Q. What is a node JS?

Node js is not language basically it is a server environment means we can say node js provide server environment for JavaScript as well as using node we can communicate with database and it is open source and using node js we can create API and call at client side and node use the chrome v8 engine to execute.

Q. Why do we use node JS?

1. Node used for creating API
2. So we can connect the same database with web APP ,mobile App or devices also.
3. Node easily understand by people who knows JavaScript
4. Node is super fast API
5. With the help of Node and JavaScript you can become a full stack developer.

Q. Are JavaScript and Node the same?

1. JavaScript and Node syntax are same
2. If you know JavaScript you can easily understand Node..
3. Both are exactly not same
4. Node can connect with a database
5. Node create server environment but JavaScript create client side environment

If we want to work with node js we need to know the following things

1. **What is a server?**

   Server is a application /software which is responsible for execute application at central location and provide access of application to remote clients as well as accept the request from client and process on it may be communicate with database also and generate proper response according to client request and send response to client

   Note: if we think about java server side environment we can use apache, glassfish or web logic as server or if we think about node then we have node software which provide us server side environment

2. **What is a client?**

   Client is also an application which is responsible to send requests to server applications by using hyperlink or form etc and accept the response generated by server and show to the end users.

Note: if we think about web application browser act as client software

What do developers make with Node JS?

1. Developer make API with Node JS
2. Node js can connector or provide communication between client and server
3. Node JS Can communicate with database

4. Node JS create API and integrate with web application /mobile application etc

How to work with Node JS?

---

1. **Download and install the node js**
   If we want to download the node js you can visit following URL
   https://nodejs.org/en/download

Note: when we install node we get one inbuilt tool known as NPM

Q. What is NPM?

NPM stands for Node Package Manager which is used for downloading the external libraries required for node applications like maven in java.

If we want to check the installed property or not we have to use the following steps.

1. **Open command prompt**
2. **Type the following command**

If we want to check version of npm we have to use following command

How to create application by using node

If we want to create application using node we have two ways

1. **Script with command line**
2. Make folder and files

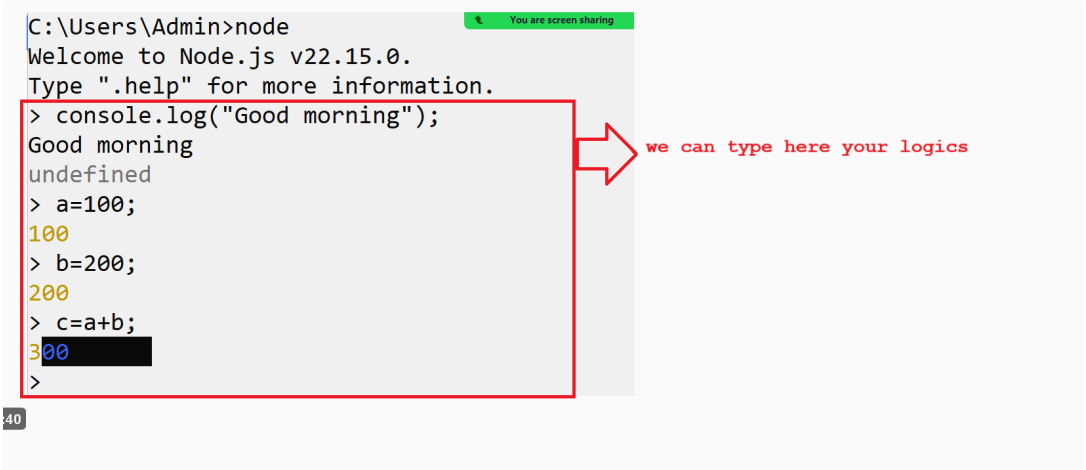
Script with command line

Steps

1. **Open command prompt and type command node on it.**

```
C:\Users\Admin>node
Welcome to Node.js v22.15.0.
Type ".help" for more information.
> 
```

2. **We can write code on command prompt.**



Note: this is not standard way to work with node

We have one more way to work with nodes: use any text editor and write code in text editor save file using .js extension and save and run using node.

Steps.

1. Write javascript code using any text editor
2. Save file using .js extension
3. Run the file by using node environment

Core Module In Node JS

Q. What are the core modules?

In every programming language there is some basic feature like as database connection feature, file creation feature , API calling feature as well as to process the process the code called as core module

Example: FS,Buffer, Http etc

There are two types of core module in Node

---

1. **Global Module :** global module means module not needed to import in an application called as global module.
   Example: console.log(“good morning”);

```
console.log(__dirname);
console.log(__filename);
```

Output

```
D:\nodetutorial>node demo.js
D:\nodetutorial
D:\nodetutorial\demo.js

D:\nodetutorial>
```

2. **Non Global module :** non global module means need to import in application called as non global module

Interview Question

---

1. What is node and why use it?
2. What does server and client explain with an example?
3. How to create and run a node program?
4. What is the difference between JavaScript and node?
5. What is the core module in JavaScript and explain its type?
6. Is it compulsory to use the required() function at top?

Now we want to discuss about non global http module

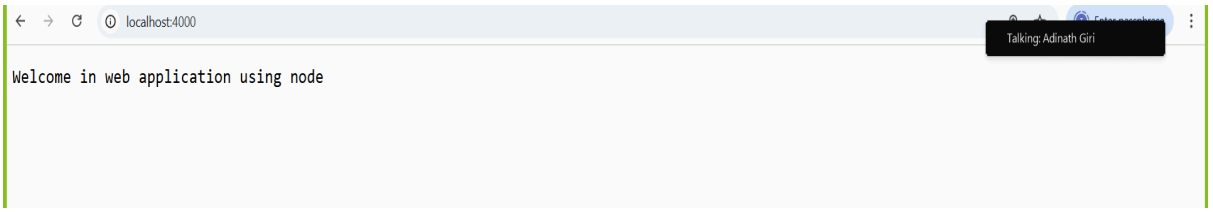
Http module help us to create server means http module has one function name as createServer() which is used for create server and this function contain call back as parameter and callback function contain two parameters req,res and when we create we required to use one method name as listen() and this method is used for provide port number to server

```
let http=require("http");


http.createServer(function(req,res){
   write here logics means
   accept request from client and send response to
   client
}).listen(portnumber,fuction(){

    });
```

Example:

```
let http=require("http");

http.createServer((req,res)=>{
    res.write("Welcome in web application using node");
    res.end();
}).listen(4000,function(){
    console.log("server started");
});
```

Output



Q. What is the port number & Why use it?

Port number is unique identity to server means if we want to start or access any server we require a unique identity number to server known as port number.
Means two server cannot work on same port number
Because single computer can have more than one server and so if we think about network then if we want to access any machine or server machine then we can access using ip but if we want to access particular server from that machine then we required access it using port number

Q. What is the request and response parameter?

Request means when we visit any web page or click on hyperlink or send data via hyperlink using browser or submit form using browser to server page known as request
If we think about createServer() method it contain one parameter name as req which help us to access the data send by browser and process on it
Example: send database also.

Response: response means when the server page sends data to the browser or displays data on the browser called as response.

Example: res.write("welcome in web application using node");
This message display on web browser so it response data send by node server to browser

Package.json file

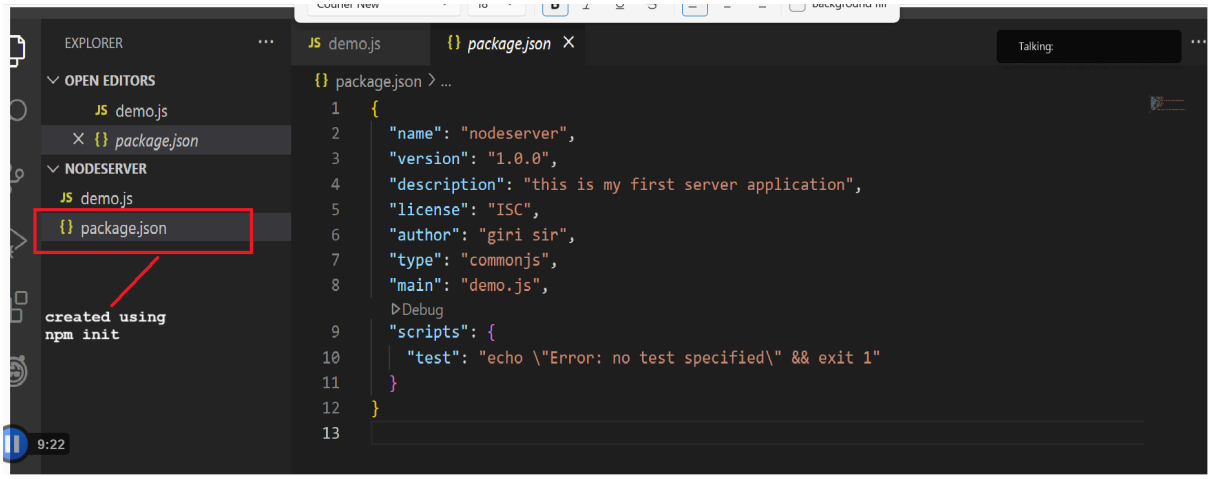
What is package.json?

Package.json file hold the all detail about project like as project name,project version ,repository name, command used in project ,installed packages like as mongodb,logging etc

How to create package.json file in project

If we want to create package.json file in project we have following command

Syntax: npm init



When we work with node js we required to create one more folder in project name as node_module
Node_module is folder where all libraries present or when we install any library then Library install under node_module folder.

If we want to create node_module folder project we have command

Example: we want to create a project to display output text in color.

Step

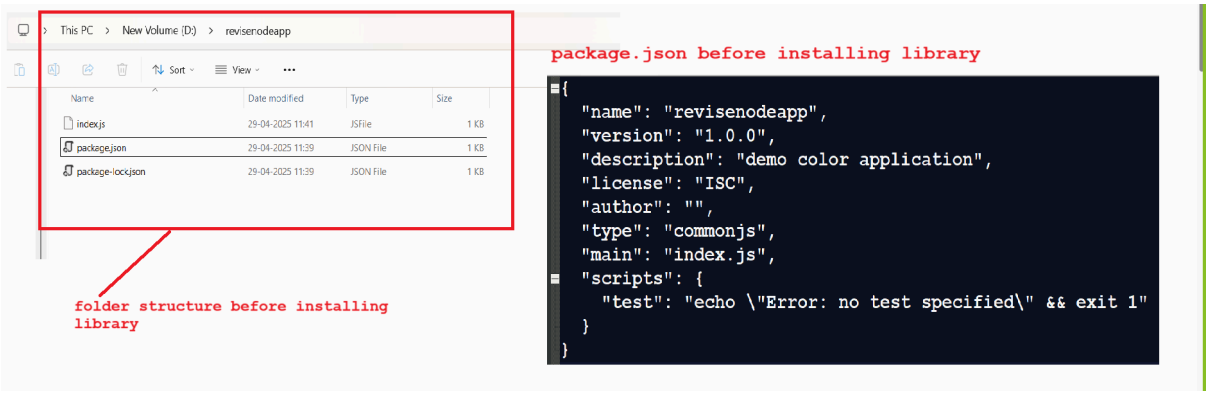
1. **Create folder for project**
2. **Create package.json file**
   Syntax: npm init
3. **Create package-lock.json**
   Syntax: npm install
4. **Create index.js file and write following logics**

   index.js
   console.log("good morning");
   Output

```
D:\revisenodeapp>node index.js
good morning

D:\revisenodeapp>_
```

If we think about above output we get output text in black color we want to display this output text in red color
So if we want to show output text in different colours we have one dependency or library provided node and we are required to install this library in the project and the name of the library is color.

Note: check the folder folder structure and package.json file before installing library



Note: if we want to install any dependency required for project we have generalize command or syntax

Syntax: npm install dependency name;

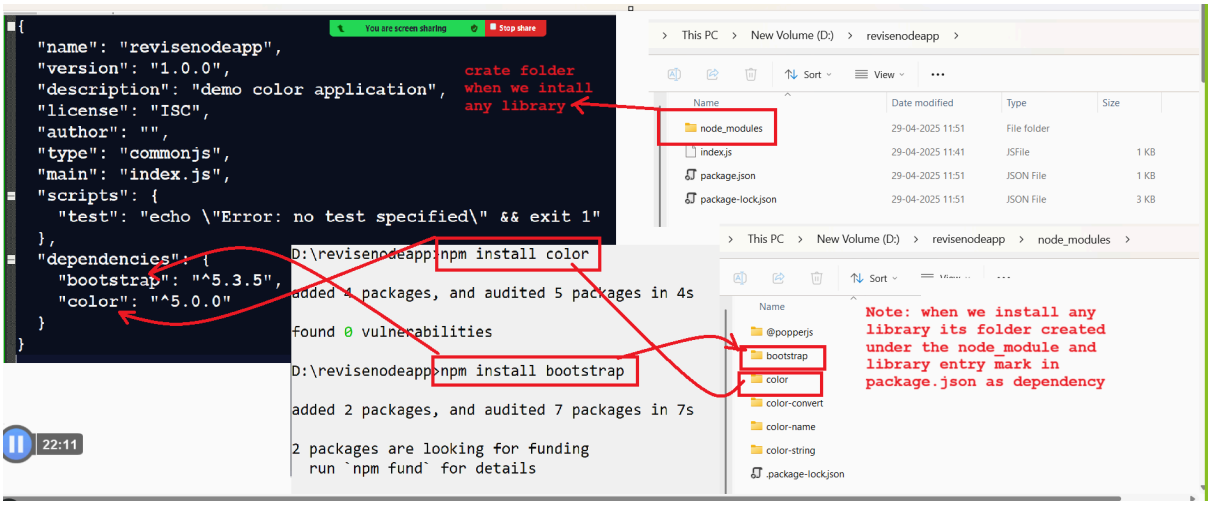
Or

Syntax: npm i dependency name;

So if we want to display output text in color we have color dependency

```
D:\revisenodeapp>npm install color

added 4 packages, and audited 5 packages in 4s

found 0 vulnerabilities

D:\revisenodeapp>
```

Check folder and package.json file after installing library



Note: package.json maintain the record of installed libraries because when we push project on give we not need to push node_module just we can push project
Without node_module and push package.json so when pull project anywhere and run command npm install then node_module created automatically with dependencies whose record maintain in package.json file

```
let colors=require("colors");
console.log("good morning".green);
```

```
D:\revisenodeapp>node index.js
good morning

D:\revisenodeapp>node index.js
good morning
```

Express JS

Q. What is express JS?

Express JS a framework of node which is used for web application development purposes and mobile applications.
It simplifies the development of server side applications by offering easy to use API for routing middleware , http utilities

Q. What kind of task can we develop using express JS?

---

1. Building RESTFUL API'S
2. Creating Single page , multi page and hybrid applications
3. Developing Server side logics for web application and mobile applications
4. Handing routing and middleware
5. Integrate view and rendering engines
6. Adding request processing middleware
   Etc

How to use express practically using node environment

Steps.

1. **Install express library**

```
D:\july 2024\testexpressapp>npm i express

added 66 packages, and audited 67 packages in 6s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

2. **Import express library in application**

3. **Use standard http methods to work with web applications to handle request and response.**

get() : get() method is used for access resources from a server or read resources from a server and send data to client as response
Get method normally use in url rewriting technique or use using path variable working rest API

post() : post() method is used for creating resources at server side
Example: suppose we have a form at client side like name,email, contact and we want to store it in the database table then we can use the post method at server side.

put() : if we want to update or modify the resource at server side then we have one standard resource at server side name as put.

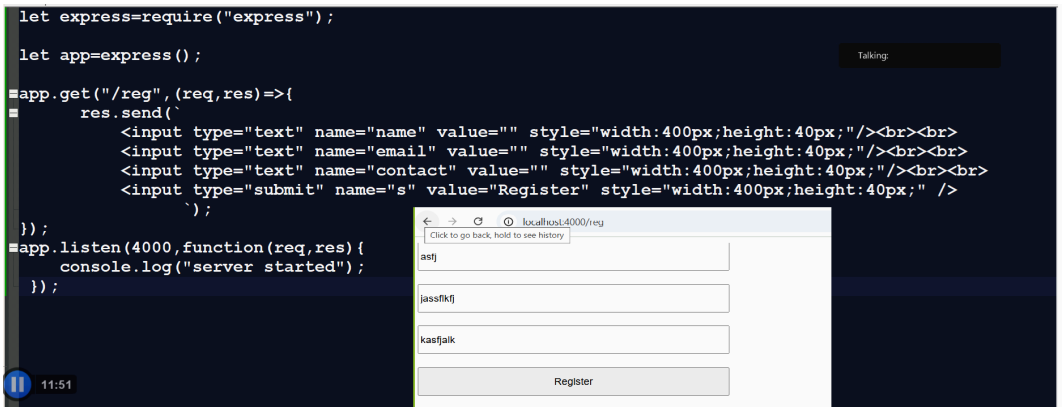
delete() : this method is used for delete resources from a server side.
patch(): this method is used for updating partial resources at the server's side.

Example: we want to create web page at server side name as home and display the message

```
let express=require("express");

let app=express();

app.get("",(req,res)=>{
   res.send("welcome on first web page using express");

});

app.get("/about",(req,res)=>{
           res.send("I am about page");
});

app.get("/service",(req,res)=>{
          res.send("I am service page");
});

app.get("/test",(req,res)=>{
            res.send("i am test page");
});
app.listen(5000,()=>{
           console.log("Server Started");
```

});

Note: using express JS we can generate the HTML content or html controls on a web page.



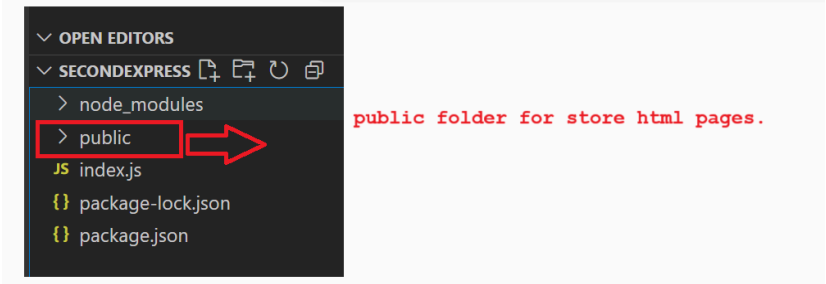
Note: if we think about above code we design UI using express JS but it is not standard way to design UI so we want to design ui pages using html or ejs and call it using express JS

How to create web page call it using express JS

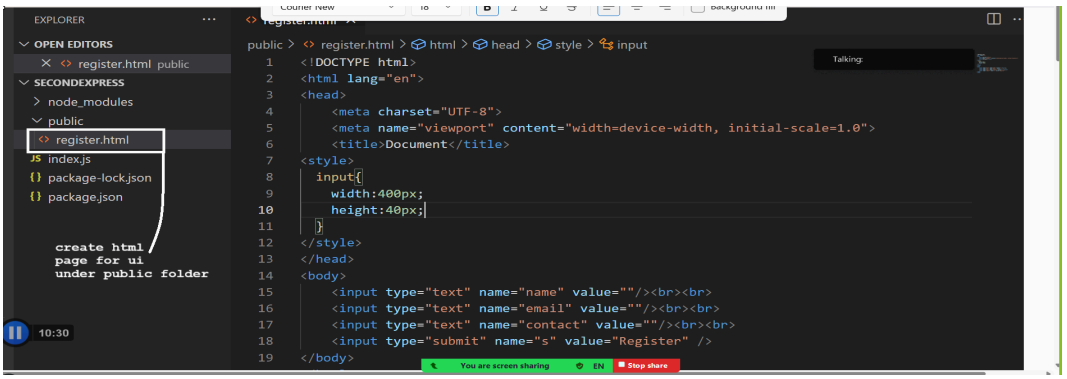
Steps.

---

1. **Create public folder in project**



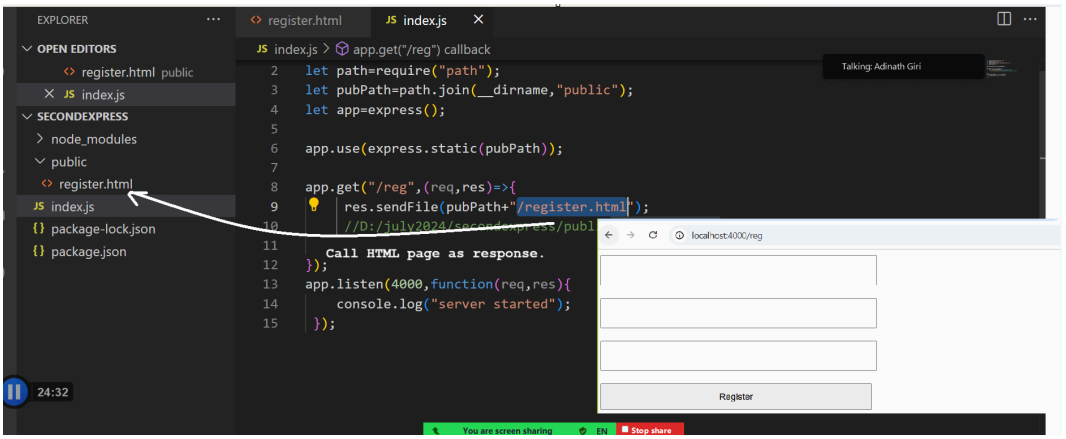
2. **Create html pages under the public folder**



3. Access path of public folder in express js code or js file

4. Access static pages or take permission to access statics or html pages in express
   Note: if we want to access static resources we have use() method of express and static method of express

5. **Call HTML page as response from express**
   If we want to call html page as response from express we have method of response object known as sendFile()



How to submit HTML form request to server page or express page

---

Step1: create project and generate package.json and package-lock.json file
Step2: install express library

```
D:\july 2024\testformsubmission>npm i express

added 66 packages, and audited 67 packages in 7s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Step3: create a public folder under the project and design the html page and save it.

Style.css

```
input{
  width:400px;
  height:40px;
}
```

Register.html

```
<html>
 <head>
  <title>registration form</title>
  <link rel="stylesheet" href="style.css" />
 </head>
 <body>
   <input type='text' name='name' value=''/>
   <input type='text' name='email' value=''/>
   <input type='text' name='contact' value=''/>
   <input type='submit' name='s' value='Register'/>
 </body>
</html>
```

4. Create index.js file or .js call html page as response to browser or client

index.js

```
let express=require("express");
let path=require("path");
let app=express();
let pubPath=path.join(__dirname,"public");
```

```
app.use(express.static(pubPath));
app.get("/",(req,res)=>{
   res.sendFile(pubPath+"/register.html");
});

app.listen(4000,()=>{
  console.log("server started");
});
```

5. Submit to server page or express url

If we want to submit html form to server page means we going send to request to server page from client page or client side page and for that we have to use form tag in HTML

Syntax:

<form name='formname' action='serverurl' method='GET/POST'
enctype='application/x-www-urlencoded or multipart/form-data'>

</form>

<form>: normally help us to submit html form data to server as request
name: name is identity of form at server side
action: a server page url where we want to send request or we want to accept data as request by server
Method: method means way to submit form to server
GET : when we submit form using get method then form data sent via URL of web page in the form of name and value pair.
When we send data via URL the technical name is query string

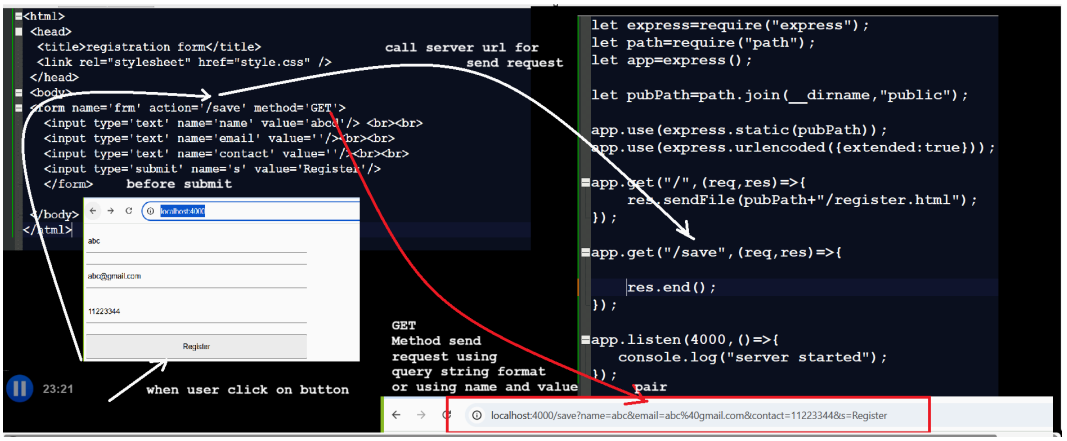
POST: when we submit a form using the post method then form data not sent via URL address bar or query string.
Form data sent via the body of the page means we can say the post method is more secure than the GET method.

enctype='application/x-www-urlencoded or multipart/form-data': this parameter decide how form data should access at server side

Note: application/x-www-urlencoded is default enctype means when we not specify enctype then form data by default application/x-url-encoded and this enctype decide access html form control name and return its value means when we submit request using name and value pair then we can use application/x-www-urlencoded we not specify it because it is default

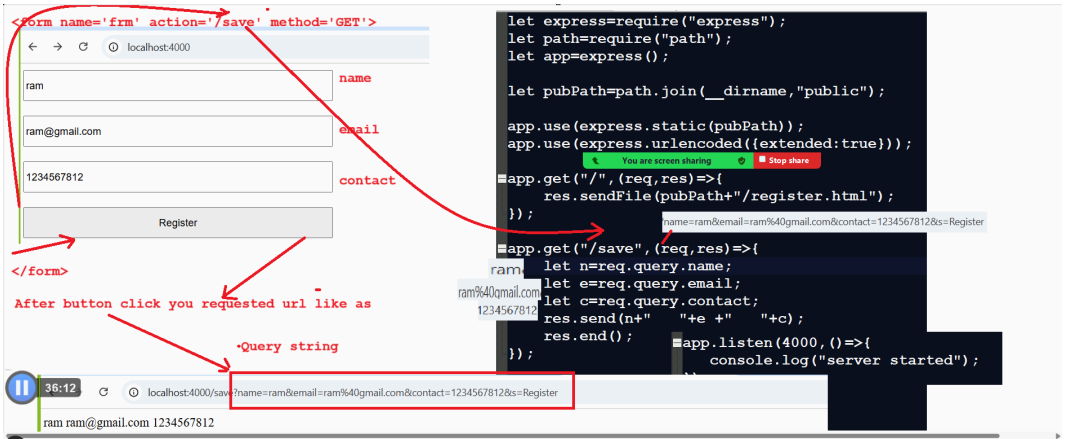
We have one more enctype known as multipart/form-data : this enctype we will use when we want to upload file to server

Example of form submission using GET



Note: when we submit form to server page or express page then we can access form data in express url function using req parameter
And for that we can use following syntax

Syntax: let variablename = req.query.requestparamname



Example with source code.

Style.css

```
input{
  width:400px;
  height:40px;
}
```

register.html

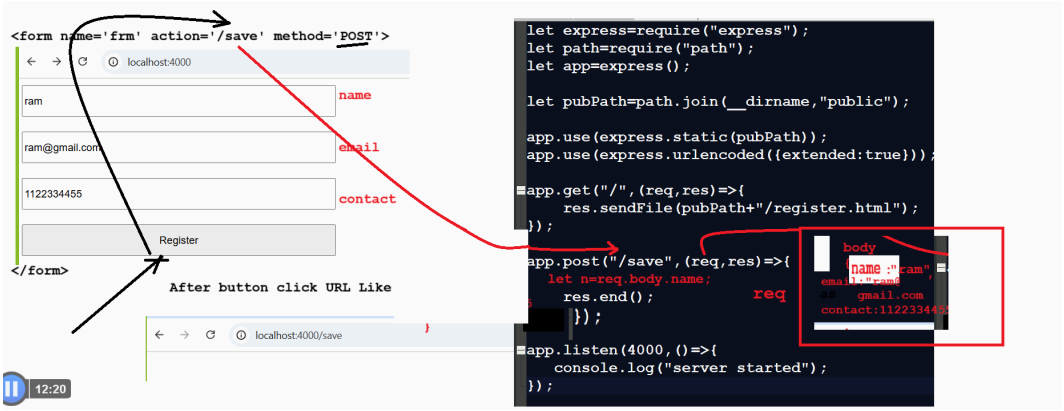
```
<html>
 <head>
  <title>registration form</title>
  <link rel="stylesheet" href="style.css" />
 </head>
 <body>
```

```
 <form name='frm' action='/save' method='GET'>
   <input type='text' name='name' value='abcd'/> <br><br>
   <input type='text' name='email' value=''/><br><br>
   <input type='text' name='contact' value=''/><br><br>
   <input type='submit' name='s' value='Register'/>
   </form>

 </body>
</html>
```

index.js

```
let express=require("express");
let path=require("path");
let app=express();

let pubPath=path.join(__dirname,"public");

app.use(express.static(pubPath));
app.use(express.urlencoded({extended:true}));

app.get("/",(req,res)=>{
    res.sendFile(pubPath+"/register.html");
});

app.get("/save",(req,res)=>{
          let n=req.query.name;
          let e=req.query.email;
          let c=req.query.contact;
          res.send(n+"   "+e +"   "+c);
    res.end();
});

app.listen(4000,()=>{
   console.log("server started");
});
```

Example: form submission using post method



Style.css

```
input{
  width:400px;
  height:40px;
}
```

Register.html

```
<html>
 <head>
  <title>registration form</title>
  <link rel="stylesheet" href="style.css" />
 </head>
 <body>
 <form name='frm' action='/save' method='POST'>
   <input type='text' name='name' value='abcd'/> <br><br>
   <input type='text' name='email' value='/><br><br>
   <input type='text' name='contact' value='/><br><br>
   <input type='submit' name='s' value='Register'/>
   </form>

 </body>
</html>
```

index.js

```
let express=require("express");
let path=require("path");
let app=express();

let pubPath=path.join(__dirname,"public");

app.use(express.static(pubPath));
app.use(express.urlencoded({extended:true}));
```

```
app.get("/",(req,res)=>{
    res.sendFile(pubPath+"/register.html");
});

app.post("/save",(req,res)=>{
   let n=req.body.name;
   let e=req.body.email;
   let c=req.body.contact;
   res.send(n+"      "+e+"   "+c);
    res.end();
});

app.listen(4000,()=>{
   console.log("server started");
});
```

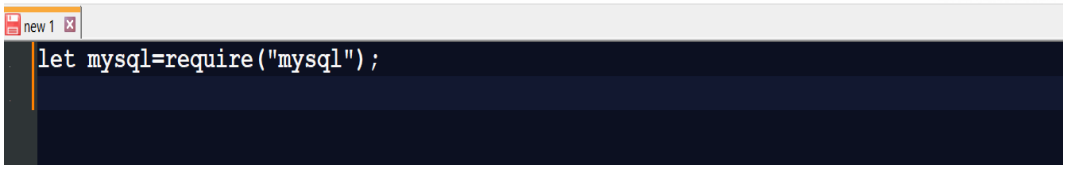
How to connect node with mysql

If we want to connect node with MYSQL we have some important steps.

1. **Install mysql library or dependencies**
   If we want to install dependency we have statement
   Syntax npm i mysql or npm install mysql

```
D:\july 2024\dbconnect>npm install mysql

added 11 packages, and audited 12 packages in 4s

found 0 vulnerabilities

D:\july 2024\dbconnect>
```

2. **import mysql dependency in application**



3. **Establish the connection with database**
   If we want to establish connection between node environment and mysql database we have function name as createConnection()
   And this function accept the JavaScript object as parameter which contain host name, username of database, password of database and database name shown in following screen shot.

```
let mysql=require("mysql");

let conn = mysql.createConnection(
 {
    host : "localhost or ip",
    user : "databaser user",
    password :"database password",
    database: "databasename"
 });
```

Note: if we think about the above syntax we are required to pass four parameters in JavaScript objects.

host: this parameter indicates location of the database means if your database is present on the same machine then we are required to write localhost or if your database is present on a different machine then we are required to provide an ip address.

user: here we are required to pass the username of the database provided by the user at the time of database software installation and if we think about mysql the default user name is root.

password: password of database provided by user at the time of installation

Database: here provide database name on which we want to perform operation

```
let mysql=require("mysql");


let conn = mysql.createConnection({
   host:"localhost",
   user:"root",
   password:"root",
   database:"junejulyadvjava"
});
```

If we want to check your connection is established or not we have connect() function provide by connection object or createConnection() method reference

```
let mysql=require("mysql");

let conn = mysql.createConnection(
 {
    host : "localhost or ip",
    user : "databaser user",
    password :"database password",
    database: "databasename"
 });

conn.connect(function(err){

});
```

this object catch error at run time
if occur between connection establishment

```
let mysql=require("mysql");
let conn = mysql.createConnection({
   host:"localhost",
   user:"root",
   password:"root",
   database:"junejulyadvjava"
});
conn.connect((err)=>{
    if(err)
    {  console.log("Error is  "+err);
    }
    else{
      console.log("database  is connected");
    }
});
```

Once we establish the connection between node and mysql we can work with mysql database or perform SQL operation like as insert/delete/update/select etc
So if we want to perform any operation on database we have query() function provided by conn object

```
connref.query("sql statement",function(err,result){

});
or
connref.query("sql statement",[runtime parameters],function(err,result){

});
```

Example:

Example: index.js

```
let mysql=require("mysql");
let conn = mysql.createConnection({
   host:"localhost",
   user:"root",
   password:"root",
   database:"junejulyadvjava"
});
```

```
conn.connect((err)=>{
   if(err)
         {  console.log("Error is  "+err);
         }
         else{
           console.log("database  is connected");
         }
});

conn.query("select *from player",function(err,result){
         if(err)
         { console.log("Operation Failed "+err);
         }
         else
         {
      for(var i=0; i<result.length;i++)
           {
                    console.log(result[i].name+"\t"+result[i].email+"\t"+result[i].contact);
           }
         }
});
```

Output

```
D:\july 2024\dbconnect>nodemon index
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
database  is connected
abc     abc@gmail.com   123456
name    email   contact
ram     ram@gmail.com   1234567891
```

How to insert record in database table using node

```
let mysql=require("mysql");
let conn = mysql.createConnection({
  host:"localhost",
  user:"root",
  password:"root",
  database:"junejulyadvjava"
```

```
});
conn.connect((err)=>{
   if(err)
         {  console.log("Error is  "+err);
         }
         else{
           console.log("database  is connected");
         }
});

conn.query("insert into player values('abcd','abcd@gmail.com','444444')",function(err,result){
         if(err)
         { console.log("Operation Failed "+err);
         }
         else
         {console.log("Record Save"+result);
         }
});
```

How to pass run time parameter to SQL Query using Node

```
let mysql=require("mysql");
let conn = mysql.createConnection({
  host:"localhost",
  user:"root",
  password:"root",
  database:"junejulyadvjava"
});
conn.connect((err)=>{
   if(err)
         {  console.log("Error is  "+err);
         }
         else{
           console.log("database  is connected");
         }
});
let pname="rajesh";
let pemail="rajesh@gmail.com";
let pcontact="4455667788";
conn.query("insert into player values(?,?,?)",[pname,pemail,pcontact],function(err,result){
         if(err)
         { console.log("Operation Failed "+err);
         }
```

```
        else
        {console.log("Record Save"+result);
        }
});
```

How to delete record from database table

```
let mysql=require("mysql");
let conn = mysql.createConnection({
  host:"localhost",
  user:"root",
  password:"root",
  database:"junejulyadvjava"
});
conn.connect((err)=>{
   if(err)
        {  console.log("Error is  "+err);
        }
        else{
          console.log("database  is connected");
        }
});
let pemail="abc@gmail.com";
conn.query("delete from player where email=?",[pemail],function(err,result){
        if(err)
        { console.log("Operation Failed "+err);
        }
        else
        {console.log("Record deleted"+result);
        }
});
```

How to update record in database table

```
let mysql=require("mysql");
let conn = mysql.createConnection({
  host:"localhost",
  user:"root",
  password:"root",
  database:"junejulyadvjava"
});
conn.connect((err)=>{
   if(err)
        {  console.log("Error is  "+err);
        }
        else{
          console.log("database  is connected");
```

```
        }
});
let pname="rama";
let pemail="ram@gmail.com";
let pcontact="8888888888";
conn.query("update player set name=?,contact=? where
email=?",[pname,pcontact,pemail],function(err,result){
        if(err)
        { console.log("Operation Failed "+err);
        }
        else
        {console.log("Record Updated "+result);
        }
});
```

EJS in express JS

EJS stands for Embedded JavaScript templating. Basically it is a template engine used by node js and EJS specially design for generate dynamic web page using HTML means using ejs file or ejs template engine we accept data send by server via express at client side and can generate dynamic view to end user

Steps to work with EJS

---

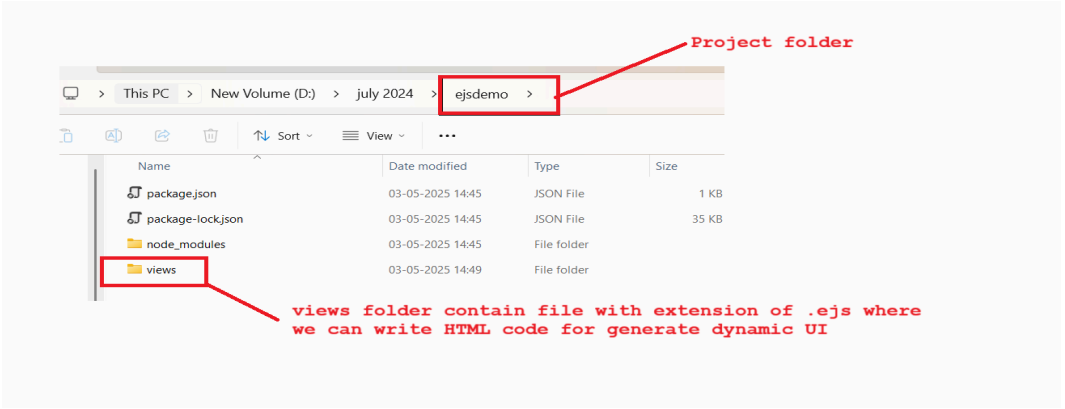
1. **Install EJS library or dependency**
   Syntax: npm i ejs - -save or npm i express ejs --save

```
D:\july 2024\ejsdemo>npm i express ejs --save

added 82 packages, and audited 83 packages in 10s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

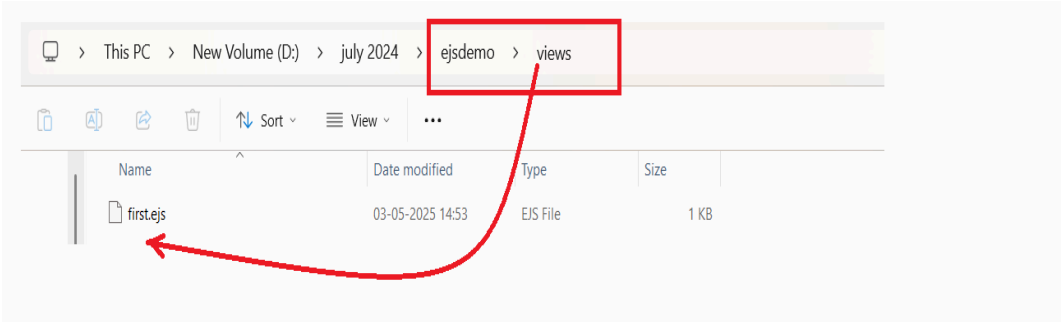
2. **Create .js file and set ejs engine with express**

```
let express=require("express");

let app=express();

//set EJS as template engine
app.set('view engine','ejs');
```

3. **Create views folder under the project folder**



4. **Create file with extension .ejs under the views folder**



Example with source code

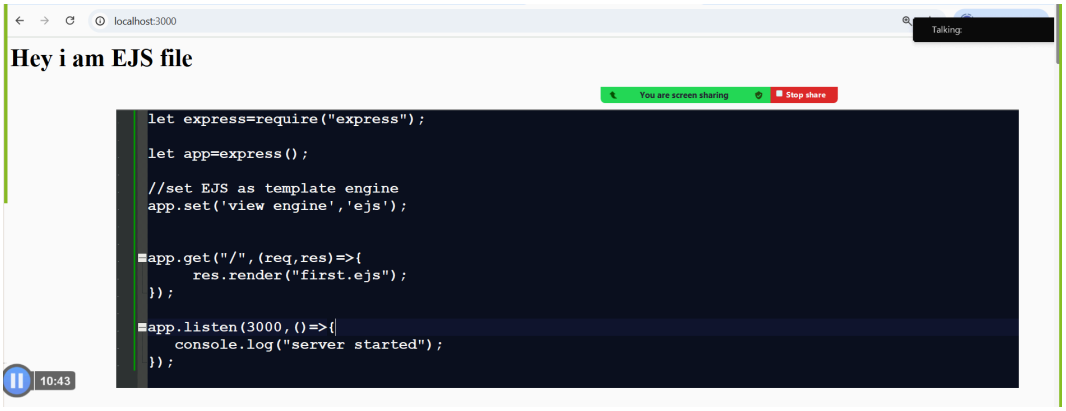
---

first.ejs

```
<html>
 <head>
   <title>i am ejs file</title>
 </head>
 <body>
   <h1>Hey i am EJS file </h1>
 </body>
</html>
```

5. **Call ejs file from express API**

If we want to ejs file from express API we have render() method res object



Note: same thing we did with HTML file so
Q. What is the difference between ejs and HTML file?

If use ejs then we can accept dynamically data from a server side and generate dynamic data at view page which is not possible using .html file

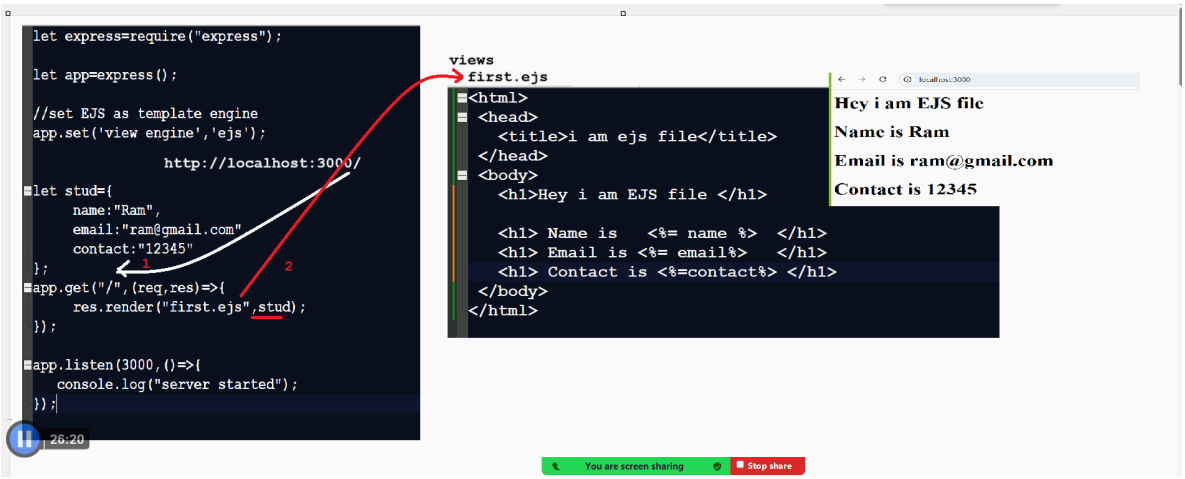
If we want to accept dynamically from a server side and display dynamic content on web page using EJS we have know the some expression and tags provided by EJS which help us to generate dynamic UI content at view page

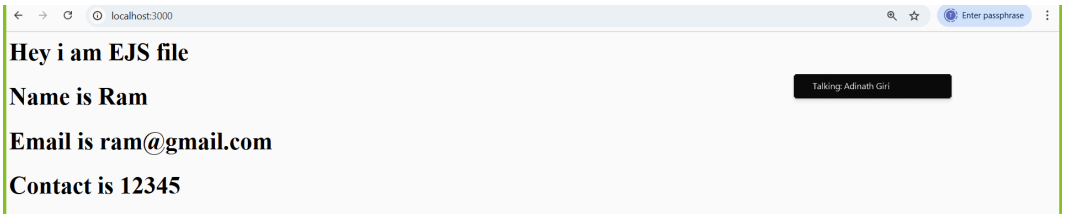
Express tag : <%= %> : this is used for display the output on web page like as document.write() of javascript
<% write here your logics like as loop, if etc %> : this express tag help us to write dynamic logic on web page using ejs

How to send data from express JS API to ejs file

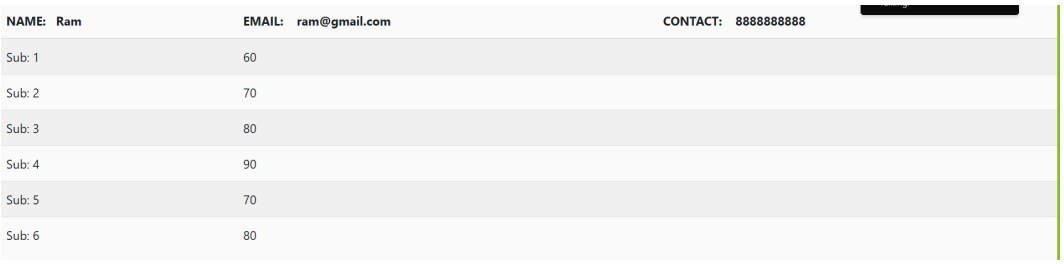
If we want to send data from node API to ejs we can send using JavaScript object format using key and value pair.
So JavaScript object from express API using render() method in the form of JavaScript object using key and value pair and we can access that data by using
Ejs expression tags by name

Example: we want to send student info from express JS API to ejs file means want to send data of student like as name,email and contact and show on web page using ejs file





Example with source code: Design page like as



NAME: Ram	EMAIL: ram@gmail.com	CONTACT: 8888888888
Sub: 1	60	
Sub: 2	70	
Sub: 3	80	
Sub: 4	90	
Sub: 5	70	
Sub: 6	80	

index.js

```
let express=require("express");

let app=express();

//set EJS as template engine
app.set('view engine','ejs');


let stud={
        name:"Ram",
        email:"ram@gmail.com",
        contact:"8888888888",
        marks:[60,70,80,90,70,80]
};
```

```
app.get("/",(req,res)=>{
    res.render("first.ejs",stud);
});

app.listen(3000,()=>{
  console.log("server started");
});
```

First.ejs

```
<html>
 <head>
  <title>i am ejs file</title>

   <!-- Bootstrap CSS -->
 <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/css/bootstrap.min.css"

integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ
8ERdknLPMO" crossorigin="anonymous">



 </head>
 <body>
  <table class="table table-striped">
 <thead>
   <tr>
     <th scope="col">  NAME:&amp;nbsp;&amp;nbsp; <%=name%>  </th>
     <th scope="col">  EMAIL: &amp;nbsp;&amp;nbsp; <%=email%></th>
          <th scope="col">  CONTACT: &amp;nbsp;&amp;nbsp; <%=contact%></th>
   </tr>
 </thead>
 <tbody>
   <%
           marks.forEach((item,index)=>{
          %>
            <tr>
             <td>Sub: <%=(index+1)%> </td>
                  <td colspan='2'> <%=item%></td>
            </tr>
          <%
            });
          %>

 </tbody>
</table>
```

```
 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi
6jizo" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.3/dist/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvC
WPIPm49" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.1.3/dist/js/bootstrap.min.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5
stwEULTy" crossorigin="anonymous"></script>
 </body>
</html>
```

Mini Project

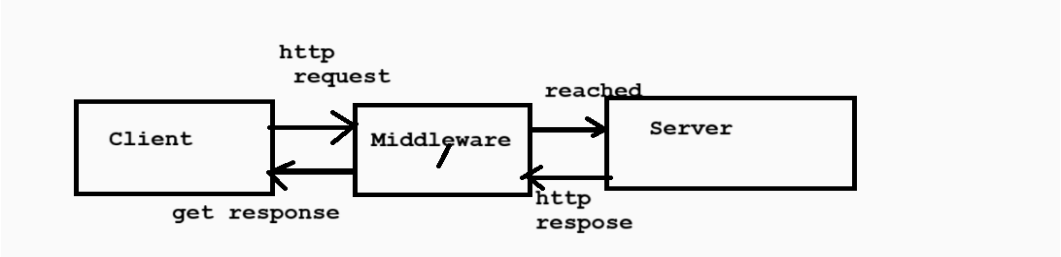
Tech Stack : HTML,CSS,bootstrap,JavaScript(validation),ejs
Server: Node, express, MYSQL ,AJAX

URL rewriting

URL Rewriting is technique where we can send requested data via URL in the form of query string or name and value pair

Middleware concept in node

Middleware in express to function that process request before reaching the router handlers. This function can modify the request and response object and end the request and response cycle or the next middleware function.
Middleware functions are executed in order by definition.
Using middleware you can perform task like as authentication ,logging or error handling etc



```
app.use((req,res,next)=>{

 //logic
//request back or process
next();
});
```

(req,res,next)=>{} : this is the middleware function where you can perform action on the request and response objects before the final handler is executed

next(): this function is called to pass control to the next middleware in the stack if the current does not end the request and response cycle.

What Middleware does in express?

---

1. **Execute the code :** middleware can run any code when request is received
2. **Modify Request and Response:** means using middleware you can modify the request object and response object.

3. **End the Request and Response :** Middleware can send a response to the client ending the cycle.
4. **Call the next middleware** : middleware can call next() to pass control to the next function in the middleware stack.

How Middleware works in express JS?

In express js middle functions are executed sequentially in the order they added to the application

1. Request arrives at the server
2. Middleware functions are applicated to request one by one
3. Each Middleware can either:
    a. Send a response and end the request and response cycle
    b. Call next() to pass control to the next middleware

4. If no middleware end the cycle the router handler is reached and final response send to client

Types of Middleware

---

1. **Application level middleware :** Application level middleware is bound to the entire express application using app.use() or app.METHOD()
   It execute for all routes in the application ,regardless of the specific path of Http method

This type of middleware is commonly used for tasks like logging,body parsing ,authentication checks or setting header for every common request.

```
//parse json data for every incoming request
app.use(express.json());

app.use((req,res,next)=>{
  console.log(req.method);
console.log(req.url());
next();
});
```

2. **Router-level middleware :** Router level middleware is applied to a specific route instance using router.use() or route.METHOD(). It only applies to routers defined within that particular router, making it perfect for modular applications where the middleware is only relevant to a specific router group.
   This type of middleware is often used to group related routes (e.g all routes related to authentication or user management) apply middleware logic to them.

```
let router=express.Router();

router.use((req,res,next)=>{
 console.log("router specific middlware");
 next();
});

router.get("/dashboard",(req,res)=>{
  res.send("router executed");
});
```

3. **Error Handling Middleware :** Error handling middleware is a special type of middleware used to catch and respond to errors during the request and response cycle. It is defined with four parameters
   a. **Err, b. Req, c. res d. Next**

This middleware is essential for sending a consistent error response and avoiding unhandled exceptions that might crash server

```
app.use((err,req,res,next)=>{
   console.log(err.stack);
  res.status(500).send("something wrong");
});
```

4. **Built in Middleware :** Express provides built in middleware to help with common tasks like serving static files or parsing data.

   Example: express.static(),express.json() etc

express.static(): serves static files like html,css or images and express.json() help parse incoming JSON data.

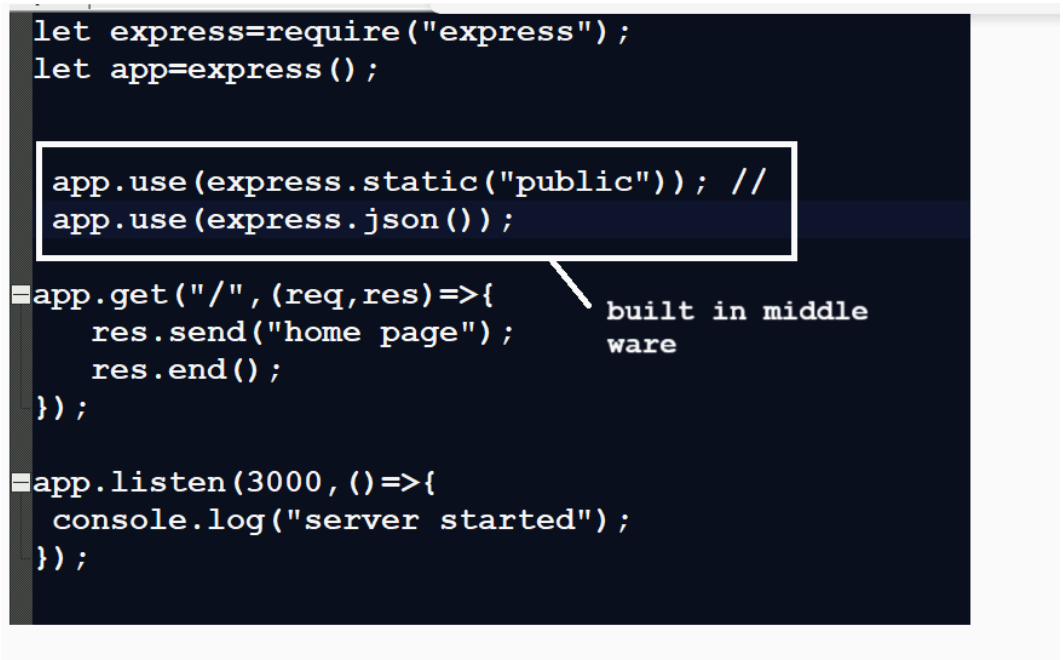
```
app.use(express.static("public"));
app.use(express.json());
```

5. **Third party middleware : Third party middleware is developed by external developers and packaged as npm modules. These middleware packages add additional functionality to your application.**
   Like as request logging, security features or data validation etc
   Example: morgan middleware is used for logs the http request etc

Example of application level middleware

```
let express=require("express");
let app=express();
app.use((err,req,res,next)=>{
   console.log(err.message);
   res.status(err.status || 500).json({
            success:false,
            message:err.message || 'Internal server error'
   });
  res.end();
});

app.get("/",(req,res)=>{
   res.send("home page");
   res.end();
});
app.get("/error",(req,res,next)=>{
         let err=new Error("Something went wrong");
         err.status=500;
         next(err);
});
app.listen(3000,()=>{
 console.log("server started");
});
```

Example of built in middleware



Third party middleware

If we think about third party middleware we required to installed it manually by npm

1. Morgan

Example with source

```
let express=require("express");

let morgan=require("morgan");
let app=express();


 app.use(morgan('dev'));
 app.use(express.static("public")); //
 app.use(express.json());

app.get("/",(req,res)=>{
   res.send("home page");
   res.end();
});

app.get("/about",(req,res)=>{
            setTimeout(()=>{
                    res.send("Hey i am about page");
                    res.end();
          },2000);
```

```
});
app.listen(3000,()=>{
 console.log("server started");
});
```

2. **Body parser**

```
D:\july 2024\middlewareapp>npm install body-parser

up to date, audited 90 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

3. **Cors :**
   The role of CORS (Cross-Origin Resource Sharing) middleware in Express.js is to manage and control which web applications, running on different origins (domains, protocols, or ports), are allowed to access resources from the server.

```
D:\july 2024\middlewareapp>npm install cors

added 2 packages, and audited 92 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

4. **Express-session :** this middleware help us to work with session or manage session using express application

```
D:\july 2024\middlewareapp>npm install express-session

added 6 packages, and audited 98 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

5. **Cookie-parser: help us to pass cookie header**

```
D:\july 2024\middlewareapp>npm install cookie-parser

added 2 packages, and audited 100 packages in 3s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

How to handle the session in NODE /express JS

Q. What is a session?

Session is a communication period between client and server over the network called as session or user login to user logout time or process called as session

Q. Why do we need to use sessions?

Basically http is a stateless protocol that means every request of a client is considered as a new client at server side even if the same client sends multiple requests . This behaviour of http protocol is known as stateless protocol.
If we want to convert stateless behaviour of http protocol to stateful behaviour then we can use session in web application.
When we use session in web application then server maintain the state of client i.e create one session id and store in header of client and one copy maintain at server side and when client revisit to server with new request then server verify session id of that client present on server or not if session id of that client present at server then consider it is existing client if session id not present on server then consider it is new client

Q. Can you give an example where I can use sessions in my project?

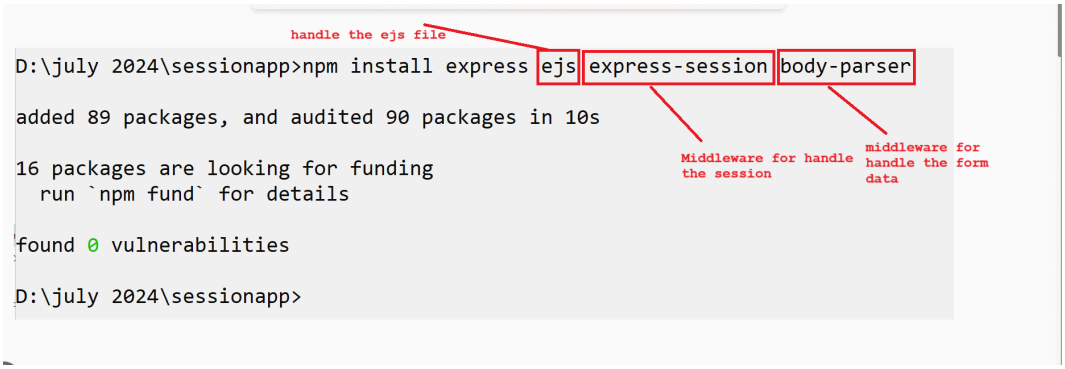
---

1. **Session help us to main user privacy :** if we session in web application then we can maintain user privacy means we can access of data to user according to his login
   Means one user cannot see the data of another users if we session because session is separate every user or client
2. **Session help us to store data temporary at client side**
3. **Session can provide security to user page**
4. **Count the active user or login user on portal**
   Etc

How to use a session practically using express?

Steps to use session practically

1. **Install the express-session middleware dependency /library**



2. **Import expression-session library**

```
let session=require("express-session");
```

3. **Configure session :** if we want to handle the session we require to configure session in the form of JavaScript object

```
app.use(session({
    secret:"secretkey",
    resave:false,
    saveUnitialized :true
}));
```

secret: A string (or array of string) used to sing the session ID. this prevents tampering
resave: if true session is saved on every request event it wasn't change so usually set false
saveUninitialized: if true session that are new but modified will be save useful for implementing login flows or tracking visit
cookie(optional): you can pass additional cookie options like maxAge,secure etc
How to generate secret key for security

D:\july 2024\sessionapp>
node -e "console.log(require('crypto').randomBytes(64).toString('hex'))"

Output:
95856526a58e003ac008930a9837c871e8acc97b2a2ea685655d24bbbce5a95713ba116440743155b1ffbb6fb34014573acfbfa9b6fffa18f5f640a5eff13395

Note: if we think about session object internally for every client /browser server generate separate session object to us and every session object has different session id means every client has different session

If we want to check session id or get session id of client we have following syntax

Syntax: let sessionid=req.sessionID;

Example with source code

```
let express=require("express");

let session=require("express-session");

let app=express();

app.use(session({

secret:"95856526a58e003ac008930a9837c871e8acc97b2a2ea685655d24bbbce5a95713ba
116440743155b1ffbb6fb34014573acfbfa9b6fffa18f5f640a5eff13395",
        resave:false,
        saveUnitialized:true
 }));

app.get("/testsession",(req,res)=>{
         res.send("Hey client your session id "+req.sessionID);
         res.end();
});
app.listen(3000,()=>{
  console.log("Server started");
});
```

4. **Store data in session**

When we store data in session means every session object has different means every client has different session means every client has different data

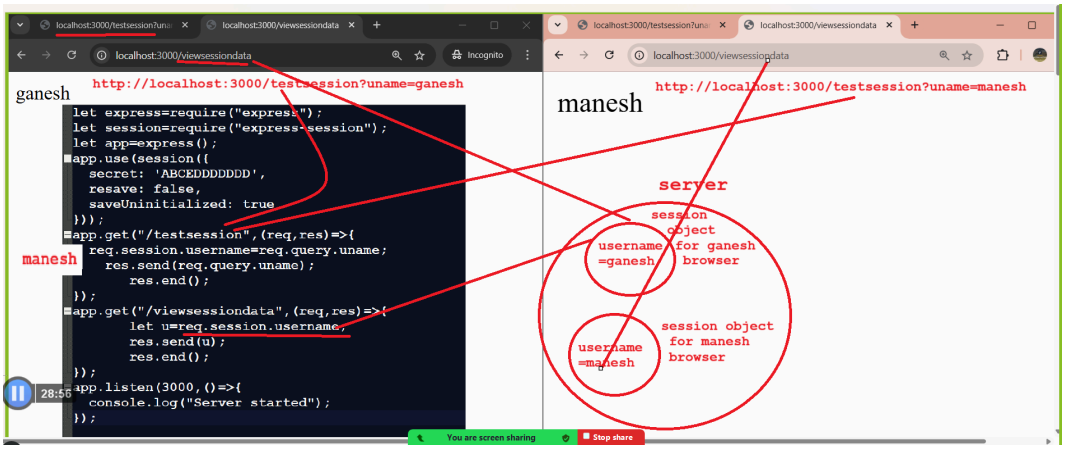
Syntax: req.session.key=data
Note: we store in session using key and value pairs

Example: req.session.username=”ABC”
.

5. **Retrieve data from a session**

If we want to retrieve data from a session we have following syntax

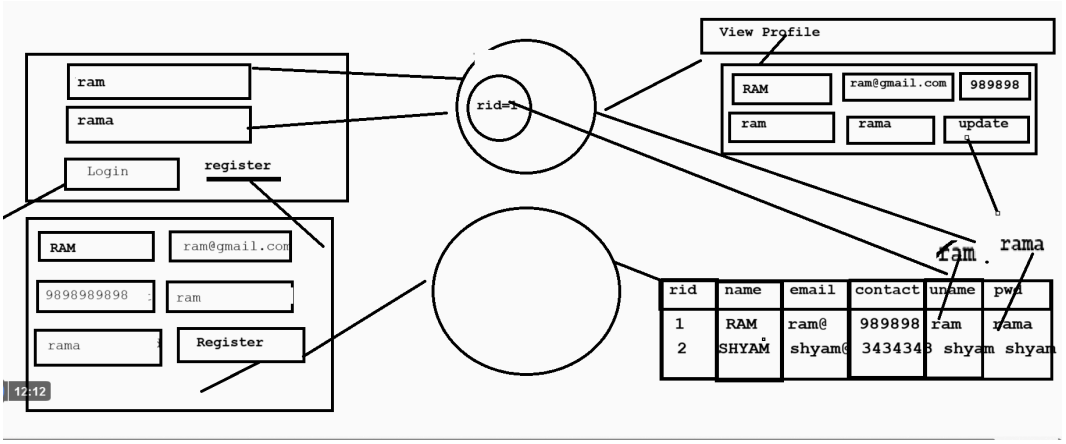
Syntax: let variablename=req.session.key;



Example with source code

```
let express=require("express");
let session=require("express-session");
let app=express();
app.use(session({
  secret: 'ABCEDDDDDDD',
  resave: false,
  saveUninitialized: true
}));
app.get("/testsession",(req,res)=>{
  req.session.username=req.query.uname;
          res.send(req.query.uname);
            res.end();
});
app.get("/viewsessiondata",(req,res)=>{
            let u=req.session.username;
            res.send(u);
            res.end();
});
app.listen(3000,()=>{
  console.log("Server started");
});
```

Assignment goal



When we design any application or project then we should have to consider following thing

1. **Requirement Gathering :**
    a. Functional Requirement : What should the system do?
    b. Non Functional Requirement : performance ,scalability ,security etc
    c. Stakeholder input: who will use it and what do they need?
    d. Constraints : Timeline, budget,technology, legal etc
2. **System Architecture (HLD)**
    a. **Architectural style : Monolithic, microservices , serverless etc**
    b. **Technological stack:** Front End, Backend , Database, third party etc
    c. **Data Flow and Module Interaction : how components talk to each other**
    d. **Deployment model : cloud, on-premises?**

3. **Database Design:**
    Data Modeling: ER diagrams, entity relationship ,normalization
    Storage technology: SQL,NOSQL,distributed DB etc
    Data Access Technology: ORM or SQL etc
4. **API design :**
    API Contracts: REST ,GraphQL etc
    Authentication/authorization: JWT , OAuth,API keys , session etc
5. **Low Level Design (LLDP)**
    Class diagram and interfaces
    Design pattern: singleton,factory,adapter, builder etc
    Object modeling: SOLID principles ,DRY/YAGNL etc
6. **Security consideration:**
    a. Input validation and sanitization
    b. Data encryption
    c. Secure deployment practices like as https,firewalls
7. **Performance and Scalability**
    a. **Caching strategy**
    b. **Load balancing**
    c. **Horizontal and vertical scaling**
    d. **Asynchronous processing**
8. **DevOps & CI/CD**
    a. **Version control**

    b. **Automated testing**
    c. **Continuous integration and deployment pipelines**
    d. **Infrastructure as Code(Terraform,docker etc)**
9. **Monitoring & Logging**
    a. **Health check**
    b. **Centralised logging**
    c.
10. Testing Strategy
11. **Documentation**
    a. **API documentation (Swagger/Postman)**

1. Requirement understanding
2. High level design
3. Low level design

Requirement understanding
Functional requirement : A functional requirement describe what a system should do , the specific behaviour , functions or problems it must support to full fill user needs or business rules

Example:
1. **User Registration :** the system must be allow use to register using their name,email and contact, username and password
2. **User Login :** The system must be allow use to log in using valid username and password
3. **Password Security : password must be stored in hashed format**
4. **View Profile :** After logging the user must be able to view their profile details
5. **Session Management :** system should maintain the session after login until logout or token expiry
6.

Non-Functional requirement : A non functional requirement defines how a system performs its function rather than what it does.
This type requirement describe the quality attributes of the system
Performance, usability , security, reliability
Example:
Performance: the login response time should under 2 seconds
Scalability: The system must be supported to 10000 concurrent users at time.
Security: Password must be encrypt by using hash etc

High level Design

Focus : Architecture , components, technology etc
Audience: Architect , senior developers , stakeholders etc
Details: No code level details, focus on module and data flow
Purpose : Guide to LDD and ensure all pieces together in picture

High level Design

Architecture: monolithic ,MVC design pattern

Database : Single table

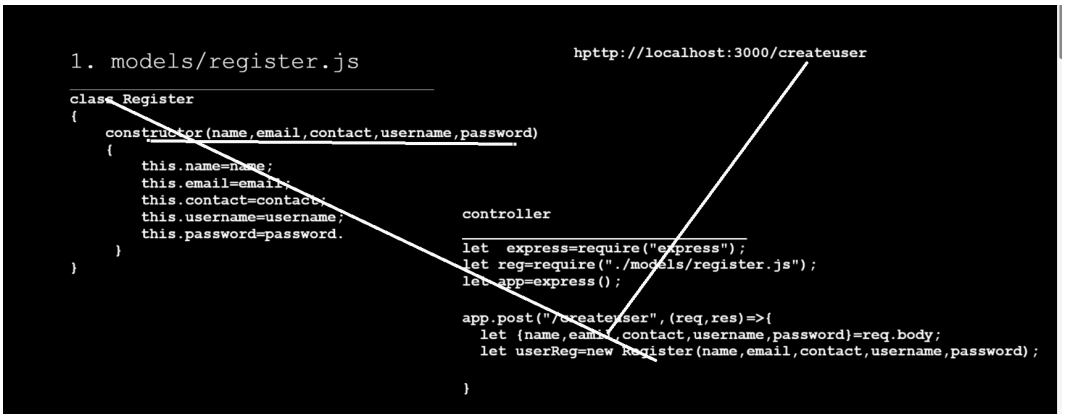
Low level design:

Focus: internal logic , class structure , function level design

Audience : developers and testers

Details: class diagram , database query , API, algorithm

Example:



Q. What is MVC?

MVC stands for Model View Controller basically MVC is standard design pattern in web application development.

M: Model : Model are the classes which is used for to store data and work with database as well as model help us to accept the data send by view and store in model object and work with db via controller

V : View : view means user interface from user can provide input and get result

If we think about view we can develop view using react,angular,ejs , html etc

C : Controller : controller is used for accept the data as request send by view and store in model and process on it and pass to service and database

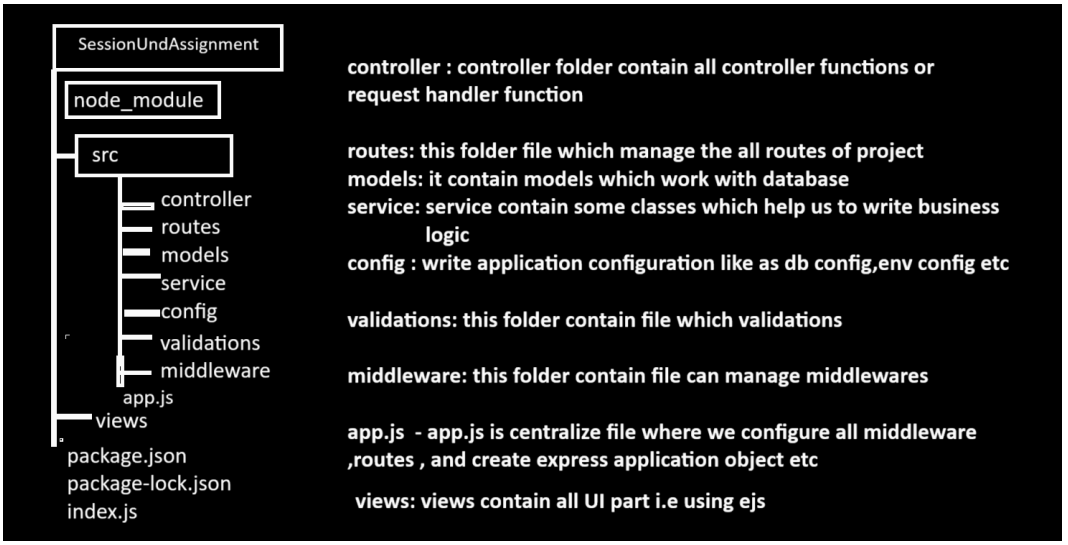
If we think about node we use request handler function as controller

Q. Why use MVC?

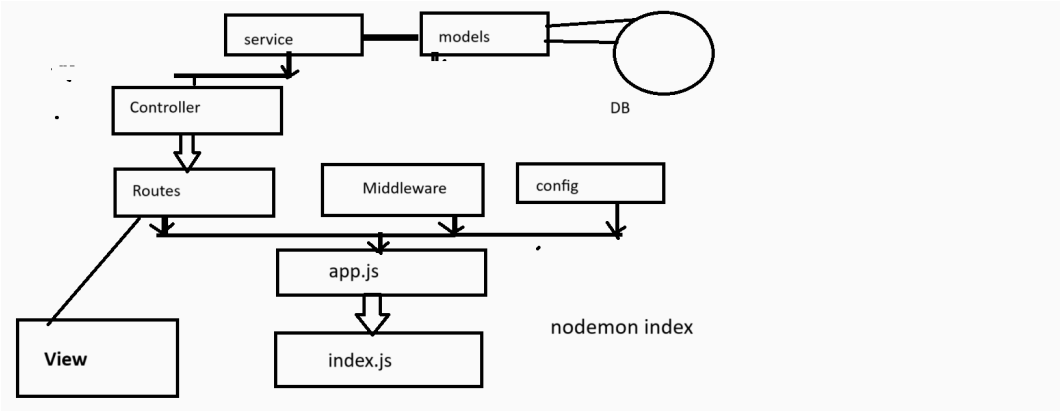
The major goal of MVC is code separation means using MVC we separate designing as well as business logic and database logic

Start Development of Project

If we want to work with node project we have some standard folder structure



How internally folders and files communicate with each



Session Management Concept

Mini Project Design by using handling

Goal : we want to design application for user registration , login and after login view own profile and update profile

Backend database : MYSQL
Express JS
Front end: ejs + HTML +CSS+Boostrap

MVC Architecture
Node JS standard folder structure

1. Session
2. JWT token
3. REST API

Cookie

Q. What is a cookie?

A Cookie is small piece of data that a website stores on a user's device (like as computer or smartphone) through the web browser
Cookie help website remember information about the user such as login status , preferences or items in shopping cart
Basically cookie is part of session handling means cookie create separate for every user or browser
Etc

Q. What is the difference between cookie and session?

---

1. Cookie stored its data at client side i.e in browser and session store its data at server side
2. Cookie can store its data for longer period at client side after browser close and session can vanish its data when client logout or close the browser or client application

Note: if we use the cookie and if user disable its cookie option at browser side then cookie vanish automatically when close the browser means work as session
If we use the cookie then try to avoid store confidential or important information in cookie because cookie at client side so there may be possibility of data leakage or privacy break

There are two types of cookie

---

1. **Session based cookie/temporary cookie :** cookie without age life called as session cookie or temporary means these types of cookie vanish after browser close
2. **Persistent cookie :** persistent cookie means a cookie with age life means when we create a cookie we set its age called as persistence cookie.

Example or Real life example of cookie

Online shopping (Amazon)

Suppose in online shopping we create cart and add data or item in cart but we leave the website without placing order or checkout and Later after some time if we visit again website e.g amazon then we get your cart which we created
So this type of data stored in cookie and set its age by server

How does amazon remember the user or facebook remember the user

---

So if we think about above scenario cart information and user information stored in cookie and when user revisit using its browser then server check the user info and cart info at server side using cookie object and if server found data then consider it is existing user and its data otherwise create new cookies.

How to use cookies in express JS

If we want to use cookie in express JS we have to use cookie-parser middleware
This middleware allows the express server to set , read or clear cookies in http request and response.

Steps to work with cookie in Express

---

1. Install cookie parser

```
D:\july 2024\cookieapp>npm install express ejs body-parser cookie-parser

added 84 packages in 19s

16 packages are looking for funding
  run `npm fund` for details

D:\july 2024\cookieapp>
```

2. **Import cookie parser in .js file**

```
let express = require("express");
let bodyParser=require("body-parser");
let cookieParser=require("cookie-parser");
```

3. **Use cookie as application middleware**

```
let express = require("express");
let bodyParser=require("body-parser");
let cookieParser=require("cookie-parser");

let app=express();

app.use(cookieParser());   set cookie parser as application middleware


app.listen(3000,()=>{
   console.log("server started");
});
```

4. **We can store data in cooke by using request object**

```
res.cookie(key,value,
 {
    maxAge: age in seconds, //1 year
    httpOnly:true | false
});
```

Once we send cookie as response to server page then we can read cookie data as request as server side
If we want to read cookie at server side we have to use following syntax

Syntax:

```
let variablename=req.cookies.key;
```

Authentication and Authorization

What is authentication?

Authentication is process of verifying who a user is

Example: suppose you are going to the airport . At the entrance ,you show your ID and ticket to prove who you are. The Security checks your identity this is called as authentication

Technical example

If we think about application oriented process when user input his username and password for login in application then first we verify its username and password present in db or not after that we decide user should access system or not called as authentication

What is authorization?

Authorization is a process of verifying what an authenticated user is allowed to do.

Example: Suppose consider after verifying your ticket and ID card on airport gate your inside of airport ,you can only enter the gate listed on your ticket ,not any gate you want. Means a ticket gives you access to a specific area.
This process is known as authorization

Technical example: suppose you input username and password in system and system verify username and password as well as your role and according to role decide what facilities you can access from system

Example: suppose consider we are developing application private training institute we have multiple users like as admin,teacher,student, etc so teacher cannot delete record of student teacher can see the record of student but admin can delete or view the record of students

How many ways to perform authentication in a web application ?

1. **Session based authentication**

   Session based authentication suitable for traditional web applications means without REST API
   If we think about node we can manage session by express-session
2. **JWT (JSON Web token) :** MOST common in REST API
   If we want to JWT token in node we have libraries like as jsonwebtoken, passport-jwt etc

3. **OAUTH 2.0 :** for social login (Google, Facebook,GitHUB) etc
   If we want to use this authentication in node we have libraries passport, passport-google-oauth2.0 etc

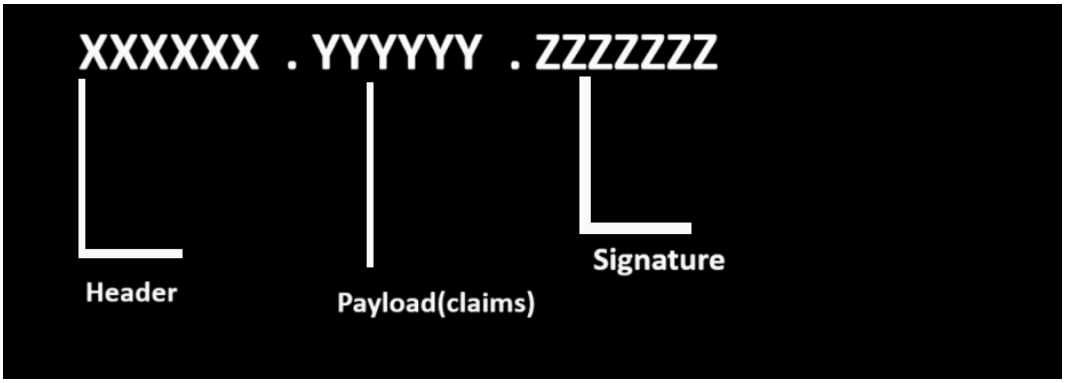
Now we want to discuss about JWT token

Q. What is JWT?

JWT(JSON web Token) is a compact , URL-safe token format used to represent claims/data between parties. It is often used for authentication and authorization in web apps.

Structure of JWT token

A JWT is made up of three parts separated by dots



Q. What is the header?

Header contain algorithm used with JWT token and type of token also

Q. What is payload?

This contains data or claims

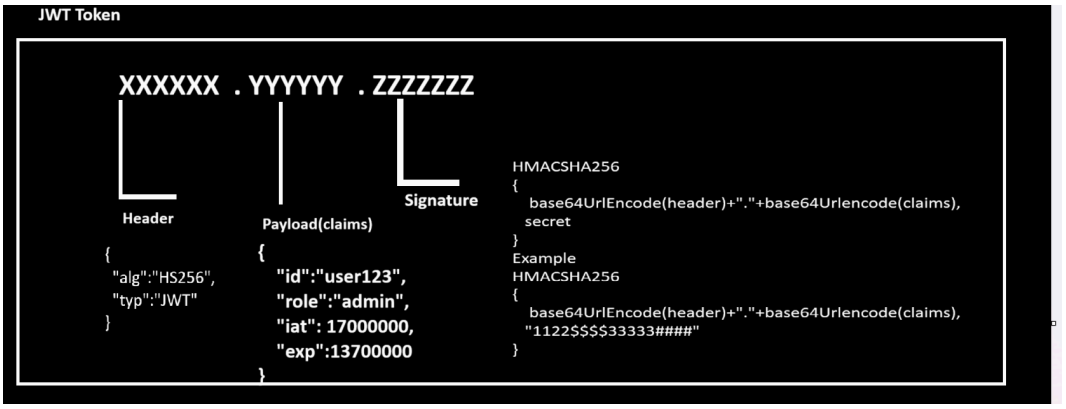
Id,role: custom data and this data may be change according to user
iat: issued time means when token generated
exp: expiration time of token

Q. What is a signature?

This is created by using the header and payload and a secret key
Signature is used to provide security to JWT tokens . Signature ensure the token is not tempered

```
HMACSHA256
{
   base64UrlEncode(header)+"."+base64Urlencode(claims),
  secret
}
Example
HMACSHA256
{
   base64UrlEncode(header)+"."+base64Urlencode(claims),
  "1122$$$$33333####"
}
```

Complete structure of JWT Token



What are the benefits of JWT token over session?

---

1. **Stateless (No server memory usage) :** if we use plain session then server need to create object for every session or client at server side and can store its information but if we think about JWT token not need to create space in server memory because JWT token stored in client header and it help us scale horizontally means JWT token work with multiple servers
2. **Better for API and SPA (Single page application)**
   JWT token ideal for single page application like as React, Angular etc
   Commonly used with RESTFUL API and mobile apps
3. **Easy Cross Domain or Mobile authentication**
   JWT can be used in cross domain authentication
   Works well for mobile apps ,where session or cookies very less practically
4. **Compact and transportable : JWTs are compact (base64 encoded) and can be easily passed in :**

   HTTP headers(Authorization : Bearer <token>

Cookie
Query String (not recommended)

Q. What is the difference between JWT token and session?

Feature	JWT	Session
Storage	Client side(usually in local storage or cookies)	Server side (in memory or Redis or database)
Scalability	High (stateless)	Limited(stateful,need centralized session store)
Cross - platform	Easily work with mobile apps, SPA,APIs	More suitable for tradition web app
API ready	Design for RESTFUL,stateless API	Not fit for REST API
No server memory usage	Server does not store JWT token data	Server needs to store session per user
Decentralized Auth	Great for microservices ,no shared session store needed	Requires centralized session management

What are the limitation of JWT token

1. **Security Risk :** if token is stolen and it has not expired the attacker can use it
2. **Token size:** JWT's are larger than session ID which can affect performance if overloaded
3. **Store management :** client must securely store the token(ideally in cookies , not local storage)

When to use?

Use Case	Use JWT	Use Session
REST API,Mobile App,SPA	**yes**	**Not ideal**
Traditional web app with a server side rendering	**Less ideal**	**yes**
Need for horizontal scaling/micro services	**yes**	**Complex**

How to use JWT token practically in node or express

If we want to use JWT token practically we have to use following dependencies

JWT - jsonwebtoken ,
Password encryption using hashing technique : bcryptjs

OAUTH 2.0

OAUTH2.0 is an authorization framework that allow third party applications to obtain limited access to user resource without expecting their credential
It is widely used in system where users grant access to their data store on one service (like as google,facebook or GITHUB) to another application

Now we want to create application login via google or gmail

Steps

1. **Open google console window**
2. **Create project using node**
3. **Add the following dependency in project**
    a. **Express**
    b. **Dotenv**
    c. **passport**
    d. **express-session**
    e. **passport-google-oauth20**

```
D:\july 2024\oauthapp>npm install express passport  dotenv express-session passport-google-oauth20

added 83 packages, and audited 84 packages in 12s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\july 2024\oauthapp>
```

Mini Project

Online Shopping

High level design

Objective of this project/Requirement gathering

1. User to browse product /search project
2. Add to Cart
3. Place orders
4. Make Payments
5. Track shipments

Architecture Components

---

1. **Client layer (Frontend) :**
   Web or mobile app (HTML/CSS/EJS ,React , for mobile = flutter, react native)
2. **Application layer(Backend)**
   RESTful APIs/GraphQL(Node.js ,express JS)
   Authentication & Authorization (JWT)
3. **Database layer :**
   Relational database : MYSQL
4. **External services**
   Payment Gateway (Stripe,paypay)
   Email/SMS notification
   Shipping Integration (FedEx API)
5. **Scalability tools**
   a. Load balancer
   b. Caching(Redis,MemCached)
   c. CDN static content

etc

Key modules

Admin Panel
User Module
Product Catalog
Shopping Cart
Order Management
Payment gateway integration

Low Level design

Admin Panel

A.login
B. add new products in database /view/delete/update
C. Reports
| view registered customers
| View customer orders
| view stock

Etc

User Module

---

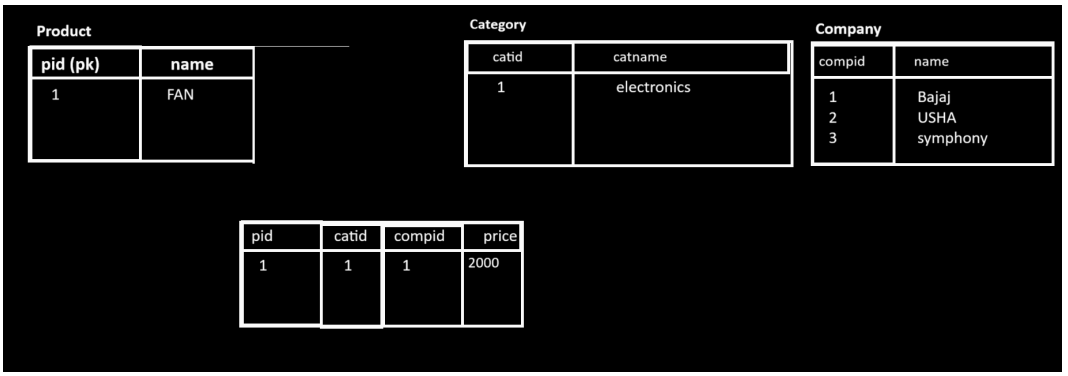
1. Search product
2. Add Product in Cart
   | register user before adding and provide login credential
   | create cart after registration
3. Place order
   | show all products from cart and display bill product wise
   | total bill + GST
   | Take confirmation from user and then place order

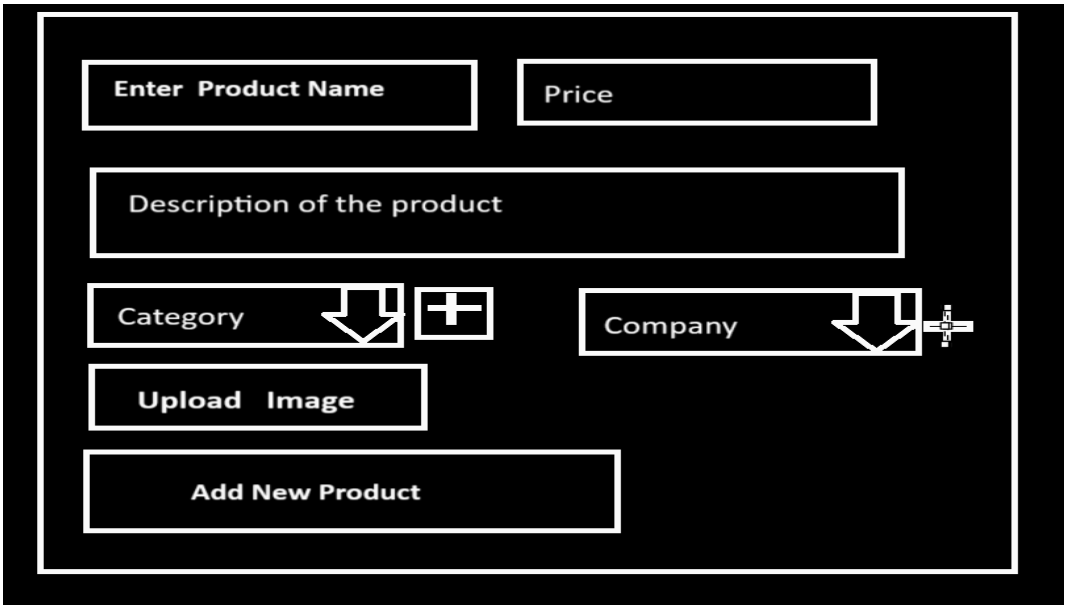
| ask online payment and finish
| send email to customer as confirmation

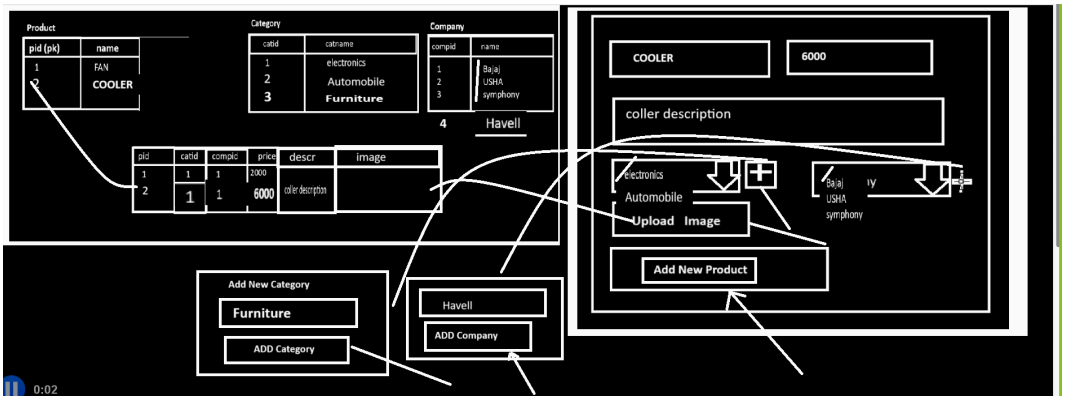
4. User can check or track the order in his own login

Database designing for table

Table structure for product







Tables

```
mysql> create table category(catid int(5) primary key auto_increment,catname varchar(200) not null unique);
Query OK, 0 rows affected, 1 warning (0.33 sec)

mysql>
```

```
mysql> create table company(compid int(5) primary key auto_increment,compname varchar(200) not null unique);
Query OK, 0 rows affected, 1 warning (0.10 sec)
```

```
mysql> create table product(prodid int(5) primary key auto_increment,prodname varchar(200) not null unique);
Query OK, 0 rows affected, 1 warning (0.17 sec)

mysql>
```

Product table

mysql> create table prodcatcompjoin(prodid int(5),foreign key(prodid) references product(prodid),catid int(5),foreign key(catid) references category(catid),compid int(5),foreign key(compid) references company(compid),price int(5),image varchar(200),proddesc varchar(1000));

How to upload the file on server by using node js

If we want to upload the file on server using express we have to use multer middleware in your application

Steps to upload the file using node or express

---

1. Create project and install express ejs and multer dependencies in project

```
D:\july 2024\onlineshopping>npm install multer

added 15 packages, and audited 148 packages in 9s

20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

   Note: we are using already created project so just we install multer dependencies we have already install express and ejs

2. Create form with file control

```
views > testup.ejs > html
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <input type="file" name="f" value="" style="width:400px;height:40px"/><br><br>
10      <input type="submit" name="s" value="'Upload File"/>
11  </body>
12  </html>
```

```
Router page

router.get("/getpage",(req,res)=>{
    res.render("testup.ejs");
})
```

3. When we upload file to the server we have to write form tag in given fashion

Note: when we upload file to server we have to use post method and it is mandatory or compulsory

Q. Why is the get method not allowed in the case of file uploading?

1. When we send data using get method then data append in browser address bar and any one can see the browser address bar data so file data may be shown in browser address bar so this is major reason we cannot use get method for file uploading
2. When we use get method then requested data append in browser address as query string in the form of name and value pair but query string can store data max 256 or 1024 kb and file size may be more than that so this is major when we upload the file then form submission method must be post.

Q.Why post method recommended in file uploading?

---

1. Post method send data via body of the page i.e header of the page means not append in browser address and send data in background so data security is achieved and when we use the post method then request send via header page means post method send two types of request in the case of file uploading first fix requested i.e file name, file size, file content type etc as well as send form data i.e form control name and control values

Note: when we upload file to server then your enctype must be multipart/form-data

multipart/form-data: multipart/form-data this enctype is used for send file name as well as file content to server like as file name, file type, file content type ,file size etc as well as send other form control data also.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form name="frm" action="/uploadfile" method="POST"
enctype="multipart/form-data">
    <input type="file" name="f" value=""
style="width:400px;height:40px"/><br><br>
    <input type="submit" name="s" value="'Upload File"/>
    </form>
</body>
</html>
```

3. **Use multer middleware in application for accept file data and store on server**

```
D:\july 2024\onlineshopping>npm install multer

up to date, audited 148 packages in 5s

20 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

D:\july 2024\onlineshopping>
```

When we use multer middleware in application we need to set storage engine for multer
When we multer storage engine we need to set destination and file name with multer storage