National University of Computer and Emerging Sciences.

NodeJS, Express, MySQL Form and view/reports

Lab 14 Tasks

Items	**Description**
Course Title	Database Systems
Lab Title	NodeJS, Express, MySQL, Form and View
Duration	3 Hours
Operating System/Tool/Language	Windows, MySQL, SQL
Objective	To get knowledge about Form and View using NodeJS, MYSQL.

NodeJS with Express and HTML.

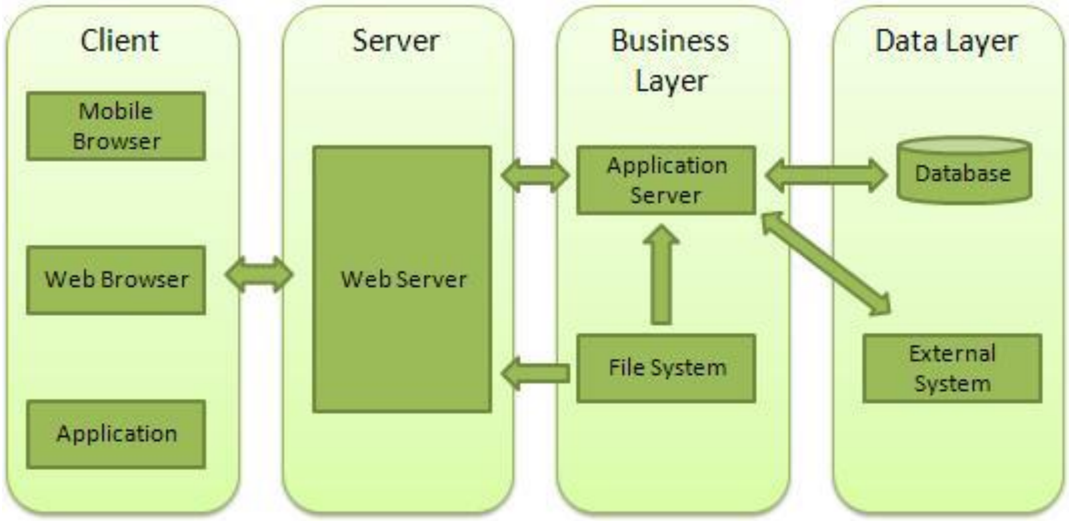
A Web Server is a software application which handles HTTP requests sent by the HTTP client, like web browsers, and returns web pages in response to the clients. Web servers usually deliver html documents along with images, style sheets, and scripts.

Most of the web servers support server-side scripts, using scripting languages or redirecting the task to an application server which retrieves data from a database and performs complex logic and then sends a result to the HTTP client through the Web server.

Web Application Architecture

A Web application is usually divided into four layers –

- **Client** − This layer consists of web browsers, mobile browsers or applications which can make HTTP requests to the web server.
- **Server** − This layer has the Web server which can intercept the requests made by the clients and pass them the response.
- **Business** − This layer contains the application server which is utilized by the web server to do the required processing. This layer interacts with the data layer via the database or some external programs.
- **Data** − This layer contains the databases or any other source of data.



Connection of nodeJS with HTML page.

A web client can be created using **http** module. Let's check the following example.

Create a js file named client.js −

File: client.js.

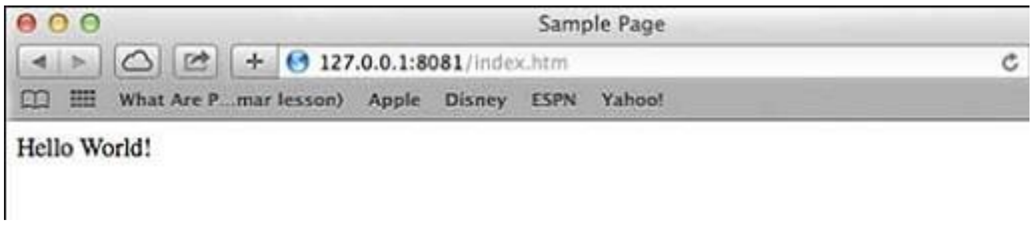
In this example, we are just connecting the nodeJs app with a simple HTML page in which we have written some stuff for displaying on the web page. We shall create two files **client.js** and **index.html** and place them in the same folder then pass patthe h of html file in the options naming **path:'index.html'** now run **node client.js**

```
var http = require('http');

// Options to be used by request
var options = {
   host: 'localhost',
   port: '8081',
   path: '/index.htm'
};

// Callback function is used to deal with response
var callback = function(response) {
   // Continuously update stream with data
   var body = '';
   response.on('data', function(data) {
      body += data;
   });

   response.on('end', function() {
      // Data received completely.
      console.log(body);
   });
}
// Make a request to the server
var req = http.request(options, callback);
req.end();
```

index.html

```
<html>
   <head>
      <title>Sample Page</title>
   </head>

   <body>
      Hello World!
   </body>
</html>
```

output



Hello World!

Express Overview(Described in the previous labs)

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node-based Web applications. Following are some of the core features of the Express framework −

- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table that is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

Express example

```
var express = require('express');
```

This line of code imports the express module which is a popular Node.js web application framework.

```
var app = express();
```

This line of code creates an instance of the Express application by calling the express() method, which returns an object that represents the web application.

```
app.get('/', function (req, res) {

res.send('Hello World');

})
```

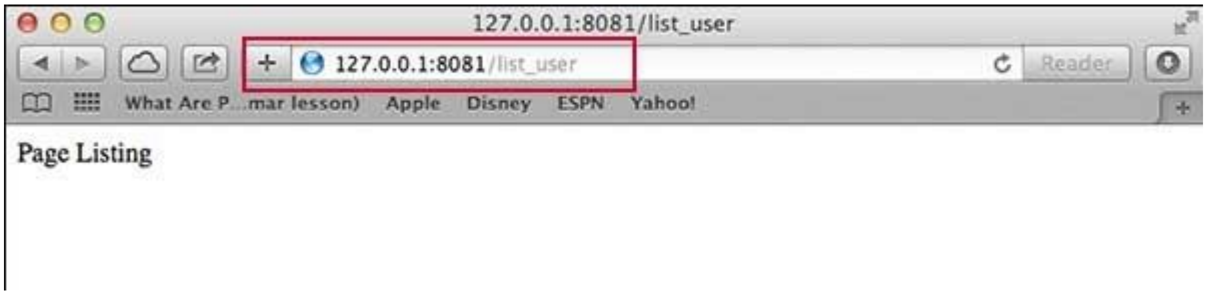
This line of code sets up a GET route for the root URL of the application ("/"). When a GET request is made to the root URL, the callback function is called with two arguments: req (the request object) and res (the response object). The res.send() method sends the string "Hello World" as the response body.

```
var server = app.listen(8081, function () {

var host = server.address().address

var port = server.address().port

console.log("Example app listening at http://%s:%s", host, port)

})
```

This line of code starts the Express application by calling the listen() method of the application object app. This method listens on the specified port number (in this case, 8081) and starts the server. The callback function is called once the server has started *and prints a message to the console with the host and port number where the application is listening.*

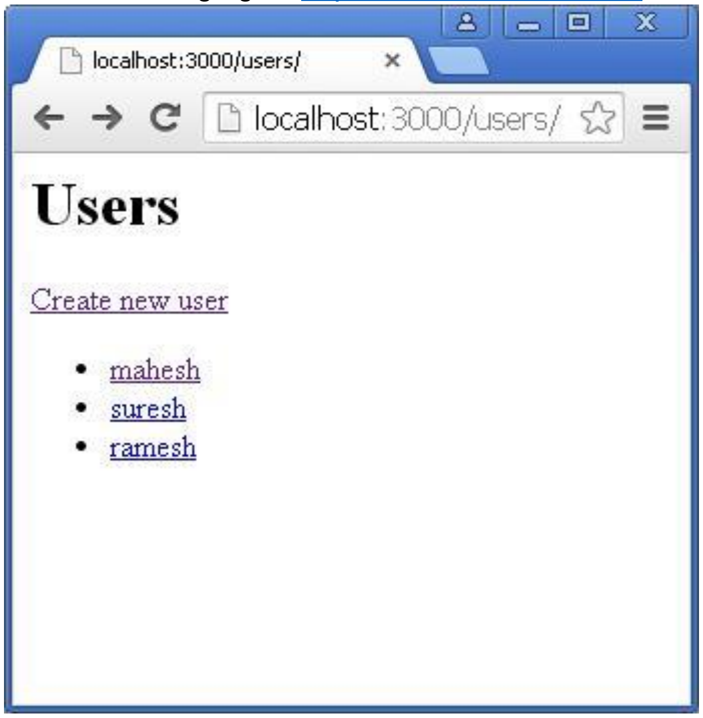
Save the above code in a file named server.js and run it with the following command.

```
$ node server.js
```



Page Listing

- Screen showing again http://127.0.0.1:8081/abcd



Users

Create new user

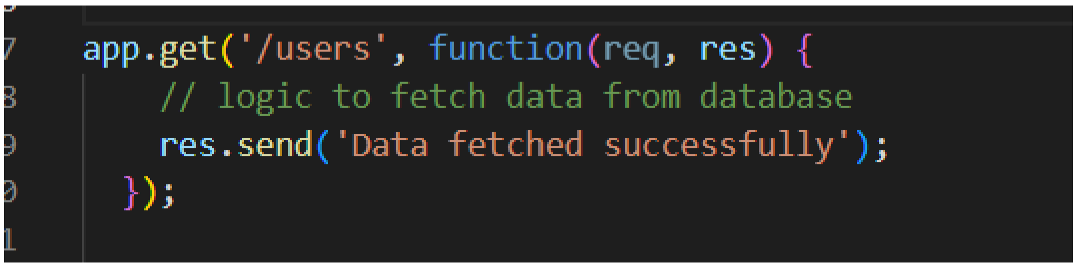
- mahesh
- suresh
- ramesh

In Node.js, HTTP GET and POST methods are used to communicate with a server. These methods allow the client to send data to the server and receive data from the server.

GET Method

The HTTP GET method is used to request data from a specified resource. In Node.js, the GET method can be handled using the app.get() method provided by the Express module. This method takes two arguments: the path of the requested resource and a callback function that handles the request and sends the response back to the client.

Here's an example of handling a GET request in Node.js:



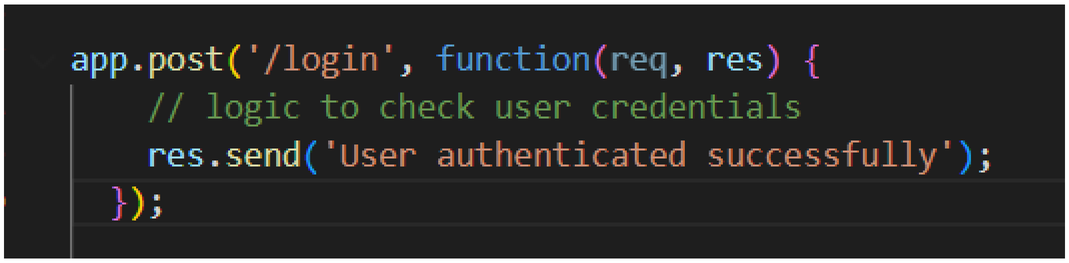
```
app.get('/users', function(req, res) {
  // logic to fetch data from database
  res.send('Data fetched successfully');
});
```

In this example, a GET request to the "/users" path triggers the callback function, which can fetch data from a database or any other source and sends a response back to the client with the message "Data fetched successfully".

POST Method

The HTTP POST method is used to submit data to be processed to a specified resource. In Node.js, the POST method can be handled using the app.post() method provided by the Express module. This method takes two arguments: the path of the resource to which data is submitted and a callback function that handles the request and sends the response back to the client.

Here's an example of handling a POST request in Node.js:



```
app.post('/login', function(req, res) {
  // logic to check user credentials
  res.send('User authenticated successfully');
});
```

In this example, a POST request to the "/login" path triggers the callback function, which can check the user credentials sent in the request and sends a response back to the client with the message "User authenticated successfully".

Lab Task

you need to generate a form using Node.js, Express, and HTML that allows users to insert customer information into the Northwind database. The form should include fields for customer name, contact name, address, city, postal code, country, and phone number, with appropriate validation for each field. Additionally, the form should include a dropdown list of all available countries in the Northwind database, and when the form is submitted, the data should be inserted into the "Customers" table of the Northwind database using a MySQL connection.