C – HTTP- PUT

R – HTTP GET  
U – HTTP Update

D- HTTP Delete

REST – Architectural Pattern for developing Webservices, Communicates over HTTP so it dependent on HTTP methods like GET, POST, PUT and DELETE

URL - Abc.com/api/contacts

MERN

M- MongoDB – nosql database

E- Express – Server-side web framework, based on nodeJs we have express –to build webservices & webapps

R-React

N-Node

**USING NODEJS –nodejs.org**

Node js is a runtime env for Javascript- Google V8 Javascript engine.

Using Node we can execute Javascript in Serverside using Express

*Node JS is modular based Development*

*Module.exports defines a node module otherwise it is a normal JS file, This enables to expose modules from 1 file to another file in a different module,*

*In common.js we have a function called as require function , Now we have to give the name of the module. In common.js we give the dependency*

*Demo .js*

let t= require("./first");

//whatever value is exported the require function return

console.log("Day-4 - Node Example is ::" +t);

*first.js*

module.exports = "Wells fargo, Hyderabad"

*greetModule.*

function greet(){

console.log ("From Greet Function")

}

module.exports = greet

*demo1.js*

let t= require("./first");

let r= require("./greetmodule"); // r contains the function definition.

//whatever value is exported the require function return

console.log("Day-4 - Node Example is ");

console.log(t);

console.log(r);

r();

*demo2 function within function*

module.exports ={

greet:function(){

console.log("Function 1");

},

test:function(){

console.log("Test Function");

}

}

*Node provides few modules – http(creates server),fs (io operations),url,querystring etc.*

*We can create our own webserver.*

*In HTTP module there is a method called Create Server which can handle request & Response as params.*

*We also have request headers as in servlet api.*

*Response Is flushed and completed using the end() method.*

*Execute server with the method by calling Listen() method*

*Eg- Create Server-*

let http = require("http");

let server = http.createServer((request,response) =>{

response.end("Welcome to Node Server");

})

server.listen(9004);

*Http – WriteHead() – Response to include the Information*

*If {Request.url } == “/”{*

*//What needs to be done, what content to be given*

*Res.WriteHead method to be used (200,(‘content- type’, ‘text/html’)*

*<<>html code>>*

*Res.end(); -- Mandatory*

*}*

let http = require("http");

let server = http.createServer((request,response) =>{

//response.end("Welcome to Node Server");

//Check the Incoming Request URL

if(request.url == "/"){

//specify the response Headers

response.writeHead(200,{"Content-Type":'text/html'})

//send html conent

response.write("<html><body><h2>Welcome Home</h2></body> </html>")

response.end();

}

else if(request.url == "/admin"){

//specify the response Headers

response.writeHead(200,{"Content-Type":'text/html'})

//send html conent

response.write("<html><body><h2>Welcome Home Admin</h2></body> </html>")

response.end();

}

else{

response.writeHead(200,{"Content-Type":'text/html'})

//send html conent

response.write("<html><body><h2>Invalid URL</h2></body> </html>")

response.end();

}

})

server.listen(9004,"localhost", () =>{

console.log("From Server");

})

*RESPONDING JSON DATA*

*Example of code Snipped from Myserver.js*

else if (request.url == "/data"){

//specify the response Headers

response.writeHead(200,{"Content-Type":'application/json'})

//send html conent

response.write(JSON.stringify({"message":"Wellsfargo"}))

response.end();

}

*Full code of Myserver.js*

let http = require("http");

let server = http.createServer((request,response) =>{

//response.end("Welcome to Node Server");

//Check the Incoming Request URL

if(request.url == "/"){

//specify the response Headers

response.writeHead(200,{"Content-Type":'text/html'})

//send html conent

response.write("<html><body><h2>Welcome Home</h2></body> </html>")

response.end();

}

else if(request.url == "/admin"){

//specify the response Headers

response.writeHead(200,{"Content-Type":'text/html'})

//send html conent

response.write("<html><body><h2>Welcome Home Admin</h2></body> </html>")

response.end();

}

else if (request.url == "/data"){

//specify the response Headers

response.writeHead(200,{"Content-Type":'application/json'})

//send html conent

response.write(JSON.stringify({"message":"Wellsfargo"}))

response.end();

}

else{

response.writeHead(200,{"Content-Type":'text/html'})

//send html conent

response.write("<html><body><h2>Invalid URL</h2></body> </html>")

response.end();

}

})

server.listen(9004,"localhost", () =>{

console.log("From Server");

})

EXPRESS.JS – example – server2.js

Written in javascript, Based on Node.js fro creating serverside webapplications :

Web Pages

Or API we can use Express JS

Diff Webserver Vs. Node Server

let express =require('express')

//Will internally create a server whenever we call the express() function

let app = express()

app.listen(9005, () =>{

console.log ("from Express webserver in 5001");

})

The app object contains the framework of express as per above code snippet – GET POST & DELETE

Server2.js

let express =require('express')

//Will internally create a server whenever we call the express() function

let app = express()

app.get("/api/wsproducts",(request,response) =>{

//response.send("Helloe")

response.send("My Products Webservice")

})

app.listen(9005, () =>{

console.log ("from Express webserver in 9005");

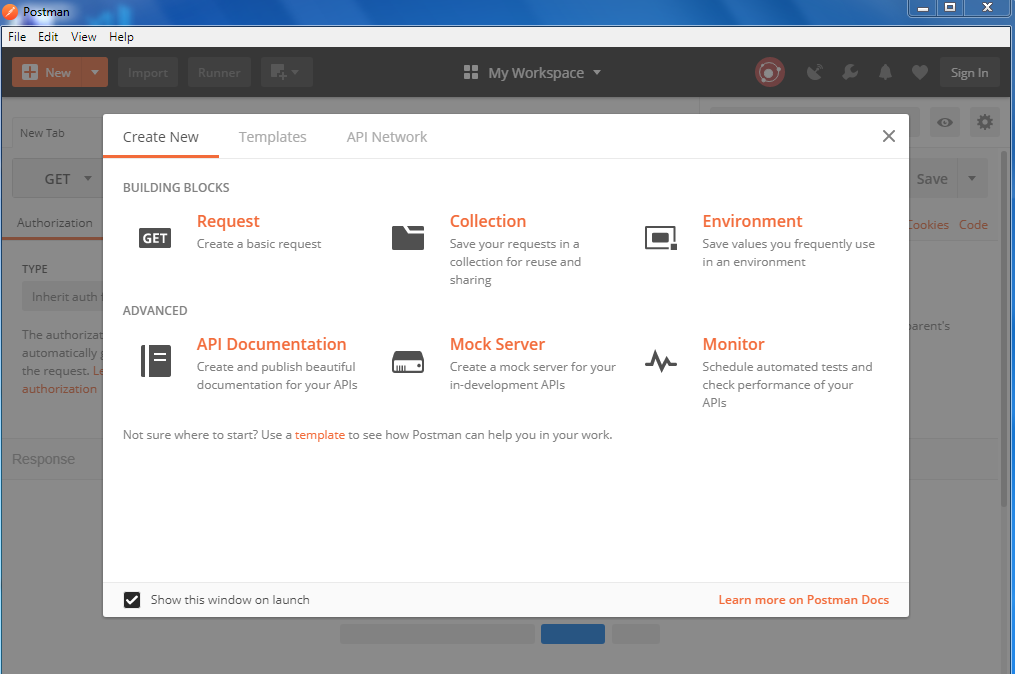
})

Use Rest client Software to test the Webservice

Install REST CLIENT

POSTMAN (get postman.com)

Or Advanced Rest Client



Retrieve data from data.js from server2.js

let data =[

{"id":1,"name": "Product 1", price:3000},

{"id":2,"name": "Product 2", price:4000},

{"id":3,"name": "Product 3", price:5000},

{"id":4,"name": "Product 4", price:6000}

]

module.exports=data

let express =require('express')

//Will internally create a server whenever we call the express() function

let app = express()

let products = require('./data')

app.get("/api/wsproducts",(request,response) =>{

//response.send("Helloe")

//response.send("My Products Wbservice")

response.json(products);

})

app.listen(9005, () =>{

console.log ("from Express webserver in 9005");

})

Find By ID

In The URL *– api/wsproducts/3 (passing the param to the path)*

*Syntax is modified as : 3 – param refererd as “:” , after which we need to give the name of the parameter.*

let express =require('express')

//Will internally create a server whenever we call the express() function

let app = express()

let products = require('./data')

app.get("/api/wsproducts",(request,response) =>{

//response.send("Helloe")

//response.send("My Products Wbservice")

response.json(products);

})

app.get("/api/wsproducts/:id",(request,response) =>{

let requestid= request.params.id;

let product = products.filter(product => product.id == requestid)

//console.log(product)

//response.send("Helloe")

//response.send("My Products Wbservice")

if(product.length ==0) {

response.json("Product "+ requestid + " Not found")

}

response.json(product);

})

app.listen(9005, () =>{

console.log ("from Express webserver in 9005");

})

*INSERTING DATA*

Express JS Middlewares 🡪 Pluggable processors

M/W can be processed in a serial fashion

M/W can aslo alter the request

To Activate middleware we have a predefined function called “use” to Plug the M/W

let express =require('express')

let bodyparser = require('body-parser')

//Will internally create a server whenever we call the express() function

let app = express()

let products = require('./data')

app.use(bodyparser.json())

app.get("/api/wsproducts",(request,response) =>{

//response.send("Helloe")

//response.send("My Products Wbservice")

response.json(products);

})

app.get("/api/wsproducts/:id",(request,response) =>{

let requestid= request.params.id;

let product = products.filter(product => product.id == requestid)

//console.log(product)

//response.send("Helloe")

//response.send("My Products Wbservice")

if(product.length ==0) {

response.json("Product "+ requestid + " Not found")

}

response.json(product);

})

app.post("/api/wsproducts",(request,response) =>{

let newproduct ={

"id":products.length+1,

"name":request.body.name,

"price":request.body.price,

"description":request.body.description

}

products.push(newproduct)

response.json(newproduct)

})

app.listen(9005, () =>{

console.log ("from Express webserver in 9005");

})

MONGO DB

Relational & No Sql Database

NO SQL DB – Instead of sql statements we make use of Functions on The NOSQL Database

Popular one is the Mongo DB

Community & Commercial version

Works based on concepts called Collections & Documents

Each collection consists of Documents

Document is a json object that stores key value pair

Document is a JSonObject like a Row in a database

Group of collection makes a database

Collections = table

Document =record

For each document \_id is mandatory represents like a Primary Key

**In a command Prompt:**

mongod –version

Mongod –dbpath data [// Starts the Mongodb]

From another command Prompt

Enter “mongo” 🡪 Starts the client

Exit 🡪 come out of DB

Show databases

Db –list the currenty database

Use myshoppingdb

Show collections

Create User

Db.users.insert({ name: ‘Chris’})

Db.user.find()

"\_id" : ObjectId("5b18f059ec53fa9a026de2f1"), "name" : "Chris" }

db.users.insert[{name: 'john'},{name:'Sheila'}]

show collections

sers

**db.users.find()**

"\_id" : ObjectId("5b18f059ec53fa9a026de2f1"), "name" : "Chris" }

**db.users.insert([{name: 'john'},{name:'Sheila'}])**

ulkWriteResult({

"writeErrors" : [ ],

"writeConcernErrors" : [ ],

"nInserted" : 2,

"nUpserted" : 0,

"nMatched" : 0,

"nModified" : 0,

"nRemoved" : 0,

"upserted" : [ ]

)

db.users.find()

"\_id" : ObjectId("5b18f059ec53fa9a026de2f1"), "name" : "Chris" }

"\_id" : ObjectId("5b18f15fec53fa9a026de2f2"), "name" : "john" }

"\_id" : ObjectId("5b18f15fec53fa9a026de2f3"), "name" : "Sheila" }

db.users.find({'name'})

018-06-07T14:19:07.812+0530 E QUERY [thread1] SyntaxError: missing : after p

operty id @(shell):1:21

**db.users.find({'name':'john'})**

"\_id" : ObjectId("5b18f15fec53fa9a026de2f2"), "name" : "john" }

**> db.users.update({name:'Chris'},{name:'Gayle'})**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.users.find({'name':'john'})

{ "\_id" : ObjectId("5b18f15fec53fa9a026de2f2"), "name" : "john" }

> db.users.find()

{ "\_id" : ObjectId("5b18f059ec53fa9a026de2f1"), "name" : "Gayle" }

{ "\_id" : ObjectId("5b18f15fec53fa9a026de2f2"), "name" : "john" }

{ "\_id" : ObjectId("5b18f15fec53fa9a026de2f3"), "name" : "Sheila" }

>

> db.users.find()

{ "\_id" : ObjectId("5b18f059ec53fa9a026de2f1"), "name" : "Gayle" }

{ "\_id" : ObjectId("5b18f15fec53fa9a026de2f2"), "name" : "john" }

{ "\_id" : ObjectId("5b18f15fec53fa9a026de2f3"), "name" : "Sheila" }

**> db.users.update({name:'Gayle'},{name:'ChrisGayle'})**

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.users.find()

{ "\_id" : ObjectId("5b18f059ec53fa9a026de2f1"), "name" : "ChrisGayle" }

{ "\_id" : ObjectId("5b18f15fec53fa9a026de2f2"), "name" : "john" }

{ "\_id" : ObjectId("5b18f15fec53fa9a026de2f3"), "name" : "Sheila" }

>

Building the REST Service API

Collection Name – cln\_products in myshoppingdb

db.cln\_products.find()

"\_id" : ObjectId("5b18f388ec53fa9a026de2f4"), "name" : "Sony Bravia", "price"

70000 }

"\_id" : ObjectId("5b18f3efec53fa9a026de2f5"), "name" : "Sony", "price" : 17000

}

"\_id" : ObjectId("5b18f3efec53fa9a026de2f6"), "name" : "Apple", "price" : 8000

0 }

"\_id" : ObjectId("5b18f423ec53fa9a026de2f7"), "name" : "Sony LCD", "price" : 7

000 }

"\_id" : ObjectId("5b18f423ec53fa9a026de2f8"), "name" : "Apple Watch", "price"

40000 }

db.cln\_products.update({"\_id" : ObjectId("5b18f388ec53fa9a026de2f4")},{$set:{name:"Samsung Galaxy S"}})

db.cln\_products.update({"\_id":Object ID Value”)},{$set:{name:"Value"}}})

db.cln\_products.update({"\_id" : ObjectId("5b18f388ec53fa9a026de2f4")},{$set:{name:"Samsung Galaxy S"}})> db.cln\_products.find()

{ "\_id" : ObjectId("5b18f388ec53fa9a026de2f4"), "name" : "Samsung Galaxy S", "pr

ice" : 70000

}

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f5"), "name" : "Sony", "price" : 17000

0 }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f6"), "name" : "Apple", "price" : 8000

00 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f7"), "name" : "Sony LCD", "price" : 7

7000 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f8"), "name" : "Apple Watch", "price"

: 40000 }

>

Add new field to existing document

db.cln\_products.update({"\_id" : ObjectId("5b18f388ec53fa9a026de2f4")},{$set:{description:"Samsung Smart Phone"}})

CRUD UPDATE

“Upsert true” update if there is a document else create

From the Notepad::

{

"name":"product5",

"price":7000,

"description":"New One Added"

}

db.cln\_products.update({"\_id":Object ID Value”)},{$set:{name:"Value"}}})

db.cln\_products.update({"\_id" : ObjectId("5b18f388ec53fa9a026de2f4")},{$set:{name:"Samsung Galaxy S"}})

> db.cln\_products.find()

{ "\_id" : ObjectId("5b18f388ec53fa9a026de2f4"), "name" : "Samsung Galaxy S", "pr

ice" : 70000 }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f5"), "name" : "Sony", "price" : 17000

0 }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f6"), "name" : "Apple", "price" : 8000

00 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f7"), "name" : "Sony LCD", "price" : 7

7000 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f8"), "name" : "Apple Watch", "price"

: 40000 }

>

Add new field to existing document

db.cln\_products.update({"\_id" : ObjectId("5b18f388ec53fa9a026de2f4")},{$set:{description:"Samsung Smart Phone"}})

db.cln\_products.update({"\_id":Object ID Value”)},{$set:{name:"Value"}}})

db.cln\_products.update({"\_id" : ObjectId("5b18f3efec53fa9a026de235")},{$set:{description:"Refigerator"}},{upsert:true})

db.cln\_products.update({"\_id" : ObjectId("5b18f3efec53fa9a026de235")},{$set:{description:"Whirlpool"}})

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

> db.cln\_products.find()

{ "\_id" : ObjectId("5b18f388ec53fa9a026de2f4"), "name" : "Samsung Galaxy S", "pr

ice" : 70000, "description" : "Samsung Smart Phone" }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f5"), "name" : "Sony", "price" : 17000

0 }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f6"), "name" : "Apple", "price" : 8000

00 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f7"), "name" : "Sony LCD", "price" : 7

7000 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f8"), "name" : "Apple Watch", "price"

: 40000 }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de235"), "description" : "Refigerator" }

> db.cln\_products.update({"\_id" : ObjectId("5b18f3efec53fa9a026de235")},{$set:{d

escription:"Whirlpool"}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.cln\_products.find()

{ "\_id" : ObjectId("5b18f388ec53fa9a026de2f4"), "name" : "Samsung Galaxy S", "pr

ice" : 70000, "description" : "Samsung Smart Phone" }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f5"), "name" : "Sony", "price" : 17000

0 }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de2f6"), "name" : "Apple", "price" : 8000

00 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f7"), "name" : "Sony LCD", "price" : 7

7000 }

{ "\_id" : ObjectId("5b18f423ec53fa9a026de2f8"), "name" : "Apple Watch", "price"

: 40000 }

{ "\_id" : ObjectId("5b18f3efec53fa9a026de235"), "description" : "Whirlpool" }

>

db.cln\_products.remove({"\_id" : ObjectId("5b18f3efec53fa9a026de235")},{$set:{description:"Whirlpool"}})

db.cln\_products.update({"\_id" : ObjectId("5b18f3efec53fa9a026de235")},{$set:{d

scription:"Whirlpool"}})

riteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

db.cln\_products.find()

"\_id" : ObjectId("5b18f388ec53fa9a026de2f4"), "name" : "Samsung Galaxy S", "pr

ce" : 70000, "description" : "Samsung Smart Phone" }

"\_id" : ObjectId("5b18f3efec53fa9a026de2f5"), "name" : "Sony", "price" : 17000

}

"\_id" : ObjectId("5b18f3efec53fa9a026de2f6"), "name" : "Apple", "price" : 8000

0 }

"\_id" : ObjectId("5b18f423ec53fa9a026de2f7"), "name" : "Sony LCD", "price" : 7

000 }

"\_id" : ObjectId("5b18f423ec53fa9a026de2f8"), "name" : "Apple Watch", "price"

40000 }

"\_id" : ObjectId("5b18f3efec53fa9a026de235"), "description" : "Whirlpool" }

**db.cln\_products.remove({"\_id" : ObjectId("5b18f3efec53fa9a026de235")},{$set:{description:"Whirlpool"}})**

018-06-07T14:47:08.306+0530 E QUERY [thread1] SyntaxError: missing : after p

operty id @(shell):2:0

**db.cln\_products.remove({"\_id" : ObjectId("5b18f3efec53fa9a026de235")},{$set:{d**

**scription:"Whirlpool"}})**

riteResult({ "nRemoved" : 1 })

db.cln\_products.find()

"\_id" : ObjectId("5b18f388ec53fa9a026de2f4"), "name" : "Samsung Galaxy S", "pr

ce" : 70000, "description" : "Samsung Smart Phone" }

"\_id" : ObjectId("5b18f3efec53fa9a026de2f5"), "name" : "Sony", "price" : 17000

}

"\_id" : ObjectId("5b18f3efec53fa9a026de2f6"), "name" : "Apple", "price" : 8000

0 }

"\_id" : ObjectId("5b18f423ec53fa9a026de2f7"), "name" : "Sony LCD", "price" : 7

000 }

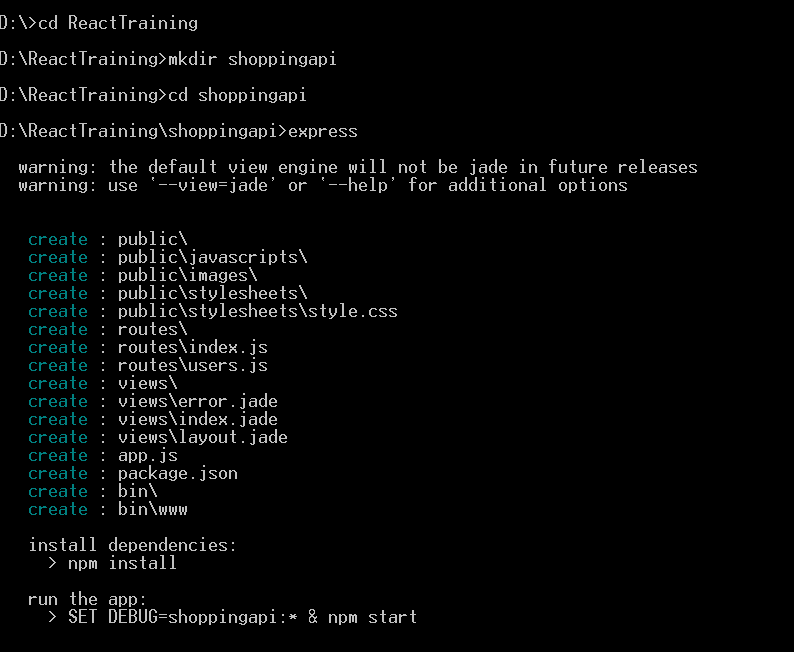
"\_id" : ObjectId("5b18f423ec53fa9a026de2f8"), "name" : "Apple Watch", "price"

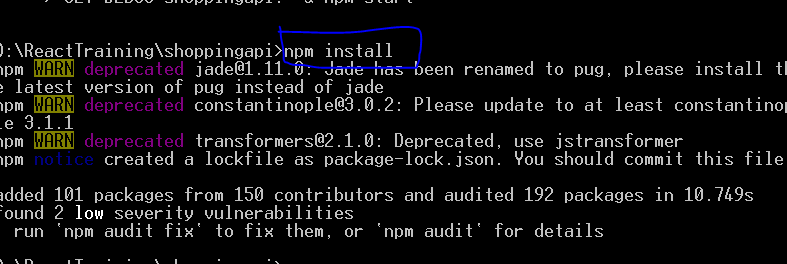
40000 }

CREATE REST API

Express –version

Or npm install express-generator –g





Comment 4 line in app.js

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

//app.set('view engine', 'jade');

//app.use('/', indexRouter);

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

MongoDB Drivers – mongoose (drivers for MongoDB)

npm install -S mongoose

Define Models – and Schema

Create folder – models

& require mongoose

POST -= EXRPRESS & MONGO

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

var express = require('express');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

var indexRouter = require('./routes/index');

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

var mongoose = require("mongoose")

var app = express();

// view engine setup

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

//app.set('view engine', 'jade');

app.use(logger('dev'));

app.use(express.json());

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

app.use(cookieParser());

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

//app.use('/', indexRouter);

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

mongoose.connect("mongodb://localhost:27017/myshoppingdb")

console.log("Mongodb connected");

var products = require("./model/product")// file name

app.get("/wsproducts", function (request, response) {

products.find(function (err, data) {

if (err) {

throw err

}

response.json(data)

})

})

app.post("/wsproducts", function (request, response) {

let newproduct=request.body;

products.create(newproduct,(err,data) => {

if (err) {

throw err

}

response.json(data)

})

})

app.get("\*", (request, response) => {

response.send("MyShopping Api");

})

// catch 404 and forward to error handler

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

next(createError(404));

});

// error handler

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

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

module.exports = app;

**Product.js**

const mongoose=require("mongoose")

var productSchema =mongoose.Schema({

name:String,

price:Number

})

module.exports=mongoose.model("cln\_products",productSchema)