

## Agenda

- http server routes
- Express
- Mysql

## To handle the routes in http server

```
const server = http.createServer((req, res) => {  
  if (req.url === "/" && req.method === "GET") {  
    res.writeHead(200, { "Content-Type": "text/html" });  
    res.end("<h1>Home Page</h1>");  
  } else if (req.url === "/about") {  
    res.writeHead(200, { "Content-Type": "text/html" });  
    res.end("<h1>About Page</h1>");  
  } else {  
    res.writeHead(404);  
    res.end("Page not found");  
  }  
});
```

## Routing

- Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).
- It maps a particular URL or URL pattern to a specific function that handles the request.
- Routing is like a map that helps the server know how to respond to different types of requests made by a user or client.
- For example, when a client sends a request like:
  - GET /home
  - POST /user
- The routing system determines which function or "route handler" should be called to respond to that request.

## nodemon

- It is a developer tool that automatically restarts your Node.js app when file changes in the directory are detected.
- saves you from having to manually stop and restart your server every time you edit code.
- In short it monitors your files and restarts server automatically
- install it globally using below command

```
npm install -g nodemon
```

- to Use It

```
nodemon server.js
```

## Express

- Developed by many developers around the world
- It is a fast,minimilistic web framework for nodejs
- It is a lightweight and flexible routing framework with minimal core features meant to be augmented through the use of Express middleware modules.

```
# to start using express application install express using npm or add using yarn
npm install express
# OR
yarn add express
```

## Advantage of express over http server

### http server

- It's the core built-in module used to create HTTP servers.
- Gives low-level control over requests and responses.
- we manually handle Routing (URLs),Headers, Response codes, Parsing JSON, etc.

### express server

- A framework built on top of http module.
- Simplifies routing, middleware, JSON handling, templating, etc.
- Makes your code shorter, cleaner, and easier to maintain.

## Routing in express

- In a framework like Express, routing is typically implemented by setting up specific endpoints with combination of paths and methods (like GET, POST, etc.) that will trigger certain functions to handle those requests.
- These specific endpoints are created called as **Routes**
- Each route can have one or more handler functions, which are executed when the route is matched.
- A route is essentially a combination of a URL and a method, and it is where the logic of handling requests resides.
- In express Route definition takes the following structure:

```
app.METHOD(PATH, HANDLER)
```

- Where:

1. app is an instance of express.

2. METHOD is an HTTP request method, in lowercase.
3. PATH is a path on the server.
4. HANDLER is the function executed when the route is matched.

- we define routing using methods of the Express app object that correspond to HTTP methods
- for example, `app.get()` to handle GET requests and `app.post` to handle POST requests.
- These routing methods specify a callback function (sometimes called "handler functions") called when the application receives a request to the specified route (endpoint) and HTTP method.
- In other words, the application "listens" for requests that match the specified route(s) and method(s), and when it detects a match, it calls the specified callback function.

```
const express = require("express");
const app = express();

// Route for homepage
app.get("/", (req, res) => {
  res.send("Welcome to Home Page");
});

// Route for /about
app.get("/about", (req, res) => {
  res.send("About Us Page");
});

// Route for POST request to /login
app.post("/login", (req, res) => {
  res.send("Login data received");
});

app.listen(3000, 'localhost', () => console.log("Server running on port 3000"));
```

## Connecting with mysql database

1. add/install the mysql module

```
npm install mysql2
```

2. Create the pool object

```
const pool = mysql.createPool({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'kdac_db'
});
```

### 3. Call the query method

```
const sql = `SELECT * FROM product`  
pool.query(sql, (error, data) => {  
  if(data)  
    response.send(data)  
  else  
    response.send(error)  
})
```

SUNBEAM INFOTECH