Phase 3

Day 1

20-11-2021

Node JS

Core

Console

Process

Fs module

http module

User-defined Module

External Module

Express JS Module

Create Web Application

Creating REST API

Socket IO networking programming

Mongo DB Database

All types of queries may basic and Adv query

Connecting mongo db database using

Mongo DB Module

Mongoose module

Developing Node JS Application using MVC Style

MEAN Stack

From Angular to Express JS Node JS Application and Connecting to Mongo DB Database.

Node JS

HTML, CSS and JavaScript

Read , Write and update DOM (Document Object Model).

Any html tag code

JavaScript library or Framework

jQuery : it is a library

Angular Framework :

React JS with Redux and Flux

Vue JS

Coffee JS

Backbone JS

Ext JS

Node JS : Node JS is not a framework and library. Node JS is run time environment for JavaScript library and Framework.

In Java JRE

In JavaScript Node JS

Before Node JS

Front end technologies backend technologies

HTML,CSS,JavaScript using ES5 style Java : JEE

Bootstrap and jQuery Spring boot

Asp.net

Php

Python

Node JS

Before Node JS JavaScript is known as Client Side scripting language But After node JS JavaScript also known as Client side as well as Server Side Scripting language.

Using JavaScript ie Node JS we can do file handling, we can create web application, we can create Rest web service, we can connect any database it mongo db or RDBMS, we can do networking programming .

Node JS provided lot of pre-defined modules to run those module we required run time environment ie node JS.

In node JS we can’t use BOM and DOM.

Browser Object Model and Document Object Model.

Client side JavaScript follow object hierarchy concept

Object

Property

Behaviour

Object

Property

Behaviour

Object

Property

Behaviour

Object

document.write(“Welcome”);

window.document.write(“Welcome”)

Client Side Scripting language : In this scripting language we can use BOM and DOM but we can’t do file handling program, can’t connect database or can’t create REST API.

Server Side Scripting language : We can’t use DOM and BOM.

REPL Terminal : Read Eval Print Loop

To display the output using Node JS

Node JS provide pre-defined global object ie

console.

Synchronous communication

Statement level

L1

L2

L3

Function call

dis1()

dis2()

dis3();

1 req

2 req

3 req

Client Server

Asynchronous communication

Statement level

L1

L2

L3

Function call

dis1()

dis2()

dis3();

1 req

2 req

3 req

Client Server

Ajax Asynchronous JavaScript and XML

XMLHttpRequest or ActiveXObject

Callback : passing the function name or function body or function itself to another function as a parameter is known as callback function. Callback function can be synchronous or asynchronous.

Node JS goal is to provide an easy way to build scalable network program.

JavaScript support asynchronous communication and callback back concept.

If we send the data through network environment using any language like Java, python, C or other programming language. Data can be lock or block.

We can send huge data in network environment without blocking. Node JS provide Non block IO Operation with the help of callback and asynchronous communication.

Node JS is open source, cross platform run time environment for server side and networking application.

It provide an event driven architecture and non blocking IO features.

Node JS internally use JavaScript V8 Engine to execute the Node JS Application.

Day 2

21-11-2021

Node JS Modules

Modules in Node JS are simple or complex functionality organized in a single or multiple JavaScript file which can be reuse throughout the application. Module is like a package or namespace.

Type of modules

Core modules ( by default available in node js mean no need to install )

Local module or user-defined module

Third party module ( we have install using npm command).

Syntax to use the module

ES5 style

var obj = require(“moduleName”) : can be core module or local module or third party module.

let obj = require(“module”);

once you load the module then using that reference we have to all function part of that module.

obj.functionName();

Global objects

console : it is a type of global object you can use directly without loading. This object provide few methods to display the object in console.

process : it is a type of global object which help to find the process details.

fs module : Node JS Provide pre-defined core module ie fs module which help to file handling program may be synchronously or asynchronously using JavaScript.

We work with objects.

JSON : Java Script Object Notation

Store and Retrieve JSON Data In File System.

In JavaScript we can create the user-defined object using

1. Function style
2. Class style
3. Literal style

To convert object to string format.

JSON.stringify

Multi page application

DOM Hierarchy DOM Hierarchy

One.html welcome.html

Hyper link

Submit button

Command button

Using JavaScript

SPA (Single Page Application)

<body>

<app-root></app-root>

</body>

Pipe(more than one operator to filter the response)

Day 3

27-11-2021

Taking the value through keyboards in Node JS.

readline : it is a type of node js core module which help to take the value through keyboards asynchronously.

readline-sync This module is use to take the value through keyboards synchronously

This module is not a core module. It is a type of external module we have to install it.

npm install readline-sync -g (globally )

or

npm intstall readline-sync (locally)

http module : http is a type of core module which help to create the dynamic web application using JavaScript (Node JS) application.

http module provide http protocol features. It provide the Node JS server.

Java, Asp.net, Php and Python etc.

Servlet /JSP

Spring boot

Once we created web application using respective technologies we have deploy (run) those application in sever tomcat, web logic, jboss, was and iis etc.

Non Node JS server is thread base server.

Node JS server is not a thread base serve it internally provide great features is event loop.

Tomcat is type of open source web server which help to deploy java application.

Tomcat server is thread base server.

Ticket reservation application.

Thread is a small execution of a code within a process.

Process

Thread

Tomcat

WebLogic

JBoss

100 or 10000 or 100000 that server can take 100 request concurrently.

Node JS tells if server is thread based request can be lock or block.

JavaScript is single thread programming language.

http module

url module : it is a type of core module which help to find the URL Details.

<http://www.google.com:80?id=100&name=Raj>

Query param concept :

URL?key1=value&key2=value2&key3=value3

First information append using and second, third and nth information append using &.

Using http module base upon the path we want to display different message.

Day 4

28-11-2021

Open source Node JS Web Framework came in picture.

Express JS : Express JS is a type of open source web framework which help to develop the web application very easily. Express JS internally use http module to create the web application.

Express JS support all http protocol method

Get, Post, Put and Delete

With the Help of Express JS We can create REST Full web service.

Express MVC style.

We develop small application using Express JS

While developing any node base application we have to create package.json file

Package.json file is known as configuration which all projects details.

Syntax to create this package.

npm init

node jsp provided global property ie \_\_dirname. This property is use to display the current directory path.

If method is Get in Express JS we can call those method using url or hyperlink or form tag with method as get (by default get consider).

If method get we can send the data through url using query param and path param.

Means we can send the data through URL. Get method is not a secure.

So if you want secure data we have to use method as post.

In Express JS if method is post we can call that method through form with post method.

We can’t call post method using hyperlink or URL.

Performance wise get is faster than post method.

In Get we can send maximum 255 character data through URL.

In Express js if data send through Get method we can receive those value using

req.query.fieldname

In Express JS if data send through Post method we can receive throuse value using

req.body

In Express JS framework to enable request body data we have to use another external module

Ie body-parser . That module provide pre-defined property which help to enable data from request body.

Old node JS version body-parser was external we have to download separately. But new version of Express it downloaded with express js itself.

Expression 4.x version

After loaded bodyParser module we have to use this module as a middleware with the help of express js reference as use function.

app.use(middlewareModuleDetails);

Day 5

04-12-2021

When we develop the web application using Express JS. The View is plain html page.

So in plain html page we can’t use variable value or dynamic value which we receive from Express JS application.

Express js framework provided external view engine

Like

Jade

Pug

Haml

Express-view-dom

Limitation of express JS view or template is

The view become tightly coupled with Express JS Application.

View controller

JSP Servlet

Req –http/https JS req

Client HDFC HSBC

Res-🡪 http/https Express java res Java

Asp.net

Python

JS Req

Client Amazon Payment

Express Res Phone pay Java

Paytm Python

Google pay asp.net

Net banking

Credit card

Cash delivery

Web Service : Giving the service for web application when both the application running using different technologies may be same platform or different platform.

1. SOAP base web service.
2. REST Full web service

SOAP Web Service : Simple Object Access Protocol. SOAP Web service base upon SOA (Service Oriented Architecture).

SB (Service Broker) Optional

Register the service details

Lookup In broker.

Service details in SB

SR (Service Requester) SP (Service Provider)

Or

SC (Service Consume)

SOAP web service is one of the implementation of SOA.

SB UDDI Registry

WSDL

Search WSDL file in registry SP

Asp.net

SR (Express JS) SOAP Req checkBalance(accno)

SOAP Res return balance.

Req and Response will send the XML format only.

XML /JSON

Web Service Description language : it is a type of xml file which provide service details. Means service name, service url, service method details(ie number of parameter receive and return the value).

UDDI : Universal Description Discovery Integration. UDDI registry is a type of database which help to store WSDL file.

Service Request download or search wsdl file in registry and using that registry it will create code respective that language which help to call Service in different language.

XML is heavy

SOAP Request and SOAP Response

SOAP Request

Envelop

Header

SOAP Version details

Body

Function name

Data.

XML

DTD

XSD

Xpath

XSL

XSLT

xQuery

In SOAP Web service can consume and produce data only in XML format.

REST Full : Representational State Transfer :

Resource in Web Application Express, Servlet, JSP and Asp.net

Exposing Resources as a web service.

Using Rest Full Web Service we can consume and produce the data in format base upon the client requirement like text, html, xml, json format etc.

Servlet / JSP

Asp.net

Express.js

app.get(“/path”,(req,res)=> {

getBalance(accno);

})

function getBalance(accno) {

return balance.

}

REST Full Web service using all http protocol method to create the Web service.

Ie get, post, put and delete.

Customer, Employee, Account, Manager, Product etc.

Get the resources

Get all Employee details in json or xml format.

Get specific employee details in json or xml format.

Database

Select query

Post the resource or create the resource

Store the employee details they have to send you json or xml.

Store the customer details .

Insert Query

Delete the resource

Delete employee record base upon property empId, name, age etc

Delete query

Put the resource

Update the resource property using another property

Update employee salary using empid

Update employee age using empid

Update query in database.

Create folder

RestFull Web Service Sample Example

Create package.json file using command npm init

Then install express js module

Npm install express

1. <http://localhost:9090/sayHello> return in plain text format
2. <http://localhost:9090/empInfoInJson> return in json format
3. <http://localhost:9090/empInfoXmlFormat> return in xml format
4. <http://localhost:9090/employees> return array of object in json format.

While calling get method if we want to pass the value

1. Using query param concept

URL?key=value passing single value

http://localhost:9090/singleQueryParam?name=RajDeep

URL?key=value&key=value&key=value multiple value

http://localhost:9090/multiQueryParam?user=Raj&pass=123

By default html form with method as Get internally use query param concept.

1. Using Path param concept

URL/value1 passing single value

URL/value1/value2/value3 multiple value

**Post method**

Post method we can’t call through browser URL.

We can call this method through form with post method option.

To test this method we have to use browser plugin.

Arc rest client plugin for chrome

Post man client

Git bash with cURL command

For RestFull Web Service

Client application can be

1. Angular
2. React JS
3. Using JavaScript with promise or Using Ajax with jQuery
4. Using Java Rest Client (Servlet or JSP or Spring boot)
5. Using Python Rest Client
6. Asp.net Rest Client
7. Another Express JS Application
8. Browser plugin (testing purpose)
9. CURL command through Unix OS.

Product Management System

Product entity : pid, pname, price, url

Rest API :

Get all product details : Get method

Get product by using pid : Get method

Store / Add new product information : post method

Pid must be unique

Delete product information using pid : delete method

Update product price using pid : put method

Day 6

05-12-2021

Create the folder product management system

Then create two sub folder

**Backend : Express JS with FS module**

**Frontend : Angular framework**

**Backend**

Create package.json file using the command npm init

npm install express

create app.js file

Create five REST API method

After that now create the Front Angular project

In Frontend folder

ng new angular-product-app

routing 🡪 no

styling 🡪 CSS

cd angular-product-app

ng g c product : This command is use to create the component

ng generate component product

ng g s product : This command is use to create the service

ng generate service product

ng g class product : This command is use to create the model or product class

ng generate class product

We are running two application

1. Express Js : Backend on port number 9090.
2. Angular Framework : Frontend on port number 4200

Two domain are going to communicate to each others.

CORS : Cross Origin Resource Sharing :

So in backend technologies develop any language we have to enable cors policy.

In Express JS we have to install cors module.

Day 7

11-12-2021

We can store the data permanently.

1. File system
2. Database system

File system :

Limitation of file system

1. Data redundancy (duplicate records we can store again and again). We can avoid but depending upon the programming language.
2. Data inconsistency. a. type of file: txt, doc, pdf, excel etc. format of the data in file system.

Id name salary

1 Raj 12000

3. security :

4. CRUD Operation : Insert, Delete, update and retrieve

Database System :

Store the records in table format.

DBMS : Database Management system : it is a software which help to store the data in table format.

Database model

Hierarchical model

Network model

Relational model

12 rules : Dr EF codd’s rules.

Excel sheet is type of DBMS software.

PK

Id Name Salary

1

1

1

MySQL, Oracle, db2, SQL Server and postgres,h2 SQL they are RDBMS software.

RDBMS database schema base Database. Before storing the records in table we have to define the schema. Ie

Table name , number of columns with their data types.

Employee --🡪 Table

Id(int) name(varchar(10)) salary(float) age phonumber

1 Raj 12000 null null

2 Ravi 14000 null null

3 Ramesh 16000 21 null

4 Ajay 18000 null 9900

Trainer\_Student\_Details

TID TName Tech SId SName Age

1 Raj Java 100 Seeta 21

1 Raj Java 101 Reeta 22

1 Raj Java 102 Meeta 23

RDBMS

Trainer

PK

TId TName Tech

1 Raj Java/Pytnon

2 Ravi Python

Student

PK FK

SID SName Age TSId

100 Seeta 21 1

101 Reeta 22 2

102 Meeta 23 1

103 Leeta 25 2

Join

We need to convert JS object into database (query format ) and vice-versa.

No SQL Database :

No SQL Database

Key-value : redis

Graphs database : Neo4J

Document Oriented : Mongo DB

Column family : Casandra

MongoDB is a open source document oriented high performance database.

Mongo DB is use to store the records in json format.

{“id”:100,”name”:”Ravi”,”salary”:120000,”skillset”:[“C”,”C++”,”Java”,”Angular”],”address”:{“city”:”Bangalore”,”state”:”Kar”},”products”:[{“pid”:100,”tech”:”Java”},{“pid”:101,”tech”:”python”}]};

No SQL

Not only SQL

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

Download the mongo DB database and install it

C:\Program Files\MongoDB\Server\5.0\bin

Before run we have to create folder or directory. Default path for MongoDB Server is C Drive : data and inside data db folder.

Then open the command prompt in bin folder and run the command as.

mongod : This command is use to run the mongo db server.

Open another command prompt in bin folder and run the command as mongo. It open the client terminal to interact with mongo db database.

mongo

to check databases in mongo db we can use command as

show databases

or

show dbs

Day 8

12-12-2021

In My SQL we will create the database using command as

create database databaseName

use databaseName;

In Mongo DB

use databaseName; if database present it move to existing database else it will create the database.

Database is a collection of tables.

But in mongo DB table is known as collection.

To check the collection or tables we have to use the command as

show collections

Or

show tables

syntax to create the collection

db.createCollection(“collectionName”);

in Mongo DB collection name are case sensitive.

In Mongo DB records is known a document.

Collection hold more than one document.

db.collectionName.insert({jsonData});

View the documents from a collection

db.collectionName.find();

By default mongo db created one pre-defined property for every document with name as \_id with unique value using ObjectId data types.

\_id is primary key consider in mongo db.

We will create collection with name emp

With fields are \_id, name, salary, age, city

6 to 7 documents.

In mongo db we can insert the document in collection without creating collection.

Insert many records

db.Emp.insertMany([{\_id:7,name:"Seeta",salary:28000,age:31,city:"Bangalore"},{\_id:8,name:"Reeta",salary:34000,age:34,city:"Mumbai"}]);

Retrieve the document from collection using index position

db.collectionName.find()[indexPosition];

db.Emp.find()[1];

db.Emp.find()[2];

retrieve specific index position document field value.

db.Emp.find()[2].name;

db.Emp.find()[1].\_id;

db.Emp.find()[6].city;

retrieve more than one fields values

all

db.collectionName.find({condition},{fieldName});

> db.Emp.find({},{name:1}); : this query display name with pre-defined field \_id

db.Emp.find({},{name:1,\_id:0}); : this query display only name no \_id.

db.Emp.find({},{name:1,\_id:0,age:1}); : this query display name and age fields.

Retrieve more than one fields value with specific documents

db.Emp.find({},{name:1,\_id:0,age:1})[1];

retrieve the document from a collection using conditions.

Conditions with equality

db.Emp.find({\_id:1});

db.Emp.find({name:"Ramesh"});

db.Emp.find({city:"Bangalore"});

db.Emp.find({age:23});

Relational operators

db.Emp.find({salary:{$gt:26000}});

db.Emp.find({salary:{$gte:26000}});

db.Emp.find({salary:{$lt:26000}});

db.Emp.find({salary:{$lte:26000}});

db.Emp.find({salary:{$eq:26000}});

db.Emp.find({salary:{$ne:26000}});

to check more than one condition (logical operator)

$and : both condition must be true

$or : any one condition must be true

db.Emp.find({$and:[{\_id:1},{salary:23000}]});

db.Emp.find({$and:[{\_id:1},{city:"Mumbai"}]});

db.Emp.find({$or:[{\_id:1},{city:"Mumbai"}]});

db.Emp.find({$and:[{city:"Delhi"},{$or:[{salary:32000},{salary:26000}]}]});

db.Emp.find({$or:[{\_id:{$gt:2}},{salary:90000}]}).pretty();

sort the records using fields property

db.Emp.find().sort({name:1}); Ascending order

db.Emp.find().sort({age:-1}); Descending order

db.Emp.find().sort({city:1,age:-1}); City Ascending and age descending order.

Update Query

db.CollectionName.update({condition},{$set:{key:value}});

db.Emp.update({\_id:1},{$set:{age:25}}); update age using \_id property

db.Emp.update({\_id:1},{$set:{age:32,city:"Mysore"}}); update age and city using \_id property

db.Emp.update({city:"Mumbai"},{$set:{salary:45000}}); : if query satisfied more than one document still it will update only one document.

Adding new filed or property to existing documents.

db.Emp.update({\_id:1},{$set:{desg:"Tester"}}); This query check with conditions \_id 1 if desg field already present then it will change the value else add the new fields.

db.Emp.updateMany({},{$set:{dept:"IT"}}) : adding new fields to all documents.

db.Emp.update({\_id:1},{$unset:{desg:1}}); : This query is use to remove specific field property from a collection with conditions.

db.Emp.updateMany({city:"Bangalore"},{$set:{desg:"Tester"}});

db.Emp.find({desg:{$exists:1}}); : display those document which have desg fields or property

db.Emp.find({desg:{$exists:0}}); : display those document which doesn’t have desg field or property

db.Emp.update({$and:[{\_id:1},{age:32}]},{$set:{salary:450000}}); It will update one field value with two condition must be satisfied.

db.Emp.update({\_id:1},{$set:{age:32,city:"Mysore"}});

remove document with or without conditions.

db.sample.remove({}); : This query delete all records from a sample collection.

db.Emp.remove({\_id:1}); : This query delete the document from collection with condition.

Remove collection

db.sample.drop(); : this query is use to remove collection with all documents.

We will create another collection with array values.

Student

\_id, name,age,marks, subjects

Checking the array value

db.Student.find({subject:"Math"}).pretty(); : it check math subject in any index position in subject fields or property

db.Student.find({"subject.0":"Math"}).pretty(); : it check math subject must be in 0 index position in subject field or property

18-12-2021

In RDBMS using Primary Key and Foreign Key

Relationship

4 types of relationship

One – to – many : Trainer --- Student

Many – to – One Employees --- Department or Project

One to One Person Passport

Many – to – Many Students Technologies

In Mongo DB We can do relationship using two ways

1. Embedded Collections (Only one collection)
2. Linking Collections (more than one collection)

Employee has one address : one – to – one

Employee working in more than one project : one – to – many

EmployeeDetails -🡪 Collection

\_id :1

name :Raj

age :21

salary : 25000

skillset : [“Java”,”Python”,”Angular”,”NodeJS”]

address : {city:”Bangalore”,state:”Kar”}

projects :[{pid:100,ptech:”java”},{pid:101,ptech:”python”},{pid:102,tech:[“java”,”node”]}]

\_id :2

name :Ravi

age :23

salary : 28000

skillset : [“Java”,”Python”]

address : [{city:”Bangalore”,state:”Kar”}, {city:”Mumbai”,state:”Mh”}]

projects :[{pid:100,ptech:”java”},{pid:102,ptech:”Angular”} }]

db.Employee.insert({\_id:1,name:"Ravi",age:21,salary:250000,skillset:["Java","Python"],address:{city:"Bangalore",state:"Kar"},projects:[{pid:1001,ptech:"Java"},{pid:1002,ptech:"Python"},{pid:1003,ptech:"Angular"}]});

WriteResult({ "nInserted" : 1 })

db.Employee.insert({\_id:2,name:"Ramesh",age:24,salary:280000,skillset:["Java","Python","HTML","JS","Angular"],address:{city:"Mumbai",state:"Mh"},projects:[{pid:1001,ptech:"Java"},{pid:1003,ptech:"Angular"}]});

db.Employee.insert({\_id:3,name:"Ajay",age:28,salary:320000,skillset:["AI","Python","Machine Learning"],address:[{city:"Mumbai",state:"Mh"},{city:"Bangalore",state:"Kar"}],projects:[{pid:1004,ptech:"AI"},{pid:1003,ptech:"Angular"}]});

db.Employee.find({"address.city":"Bangalore"}).pretty();

db.Employee.find({"projects.ptech":"AI"}).pretty();

Linking Style :

Trainer

PK

\_id TName Tech

1 Raj Java

2 Ravi NodeJS

db.Trainer.insert({\_id:1,tname:"Raj",tech:"Java"});

db.Trainer.insert({\_id:2,tname:"Ravi",tech:"Python"});

Student1 db.Students1.find(); using this concept to retrieve trainer details also we have to depends upon the aggregate operator or functions.

PK it is like a FK

\_id SName Age TSId

100 Seeta 21 1

101 Reeta 22 1

102 Meeta 23 2

103 Veeta 24 2

104 Ueeta 26 [1,2]

db.Student1.insert({\_id:100,sname:"Seeta",age:21,tsid:db.Trainer.find()[0].\_id});

db.Student1.insert({\_id:101,sname:"Reeta",age:22,tsid:db.Trainer.find()[0].\_id});

db.Student1.insert({\_id:102,sname:"Meeta",age:23,tsid:db.Trainer.find()[1].\_id});

db.Student1.insert({\_id:103,sname:"Veeta",age:24,tsid:db.Trainer.find()[1].\_id});

db.Student1.insert({\_id:104,sname:"Leeta",age:25,tsid:[db.Trainer.find()[0].\_id,db.Trainer.find()[1].\_id]});

Student2 db.Student2.find(); no need to depends upon the Trainer table.

\_id SName Age TSId

100 Seeta 21 {1 Raj Java}

101 Reeta 22 {1 Raj Java}

102 Meeta 23 {2 Ravi NodeJS}

103 Veeta 24 {2 Ravi NodeJS}

104 Ueeta 26 [{1 Raj Java },{2 Ravi NodeJS}]

db.Student2.insert({\_id:101,sname:"Reeta",age:22,tsid:db.Trainer.find()[0]});

db.Student2.insert({\_id:100,sname:"Seeta",age:21,tsid:db.Trainer.find()[0]});

db.Student2.insert({\_id:103,sname:"Leeta",age:25,tsid:db.Trainer.find()[1]});

db.Student2.insert({\_id:104,sname:"Keeta",age:28,tsid:[db.Trainer.find()[0],db.Trainer.find()[1]]});

Aggregate functions

It is use to group multiple documents from one collection or more than one collection and then perform aggregation operation on it and after that it return single result or multiple result depends upon the operator.

db.Student1.aggregate([{$lookup:{from:"Trainer",localField:"tsid",foreignField:"\_id",as:"Trainers"}}]).pretty();

db.Trainer.aggregate([{$lookup:{from:"Student1",localField:"\_id",foreignField:"tsid",as:"Students"}}]);

Aggregate with conditions

db.Student1.aggregate([{$match:{\_id:100}},{$lookup:{from:"Trainer",localField:"tsid",foreignField:"\_id",as:"Trainers"}}]).pretty();

db.EmployeeDetails.aggregate([{$group:{\_id:"$city"}}]); city wise group

db.EmployeeDetails.aggregate([{$group:{\_id:"$deptId"}}]); deptId wise group

db.Football.insertMany([

{ "\_id" : ObjectId("5b26835999e1647ee04cd3f0"), "fName" : "Gareth", "lName" : "Bale", "league" : "La Liga", "goalsScored" : 50, "year" : 2011 },

{ "\_id" : ObjectId("5b26835a99e1647ee04cd3f1"), "fName" : "Gareth", "lName" : "Bale", "league" : "La Liga", "goalsScored" : 50, "year" : 2011 },

{ "\_id" : ObjectId("5b27927001271eb9de48c821"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "La Liga", "goalsScored" : 25, "year" : 2011 },

{ "\_id" : ObjectId("5b2792ce01271eb9de48c822"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "La Liga", "goalsScored" : 30, "year" : 2012 },

{ "\_id" : ObjectId("5b2792ce01271eb9de48c823"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "La Liga", "goalsScored" : 35, "year" : 2013 },

{ "\_id" : ObjectId("5b2792ce01271eb9de48c824"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "La Liga", "goalsScored" : 40, "year" : 2014 },

{ "\_id" : ObjectId("5b2792ce01271eb9de48c825"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "La Liga", "goalsScored" : 50, "year" : 2015 },

{ "\_id" : ObjectId("5b2792ff01271eb9de48c826"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "Champions League", "goalsScored" : 10, "year" : 2011 },

{ "\_id" : ObjectId("5b27933b01271eb9de48c827"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "Champions League", "goalsScored" : 15, "year" : 2012 },

{ "\_id" : ObjectId("5b27933b01271eb9de48c828"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "Champions League", "goalsScored" : 20, "year" : 2013 },

{ "\_id" : ObjectId("5b27933b01271eb9de48c829"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "Champions League", "goalsScored" : 25, "year" : 2014 },

{ "\_id" : ObjectId("5b27933b01271eb9de48c82a"), "fName" : "Cristiano", "lName" : "Ronaldo", "league" : "Champions League", "goalsScored" : 30, "year" : 2015 }])

Total goal Per League

Make group by league

Make group by fname wise

Make group by lname wise

db.Football.aggregate([{$group:{\_id:"$lName",total:{$sum:"$goalsScored"}}}]);

Index : index is every import concept is every database.

Using the index we can improve the performance.

db.EmployeeDetails.find({city:"Bangalore"}).explain("executionStats");

db.EmployeeDetails.createIndex({city:1}); automatically name created

db.EmployeeDetails.getIndexes();

db.EmployeeDetails.dropIndex({city:1});

db.EmployeeDetails.createIndex({city:1},{name:"MyCityIndex"});

connection mongo db databse using mongodb module and mongoose mdoules.

19-12-2021

Connection Mongo DB database through Node JS and Express JS

1. Using mongodb
2. Using Mongoose

mongodb : it is external module which provide set function which help to connect mongo db database through node JS.

First create the folder as MongoDB Modules Programs

Then create the package.json file using command as

npm init

npm install mongodb

Express JS with Mongo Db module with Mongo DB Database.

Create new folder with name Express JS with MongDB Modules

Create the package.json file using npm init

npm install express mongodb

mongoose module

mongoose is external module which help to store the data in mongo db module. Mongoose module use ODM concept (Object data modelling). Mongoose module internally use mongo db module which support schema concept.

Mongodb is native driver.

Java

JDBC is like mongo db module

ORM is like mongoose module

Create mongodb module example folder

Create the package.json file using npm init command

npm install mongoose

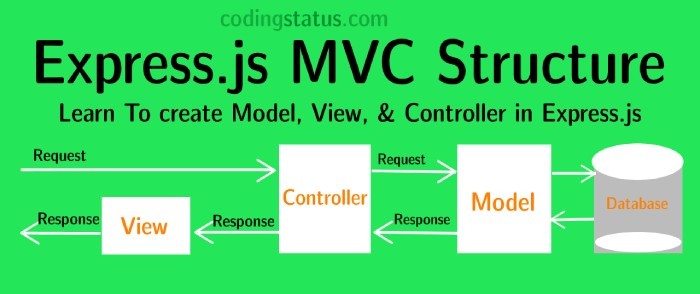
in Mongoose we are going to create schema

Schema provide the structure for the collection.

Using schema we have to create the model.

Model take the help of schema definition and provide the collection name.

Express MVC : Model View Controller



Frontend

View ---🡪 Angular Framework

Backend

Main file

app.js

Load the all modules

Express js, cors, mongoose or any other

Connect the database

Add middleware

Check main path and base upon the main path request will pass to router file.

Run the application on specific port number.

Router layer 🡪 product.router.js

This file receive the request from main file ie app.js

It check sub path as well as http request method and base upon the method it will call controller method to do the operation on collection.

Controller layer -🡪 product.controller.js

This file import the model file.

With the help of model reference do the operation on collection ie insert, delete, update and retrieve base up on request receive from the router file.

Model layer -🡪 product.model.js , customer.model.js, login.model.js

load the mongoose module

then create the schema

using schema create the model which provide collection name

export it .

Database -🡪 Mongo DB

Create the folder as

**Express MVC :**

Inside Express MVC create backend folder.

Create the package.json file using npm init command

Npm install express cors mongoose

Then create the folder as controller, router, model

App.js -------🡪Router.js --------🡪Controller.js 🡨-----model.js ----🡪 Db

Socket.io is a library base upon the Web Socket api.

Web Socket api is base on socket programming

A socket is a communication connection point (end point) that you can name and address in the network environment. Using socket programming through network we can send and receive the data. Socket is use to achieve two way communication.

Client is browser and Server is another application.

Socket programming both are client or server which help to share the two data through network.

Both are application which is use to share the data.

Chatting program.

Node js provide core module ie net module which help do socket programming.

In Socket programming both are client or server. Both application use to send and receive the data in desktop and standalone application.

Web Socket : Socket programming on web application.

Without Web Socket the web application is known as one way communication.

Client server

With web socket we can achieve two ways communication on web application.

Web Socket programing using node JS

Node JS provided two pre-defined external module ie

express and express-ws (web socket).

Client side (Client Side JavaScript programming ) they provide WebSocket is pre-defined object which help to achieve web socket programming.

Create the folder as WebSocket Programs

Create the package.json file using command as npm init

npm install express

npm install express-ws

socket.io : socket.io is a library function base upon Web Socket which help to do socket programming on web application.

Web Socket is like a JavaScript

Socket.io is like a jQuery

Socket.io internally use express js and http module.

Create folder as socket.io

Then create the package.json file using command as npm init

Install

npm install express

npm install socket.io

15-01-2021

Authentication and Authorization in Express JS with Rest API

In simple word, Authentication is the process of verifying who user is and authorization is the process of verifying what they have to access.

Product Entity

Create product

Delete product

Update the product

View the product

Order Entity

Authentication

Then we can access those rest api.

Authorization

Normal user

View the product

Order the product

Admin user

Insert, delete and update

Http is a stateless protocol.

Cookies : cookies is a small text file created by server when any client send request to server. This file contains unique id ie sessionId. This file stored in client machine.

1st req ----------------------------------🡪

Client Server

1st res 🡨------------------------------+cookies files (sessionId)

JWT (Json Web Token)

1st req ----------------------🡪

Get product + token

Client Server

🡨--------------------1st res + JWT Token

Success message + token

First section contains header part

Type of algorithms

Second section contains payload.

User information

Node JS provide external module

Jwttoken and password has ie bcryptjs module to store the password in bcrypt format.

Create the folder as Authentication and Authorization

Create the package.json file using command as

npm init

npm install express

npm install mongoose

npm install jsonwebtoken

npm install bcryptjs

123 we will different token

123@123 we will different token

admin we will different token

user ----------------------🡪 SignUp (Register ) ------------------------🡪

user ---------------------🡪Sign In (user/admin)--------------------🡪

if success + token generate

user or admin send request to REST API with Token if token is valid we have to allow to access the REST API or give error message.

First check the token

If token is valid

Then check the type of user.

If user is normal user allow to use getProductDetails REST API

If user is admin user allow to user getProdcutDetails and PostProductDetails REST API