8/04/2023

Backend development

$$\begin{tikzcd} \text{Front end} \arrow[r, "\text{interaction}"] \arrow[loop below, "\text{User}"] & \text{Backend} \end{tikzcd}$$

Front end → UI, styling, user is viewing the front end whereas front end is interacting with backend. The brain lies at the backend.

Client server interaction can be done via HTTP request where HTTP is Hyper Text Transfer Protocol where protocol is set of rules.

Ex → Link
↳ Click → OS → default browser
↓
But here browser understand IP address, here concept of domain name resolution gives the IP address of the Server
↳ machine on which the website is hosted.
↓
Now connection is established using TCP via handshake, also encryption is done
↓
3 way handshake

Scanned with Cam

After all the above discussed steps, get request is ready.

Types of request in HTTP

1) get
2) put
3) post
4) delete

* The work of get is to fetch.
* The work of put is to update.
* The work of post is to submit.
* The work of delete is to remove.

Express js
It is a framework to create server side application.

Steps
1) Create a folder & open that folder inside the terminal.
2) Write npm init -y to install package.json file
3) Write npm i express to install node modules folder.
4) Create a file named server.js in the same folder.

server.js file

```
const express = require ('express');
const app = express ();
→ Server is instantiated
//3000 port number
app.listen (3000, () => {
```

Scanned with Cam

```
    console.log ("Server started")
});
```

Now in terminal write

```
node server.js
```

Output

```
Server started
```

get request ↗ Route

```
app.get ('/', (request, response) => {

    response.send ("Hello jee");
})
```

"Hello jee" — This is printed when we are on home page

post request → Submit

```
app.post ('/api/cars, (req, res) => {

    const {name, brand} = req.body;
    console.log (name);
    console.log (brand);
    res.send ("Car submited");
})
```

* Re-run the server.
* Go to localhost : 3000 /api/cars but this is not how we can see the post request.

Verifying the post request
Here we will be using the postman.

1) Go to new → HTTP request → POST request
2) Body → raw → json

Scanned with Cam

```
{
    "name" : "DLICR",
    "brand" : "Scorpio"
}
```

But here we get an error due to parsing issue. For that we need to use body Parser.
//Used in post or put.

```
const bodyParser = require ('body-parser');
app.use (bodyParser.json ()); → Specifically
```

parse JSON data & add it to request.body Object

Just re-run the server & then submit button needs to be clicked in the postman.

Mongo DB → open source

↳ This is No-SQL database. It is a type of storage in which data can be stored in form of document, key-value pair, graph etc.
There is no concept of tables in MongoDB & the scalability is good.

Here CRUD operations can be done in Mongosh via commands. But we will be using easy method such as Mongodb-compass. It is a GUI basically.

MongoDB compass
* Create database name & collection names to be written.
↳ Cars ↳ Mahindra

Scanned with Cam

classmate
Date
Page

* Adding data
Go to documents & then add data and then insert into document. Write

```
{
    "name" : "DLICR",
    "brand" : "Mahindra"
}
```

* Reading the data
Go to documents & then write inside the box of filter write

{"name" : "DL ICR"}. Click on find

* Updating the data

{"name" : "DLIBC"}. Click on replace.

* Deleting the data.
Simply click the delete icon.

Note → By default mongoDB works on local host: 27017.

Linking of server and database
Optimized way of doing this is using mongoose.
ODM → Object Data Modelling.
ODM provides functionality to server & database both.

$$\begin{tikzcd}
\text{Server} \arrow[rr, leftrightarrow] & {} \arrow[d] & \text{DataBase} \\
& \text{Mongoose} &
\end{tikzcd}$$

Scanned with Cam

classmate
Date
Page

Steps

1) npm i mongoose in terminal.
2) const mongoose = require ('mongoose');

```
moongose. connect ('mongodb://localhost:27017/
myDataBase', {
            useNewUrlParser: true,
            useUnifiedTopology: true
})
.then (() => {console.log ("Success")})
.catch ((error) => {console.log ("Error")});
```

→ Promises.

Scanned with Cam