https://shorturl.at/3lCHe 11 Jun 2024

==

Reading Express Post parameters

==

Download express and body-parser modules

```
//initialise project
//>npm init -y
//Download express and body-parser modules
//>yarn add express body-parser --save
//import modules
let express = require('express')
let bodyparser = require('body-parser')
//create rest object
let app = express()
//create port
let port = 8080
//set JSON as MIME type
app.use(bodyparser.json())
//front end encoding form data as json
app.use(bodyparser.urlencoded({ extended: false }))
//create rest api
app.post("/login",(req,res)=>{
    //client parameters are stored in body part of req
    let uname = req.body.uname
    let upwd = req.body.upwd
    if (uname == 'admin' && upwd == 'admin')
        res.send({ 'login': "success" })
    else
        res.send({ 'login': 'failed' })
})
//assign port no
app.listen(port, () => {
    console.log('Server listening port no ', port)
})
/*
    Start server
    >node server
    Test Rest api in postman with url
    http://localhost:8080/login
    in postman
        1.  req     -> post
            body    -> raw
                    -> text <-> json
                    {"uname":"admin","upwd":"admin"}

        2.  req     -> post
            body    -> x-www...
                    -> key and values
*/
```

==
Reading url parameters from express
==

```
//initilayse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//create port
let port = 8080
//url:http://localhost:8080/login/admin/admin
app.get("/login/:uname/:upwd", (req, res) => {
    //params is the predefined key from req object used to read prameters from
url
    let uname = req.params.uname
    let upwd = req.params.upwd
    console.log("req.params", req.params)
    if (uname === 'admin' && upwd === 'admin')
        res.json({ 'login': 'success' })
    else
        res.json({ 'login': 'failed' })
})
//assign port no
app.listen(port, () => {
    console.log(`Server listening port no ${port}`)
})
```

==
Modules in Express

===
<>
        login
                - login.js
        logout
                - logout.js
        - server.js

//initialyse project
//>npm init -y
//download express module
//>yarn add express --save

login.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 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)
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
*/
```

==
Imports and exports
==

- module is the predefined object in node.
- exports is the predefined key in module object.
- exports key is used to export (JSON object or function)

Exporting and importing JSON object
==
Eg01

<>
        config
                - db_config.js
        - server.js

db_config.js
```
module.exports = {
    "host": "localhost",
    "user": "root",
    "password": "root",
    "database": "nodedb",
    "table": "products"
}
```

server.js
```
//initialyse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//create port
let port = 8080
//import db_config
let obj = require("./config/db_config")
//create rest api
app.get("/", (req, res) => {
    res.json(obj)
})
//assign port no
app.listen(port, () => {
    console.log(`Server listening port no ${port}`)
})
```

Function import and export
===
Eg02
<>
  config
     - my_fun.js
  - server.js

my_fun.js
```
module.exports = (arg1, arg2) => {
    if (arg1 === 'admin' && arg2 === 'admin')
        return "Login Success"
    else
        return "Login Failed"
}
```

server.js
```
//initialyse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//import my_fun
let my_fun = require('./config/my_fun')
//create rest api
app.get("/login", (req, res) => {
    let uname = req.query.uname
    let upwd = req.query.upwd
    res.send(my_fun(uname, upwd))
})
//assign port no
app.listen(8080)
console.log("Server listening port no 8080")

//url :- http://localhost:8080/login/?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.

==
Type of databases
==

SQL	Structured Query Language
NoSQL	Not Structured Query Language

==
Interaction with MongoDB
==

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

initialise project
>npm init

download modules
- express
- mongodb@4.11
- body-parser
- cors

>yarn add express mongodb@4.11 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
const url = require('../url')
//create mongo client
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')
        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.json(array)
                    conn.close()
                }
            })
        }
    })
})

//export router
module.exports = router
```

insert.js

```
//import modules
const express = require('express')
let mongodb = require('mongodb')
//import url
const url = require('../url')
//create mongo client
let mcl = mongodb.MongoClient
//create router instance
let router = express.Router()
//create rest api
router.post("/", (req, res) => {
    let obj = req.body
    //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').insertOne(obj, (err) => {
                if (err)
                    res.json({ 'insert': 'Error ' + err })
                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
const url = require('../url')
//create mongo client
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_costs
    }
    //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').updateOne({ p_id }, { $set: obj }, (err,
result) => {
                if (err)
                    res.json({ 'update': 'Error ' + err })
                else{
                    if (result.matchedCount != 0) {
                        console.log("Data updated ")
                        res.json({ 'update': 'success' })
                    } else {
                        console.log("Data Not updated ")
                        res.json({ '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
const url = require('../url')
//create mongo client
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:- ', err)
        else {
            let db = conn.db("nodedb")
            db.collection('products').deleteOne(obj, (err, result) => {
                if (err)
                    res.json({ 'delete': 'Error ' + err })
                else {
                    if (result.deletedCount != 0) {
                        console.log("Data deleted ")
                        res.json({ 'delete': 'success' })
                    } else {
                        console.log("Data Not deleted ")
                        res.json({ '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 Origine 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

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 ur url on upper left part of page is ready
'<web_service_name>.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

==
Token Based authentication system
==
Step 1:-
    Create database for authentication system

```
>use nodedb;
>db.createCollection("users")
>db.users.insert({"uname":"admin","upwd":"admin"})
>db.users.find()
```

Step 2 :-
    Create following directory structure

```
<>
    token
        - token.js          -> to generate token
    login
        - login.js          -> implement login module
    - server.js
```

Setp 3:-
    initialise project and download modules
    Download following modules
    1)express   -> create ReST api
    2)mongodb@4.11   -> interact with mongo database
    3)body-parser   -> post parameters from express / set
MIME type
    4)jwt-simple   -> for generating tokens

```
>npm init -y
>yarn add express mongodb@4.11 body-parser jwt-simple --save
```

url.js

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

token.js

```
//import jwt-simple
let jwt = require("jwt-simple")
module.exports = (obj, enc_key) => {
    return jwt.encode(obj, enc_key)
}
```

login.js

```
//import token
let token = require('../token/token')
//import modules
const express = require('express')
let mongodb = require('mongodb')
//import url
const url = require('../url')
//create mongo client
let mcl = mongodb.MongoClient
//create router instance
let router = express.Router()
//create rest api
router.post("/", (req, res) => {
    //connect to mongodb
    let obj = req.body
    mcl.connect(url, (err, conn) => {
        if (err)
            console.log('Error in connection')
        else {
            let db = conn.db('nodedb')
            db.collection('users').find(obj).toArray((err, array) => {
                if (err)
                    console.log('Error:- ', err)
                else {
                    if (array.length > 0) {
                        console.log('Auth Success')
                        let myToken = token(obj, new Date().toString())
                        res.json({ 'auth': 'success', token: myToken })

                    }
                    else {
                        console.log('Auth Failed')
                        res.json({ 'auth': 'falied'})
                    }
                    conn.close()
                }
            })
        }
    })
})

//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
*/
```

==
Mongoose
==
- Mongoose is an object document modelling (ODM) that sits on top of Node's MongoDB driver.
- It's similar to an ORM for a relational database.
- first stable version 1.0.1 02-02-2011

Create a folder MongooseEg
drag to vscode
create server.js
initialise application
>npm init -y
install express body-parser cors mongoose
>yarn add express body-parser cors mongoose --save

Directory Structure
<>
    apis
        - productApis.js
    model
        - Product.js => STRICTLY FIRST LETTER CAPITAL AND NO 'S' AT END
    routes
        - productRoutes.js
- url.js
- server.js

url.js*

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

in server.js
connect to mongodb as
        mongoose.connect

server.js

```
//import modules
const express = require('express')
const bodyparser = require('body-parser')
const cors = require('cors')
const mongoose = require('mongoose')
//import url
let url = require('./url')
//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 Origine Resource Sharing -> communication among various
ports
app.use(cors())
//connect to mongodb
mongoose.connect(url, { dbName: "newDb" })
    .then(() => {
        console.log('Connection Success')
    }, (errRes) => {
        console.log("Connection faild:- ", errRes)
    })

//create port
```

```
let port = 8080
//assign port no
app.listen(port, () => {
    console.log('Server listening port no:- ', port)
})
```

Test the connection here
>node server

Now create mongoose schema as
model/Product.js
Product.js

```
//import mongoose
const mongoose = require('mongoose')
//create schema
const productSchema = new mongoose.Schema({
    p_id: Number,
    p_name: String,
    p_cost: Number
})
module.exports = mongoose.model("Product", productSchema)
```

Create apis
apis/productApis.js
productApis.js

```
//import db schema
const Product = require('../model/Product')
//get all products
const products_all = async (req, res) => {
    try {
        const products = await Product.find()
        console.log('Data sent')
        res.json(products)
    }
    catch (error) {
        console.log('Fetch error :- ', error)
        res.json({ 'message': error })
    }
}
module.exports = {
    products_all
}
```

Define routes in
routes/productRoutes.js
productRoutes.js

```
//import express module
const express = require('express')
//create router instance
const router = express.Router()
//import productApi
const productApi = require('../apis/productApis')
//fetch all records
router.get("/fetch", productApi.products_all)
//export router
module.exports = router
```

import and use routes in server.js and test
fetch api in postman
        http://localhost:8080/fetch
after that perform remaining
server.js

```
//import modules
const express = require('express')
const bodyparser = require('body-parser')
const cors = require('cors')
const mongoose = require('mongoose')
//import url
let url = require('./url')
//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 Origine Resource Sharing -> communication among various
ports
app.use(cors())
//connect to mongodb
mongoose.connect(url, { dbName: "newDb" })
    .then(() => {
        console.log('Connection Success')
    }, (errRes) => {
        console.log("Connection faild:- ", errRes)
    })
```

```
//////////////////////////////////////////////////////////
//import routes
const productRoutes = require('./routes/productRoutes')
//use routes
app.use("/", productRoutes)
//////////////////////////////////////////////////////////
//create port
let port = 8080
//assign port no
app.listen(port, () => {
    console.log('Server listening port no:- ', port)
})
```

>>
now update remaining files
productApis.js

```
//import db schema
const Product = require('../model/Product')
//get all products
const products_all = async (req, res) => {
    try {
        const products = await Product.find()
        console.log('Data sent')
        res.json(products)
    }
    catch (error) {
        console.log('Fetch error :- ', error)
        res.json({ 'message': error })
    }
}
//insert a product
const insert_product = async (req, res) => {
    const product = new Product({
        p_id: req.body.p_id,
        p_name: req.body.p_name,
        p_cost: req.body.p_cost
    })
    try {
        const savedProduct = await product.save()
        console.log('Product inserted')
        res.send(savedProduct)
    }
    catch (error) {
        res.status(400).send(error)
    }
}
//update product
const update_product = async (req, res) => {
    let p_id = req.body.p_id
    const product = {
        p_name: req.body.p_name,
        p_cost: req.body.p_cost
    }
    try {
        const updateProduct = await Product.updateOne(
            { p_id }, product
        )
        if (updateProduct.modifiedCount != 0) {
            console.log('Product Updated', updateProduct)
            res.send({ 'update': 'success' })
        }
        else {
            console.log('Product not updated')
            res.send({ 'update': 'Record Not Found' })
        }
    }
    catch (error) {
        res.status(400).send(error)
    }
}

//delete product
const delete_product = async (req, res) => {
    let p_id = req.body.p_id
    try {
        const deletedproduct = await Product.deleteOne({ p_id })
        if (deletedproduct.deletedCount != 0) {
            console.log('Product Deleted')
            res.send({ 'delete': 'success' })
        }
        else {
            console.log('Product Not deleted')
            res.send({ 'delete': 'Record Not Found' })
        }
    }
    catch (error) {
        res.status(400).send(error)
    }
}
module.exports = {
    products_all,
    insert_product,
    update_product,
    delete_product
}
```

prodctRoutes.js

```
//import express module
const express = require('express')
//create router instance
const router = express.Router()
//import productApi
const productApi = require('../apis/productApis')
//fetch all records
router.get("/fetch", productApi.products_all)
//insert a record
router.post("/insert", productApi.insert_product)
//update a record
router.put("/update", productApi.update_product)
//delete a record
router.delete("/delete", productApi.delete_product)
//export router
module.exports = router
```

Perform required modifications in above code
Deploy on cyclic and render