https://shorturl.at/kAHNV

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

**Swagger api documentation OAS(Open Application Specification)**

**EJS**

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

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

Swagger api documentation OAS(Open Application Specification)

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

* Swagger for Node.js simplifies the process of designing, documenting, and testing APIs by providing a comprehensive set of tools and libraries tailored for Node.js developers.
* It promotes API consistency, improves developer productivity, and enhances API usability by providing self-descriptive documentation and interactive exploration capabilities.

*Refer MongoDB CRUD example*

update directory structure with apiRoutes

<>

Routes

* apiRoutes.js

fetch

- fetch.js

insert

- insert.js

update

- update.js

delete

- delete.js

- url.js

- server.js

\*\*\*apiRoutes.js\*\*\*

//create router instance

const router = require('express').Router()

//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

router.use('/fetch', fetch)

router.use('/insert', insert)

router.use('/update', update)

router.use('/delete', remov)

module.exports = router

>yarn add swagger-jsdoc swagger-ui-express --save

Import above modules in server.js

Configure options

Initialise swaggerJsDocs specs by passing options to it

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

//import modules express body-parser cors swagger-ui-express swagger-jsdoc

let express = require('express')

let bodyparser = require('body-parser')

let cors = require('cors')

let swaggerUI = require("swagger-ui-express")

let swaggerJsDoc = require("swagger-jsdoc")

//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

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

const options = {

definition: {

openapi: "3.0.0",

info: {

title: 'Product Management',

version: "1.0.0",

description: "Simple Product Management API"

},

servers: [

{

url: "http://localhost:8080"

}

]

},

apis: ["./Routes/\*.js"]

}

//Initialise swaggerJsDocs specs by passing options to it

const specs = swaggerJsDoc(options)

app.use("/api-docs", swaggerUI.serve, swaggerUI.setup(specs))

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

//import apiroutes

const apiRoutes = require("./Routes/apiRoutes")

//use apiRoutes

app.use("/", apiRoutes)

//assign port no

app.listen(port, () => {

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

})

/\*

>node server

http://localhost:8080/api-docs/ -> check in browser

\*/

Refer swagger components

https://swagger.io/docs/specification/components/

update apiRoutes.js

\*\*\*apiRoutes.js\*\*\*

//=================Create schema=================//

/\*\*

\* @swagger

\* components:

\* schemas:

\* Products:

\* type: object

\* required:

\* - p\_name

\* - p\_cost

\* properties:

\* p\_id:

\* type: number

\* description : Products id

\* p\_name:

\* type: string

\* description: Products name

\* p\_cost:

\* type: number

\* description: Products cost

\* example:

\* p\_id: 111

\* p\_name: p\_one

\* p\_cost: 10000

\* Product:

\* type: object

\* properties:

\* p\_id:

\* type: number

\* description : Products id

\* example:

\* p\_id: 111

\*/

//=================Name instead of default=================//

/\*\*

\* @swagger

\* tags:

\* name: Products

\* description: "Products management api"

\*/

//=================tags -> grouping under 'Products'=================//

//================= GET =================//

/\*\*

\* @swagger

\* /fetch:

\* get:

\* summary: Returns list of all products

\* tags: [Products]

\* responses:

\* '200':

\* description: List of all Products

\* content:

\* application/json:

\* schema:

\* type: array

\* items:

\* $ref: '#/components/schemas/Products'

\*/

//================= POST =================//

/\*\*

\* @swagger

\* /insert:

\* post:

\* summary: Create a new product

\* tags: [Products]

\* requestBody:

\* required: true

\* content:

\* application/json:

\* schema:

\* $ref: '#/components/schemas/Products'

\* responses:

\* '200':

\* description: "Product created"

\* content:

\* application/json:

\* schema:

\* $ref: '#/components/schemas/Products'

\* '500':

\* description: "Server Error"

\*/

//================= PUT =================//

/\*\*

\* @swagger

\* /update:

\* put:

\* summary: "Update existing Product by p\_id"

\* tags: [Products]

\* requestBody:

\* required: true

\* content:

\* application/json:

\* schema:

\* $ref: '#/components/schemas/Products'

\* responses:

\* '200':

\* description: "Product updated"

\* content:

\* application/json:

\* schema:

\* $ref: '#/components/schemas/Products'

\* '500':

\* description: "Server Error"

\*/

//================= DELETE =================//

/\*\*

\* @swagger

\* /delete:

\* delete:

\* summary: "Delete existing Product by p\_id"

\* tags: [Products]

\* requestBody:

\* required: true

\* content:

\* application/json:

\* schema:

\* $ref: '#/components/schemas/Product'

\*

\*

\* responses:

\* '200':

\* description: "Product deleted"

\* content:

\* application/json:

\* schema:

\* $ref: '#/components/schemas/Product'

\* '500':

\* description: "Server Error"

\*/

//create router instance

const router = require('express').Router()

//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

router.use('/fetch', fetch)

router.use('/insert', insert)

router.use('/update', update)

router.use('/delete', remov)

module.exports = router

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

EJS

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

* EJS, which stands for Embedded JavaScript, is a simple templating engine for JavaScript.
* It allows you to embed dynamic JavaScript code within your HTML markup, making it easier to generate dynamic content on the server side.
* EJS templates look similar to regular HTML but with the ability to include JavaScript code.

Features

1. Embedding JavaScript: EJS allows you to embed JavaScript code directly into your HTML templates using <% %> tags. For example:

<h1><%= title %></h1>

In this example, the <%= title %> expression will be replaced with the value of the title variable during rendering.

1. Control Flow: EJS supports basic control flow structures like if statements and loops using <% %> tags. For instance:

<% if (user.isAdmin) { %>

<p>Welcome, Admin!</p>

<% } else { %>

<p>Welcome, User!</p>

<% } %>

1. Partial Views: EJS allows you to create reusable components or partial views using the <%- include('partial') %> syntax.

<div>

<%- include('header') %>

<p>Main content goes here.</p>

<%- include('footer') %>

</div>

1. Escaping HTML: EJS provides automatic HTML escaping to prevent cross-site scripting (XSS) attacks. You can use <%- %> to render unescaped HTML content and <%= %> for escaped content.

<p>Unescaped: <%- unsafeHTML %></p>

<p>Escaped: <%= safeHTML %></p>

GO WITH MONGOOSE EG

Download pacakges

>yarn add express mongoose express-session ejs dotenv

Create ‘views’ directory in root directory to keep all .ejs files

Directory structure

<>

views

update server.js as per comment

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

//import modules

const express = require('express')

let bodyparser = require('body-parser')

let cors = require('cors')

let mongoose = require('mongoose')

const session = require('express-session')

//import url

let url = require('./url')

//creater 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).then(() => {

console.log('Connection success')

}, () => {

console.log('Connection Failed')

})

//\*\*\*\*\*\*\*\*\*\*\*\*UPDATE THIS PART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//use template engine by default it will use

app.set("view engine", "ejs")

/\* Optional part

// Specify the directory where your EJS files are located

app.set('views', \_\_dirname + '/myEjs'); \*/

app.use(session({

secret: "my secrete key",

saveUninitialized: true,

resave: false

}))

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

//=================================================//

//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)

})

create directory structure

<>

views

layout

header.ejs

footer.ejs

index.ejs

fetch.ejs

insert.ejs

update.ejs

delete.ejs

Create ejs templates as

\*\*\*header.ejs\*\*\*

<!DOCTYPE html>

<html>

<head>

<title> <%- title %> </title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>

</head>

<body class="container mt-5">

<h1 class="text-light bg-secondary">This is header</h1>

<ul class="nav w-50 mx-auto mb-3">

<li class="nav-item">

<a href="/insert" class="nav-link">Create</a>

</li>

<li class="nav-item">

<a href="/fetch" class="nav-link">Read</a>

</li>

<li class="nav-item">

<a href="/update" class="nav-link">Update</a>

</li>

<li class="nav-item">

<a href="/delete" class="nav-link">Delete</a>

</li>

</ul>

\*\*\*footer.ejs\*\*\*

<h1 class="text-light bg-danger my-3">This is Footer</h1>

</body>

</html>

\*\*\*index.ejs\*\*\*

<%- include('layout/header') %>

<%- include('layout/footer') %>

\*\*\*productApis.js\*\*\*

//import db schema

const Product = require("../model/Product")

const trial = async (req, res) => {

res.render('index', { title: 'Home Page' })

//res.status(200).send({'message':'Trial api responce'})

}

//get all products

const products\_all = async (req, res) => {

try {

const products = await Product.find()

//res.status(200).send(products)

res.render('fetch', {

title: "Display",

products: products,

message: { message: '' }

})

console.log('Data Sent')

}

catch (error) {

res.status(400).send({ 'message': error })

}

}

//insert a product

const insert\_page = async (req, res) => {

res.render('insert', { title: 'Create' })

}

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()

product.save()

req.session.message = {

type: 'Success'

}

console.log('Product inserted')

//res.status(200).send(savedProduct)

}

catch (error) {

res.status(400).send(error)

}

res.redirect('/fetch')

}

//update product

const update\_page = async (req, res) => {

res.render('update', { title: 'Update', updateMessage: { type: 'primary' } })

}

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) {

res.render('update', {

title: 'Update Page',

updateMessage: { type: 'success', text: 'Product updated successfully.' }

});

console.log('Product Updated', updateProduct)

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

}

else {

console.log('Product not updated')

res.render('update', {

title: 'Update Page',

updateMessage: { type: 'danger', text: 'Product not found.' }

});

//res.status(200).send({ 'update': 'Record Not Found' })

}

}

catch (error) {

res.status(400).send(error)

}

}

//delete product

const delete\_page = async (req, res) => {

res.render('delete', { title: 'Delete', deleteMessage: { type: 'primary' } })

}

const delete\_product = async (req, res) => {

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

try {

const deletedProduct = await Product.deleteOne(

{ p\_id: p\_id }

)

if (deletedProduct.deletedCount != 0) {

res.render('delete', {

title: 'Delete Page',

deleteMessage: { type: 'success', text: 'Product deleted successfully.' }

});

console.log('Product Deleted')

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

}

else {

console.log('Product Not deleted')

res.render('delete', {

title: 'Delete Page',

deleteMessage: { type: 'danger', text: 'Product not found.' }

});

//res.status(200).send({ 'delete': 'Record Not Found' })

}

} catch (error) {

res.status(400).send(error)

}

}

module.exports = {

trial,

products\_all,

insert\_page,

insert\_product,

update\_page,

update\_product,

delete\_page,

delete\_product

}

in productRoutes.js add each route with get request

\*\*\*productRoutes.js\*\*\*

//import express module

const express = require('express')

//create router instance

const router = express.Router()

//import productApi

const productApi = require('../apis/productApis')

//trial

router.get("/", productApi.trial)

//fetch all records

router.get("/fetch", productApi.products\_all)

//insert a record

router.get("/insert", productApi.insert\_page)

router.post("/insert", productApi.insert\_product)

//update a record

router.get("/update", productApi.update\_page)

router.post("/update", productApi.update\_product)

//delete a record

router.get("/delete", productApi.delete\_page)

router.post("/delete", productApi.delete\_product)

//export router

module.exports = router

\*\*\*fetch.ejs\*\*\*

<%- include('layout/header') %>

<table class="table table-dark table-striped table-hover table-bordered w-50 mx-auto">

<h4 class="text-primary w-50 mx-auto"> Fetch Page</h4>

<thead>

<tr>

<th>P\_id</th>

<th>P\_name</th>

<th>P\_cost</th>

</tr>

</thead>

<tbody>

<% products.forEach((product)=> { %>

<tr>

<td>

<%= product.p\_id %>

</td>

<td>

<%= product.p\_name %>

</td>

<td>

<%= product.p\_cost %>

</td>

</tr>

<% }); %>

</tbody>

</table>

<p><%= message.message %></p>

<%- include('layout/footer') %>

\*\*\*insert.ejs\*\*\*

<%- include('layout/header') %>

<title> <%- title %> </title>

<form class="w-50 mx-auto" action="/insert" method="post">

<h4 class="text-secondary">Create Page </h4>

<div class="form-group">

<label>P\_id</label>

<input type="number" class="form-control" name="p\_id">

</div>

<div class="form-group">

<label>P\_name</label>

<input type="text" class="form-control" name="p\_name">

</div>

<div class="form-group">

<label>P\_cost</label>

<input type="number" class="form-control" name="p\_cost">

</div>

<input type="submit" value="Insert" class="btn btn-success">

</form>

<%- include('layout/footer') %>

\*\*\*update.ejs\*\*\*

<%- include('layout/header') %>

<title> <%- title %> </title>

<form class="w-50 mx-auto" action="/update" method="post">

<% if (updateMessage) { %>

<div class="alert alert-<%= updateMessage.type %> alert-dismissible fade show" role="alert">

Status <%= updateMessage.text %>

<button type="button" onclick="navigateToFetch()" class="close" data-dismiss="alert"

>&times;</button>

</div>

<script>

function navigateToFetch() {

// You can use window.location.href or window.location.replace for navigation

window.location.href = '/fetch';

}

</script>

<% } %>

<h4 class="text-warning">Update page</h4>

<div class="form-group">

<label>P\_id</label>

<input type="number" class="form-control" name="p\_id">

</div>

<div class="form-group">

<label>P\_name</label>

<input type="text" class="form-control" name="p\_name">

</div>

<div class="form-group">

<label>P\_cost</label>

<input type="number" class="form-control" name="p\_cost">

</div>

<input type="submit" value="update" class="btn btn-success">

</form>

<%- include('layout/footer') %>

\*\*\*delete.ejs\*\*\*

<%- include('layout/header') %>

<title> <%- title %> </title>

<form class="w-50 mx-auto" action="/delete" method="post">

<% if (deleteMessage) { %>

<div class="alert alert-<%= deleteMessage.type %> alert-dismissible fade show" role="alert">

Status <%= deleteMessage.text %>

<button type="button" onclick="navigateToFetch()" class="close" data-dismiss="alert" aria-label="Close">&times;</button>

</div>

<script>

function navigateToFetch() {

// You can use window.location.href or window.location.replace for navigation

window.location.href = '/fetch';

}

</script>

<% } %>

<h4 class="text-danger">Delete Page</h4>

<div class="form-group">

<label>P\_id</label>

<input type="number" class="form-control" name = "p\_id">

</div>

<input type="submit" value="Delete" class="btn btn-danger">

</form>

<%- include('layout/footer') %>

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

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