically application performance will be improved.

- 'createPool()' is a function in mysql object used to create a pool of connections.

- connectionLimit is a key used to limit connections.

//initialyse project

//>npm init -y

//download express and mysql modules

//>yarn add express mysql --save

//import modules

let express = require('express')

let mysql = require('mysql')

//create rest object

let app = express()

//create database properties

let db\_properties = {

"host": "localhost",

"user": "root",

"password": "root",

"database": "nodedb",

"connectionLimit": 100

}

//create pool

let pool = mysql.createPool(db\_properties)

//create restapi

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

pool.getConnection((err, connection) => {

if (err)

console.log('Error in connection')

else {

connection.query(`select \* from products`, (err, recordsArray) => {

if (err)

connection.log('Error in db query:- ' + err)

else

res.send(recordsArray)

//release the connection

connection.release()

})

}

})

})

//assign port no

app.listen(8080)

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

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

Uploading Images(Files)

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

Multer module

//inialyse project

//>npm init -y

//download multer and express modules

//>yarn add express multer --save

//import modules

let express = require('express')

let multer = require('multer')

//create rest object

let app = express()

//create storage

let storage = multer.diskStorage({

destination: (req, res, cb) => {

cb(null, 'uploads/')

},

filename: (req, file, cb) => {

cb(null, file.fieldname + "-" + Date.now() + ".jpg")

}

})

//let upload = multer({ storage }).single('profileImage')

let upload = multer({ storage }).array('profileImage', 2)

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

upload(req, res, (err) => {

if (err)

res.json({ 'upload': 'err ' + err })

else

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

})

})

//assign port no

app.listen(8080)

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

/\*

Create folder namely uploads in current path

test url http://localhost:8080/profile

post request

body -> form data -> key file hover left part then down arrow will be visible

\*/

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

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