Monday, January 16, 2023 11:52 AM

Good Morning :))
Express Server = JavaScript library used to build web server

Steps1 -- create a folder
Step2 --- npm init } it will set up the package.json
Step3 --- npm install express } node_modules folder is created with the libraries
Step4 ---- create a cwserver.js

Writing the JS file ----------

1.

Import	Require
ECMA 6	ECMA5 onwards
We have to add type:"module" in package.json	We have to remove type:"module "
import express from "express" From the library express import the default exported express	let express= require("express")

Server code

1. Either import or require get the library
2. Get the object of express
    i. Let app = express()
3. app.listen (port number, callback function)

On the server side ---
All the http methods can be MAPPED to different URLs
We can give the implementation of the **Restful Service** in the CALLBACK method associated with the http method

app.httpmethod(URL , **callback-service**)

We can access the mapped methods from

1. Browser
2. Postman
3. AJAX calls from (javascript , jquery , React)

In the Service implementation ----------

1. The callback method associated with the http method gets two parameters
    i. Request, response
2. Use response we can do the following
    i. Set the response status response.status(200)
    ii. Send Text or JSON data response.send(text/json/xml)

iii. Render HTML response.render(…..)

3. Using request object we can do the following
   1. read data sent using path parameters req.params.user
   2. read data sent using query params req.query.user
   3. read data sent using request body req.body.user
4. Write libraries and call them for DB Connectivity or Data processing

---

Routing using Express

1. Routing allows us to create sub paths for RESTFul Services
2. We create a file routemodule.js
   a. We use let router = express.Router()
   b. router.httpmethod(url , callback) //httpmethod = GET POST PUT DELETE
   c. To use the routes add the following in the main server

```
var bookroutes = require('./routemodule1.js');
The above line will work only if ( export the router ===
modules.export = router is written in the routemodule1.js )

app.use('/book', bookroutes);
This will redirect all urls having /book to bookroutes
```

3. We can organize the routes as per modules of our project

TO Enable the Express to read the JSON in request body

Include

app.use(**express.json()**)

Otherwise we get an error as follows

Cannot read properties of undefined (reading 'num1')

While we are using the following

req.body.num1

Templates Template Engines in Express ----

1 . To create Dynamic HTML we need TEMPLATES (files having html embedded with variables)

2. When client request arrives --- depending on client data
   The template variables are populated
   The templates are converted to HTML
3. The client gets HTML

In Express we can Choose for many Template Engines (pug, hbs, EJS ,…..)

This is necessary for building MVC web application --- where VIEW generation is crucial

Steps1 -- first install the template engine of your choice

Go to the node project folder

npm install hbs (or give any template engine)

Step 2 ----- Study the syntax of that template engine and create the html files

welcome.hbs

changeBook.hbs

….

….

Step 3 ---- Inform the express server about the template engine and the view folder

```
app.set('views', './views');
app.set('view engine', 'hbs');
```

Step 4 ------ In the controller **render** the template

app.get(url , (req,res) => {

```
                    Res.render("template name without extension " )
                     for ex  resp.render("changeBook")
                })
```

Node JS -------

Follows the ECMA 6 specification

You can revise over here - http://es6-features.org/

Setter and Getter

Used to make some properties as private in a JS class

```
class
{
        set  (propertyname exposed to others)  (parameter)  { _
                APPLY proper conditions here
                propname = parameter
            }
        get (propertyname exposed to others ) ()  {
            We can create copies and return
            return  _propname
        }
}
```

Actual property that is private is _propertyname

Dummy property is propertyname

Whenever the user changes propertyname = something // SETTER is called

Whenever the user user propertyname // Getter is called

---

async and await keywords in JS ------

1. async function hi()
   {
   }

   This function always returns a Promise !!!

2. within an async function - we can add await
3. Await statement --------
   await Promise
4. The code will BLOCK on the **await** line
   Till the promise on which await is waiting does not resolve
   Once the promise resolves the code after await is executed

---

File Handling in node JS -----

1. Write to File
2. Read From File

These two operations can happen in Two ways

1. Synchronously ---- writeFileSync , appendFileSync , readFileSync

2. Asynchronously ---writeFile , appendFile, readFile