NodeJS Training - Day 4

By : LAKSHMIKANT DESHPANDE (M.Tech)

Express Framework

- Day 3 Recap
- Web Server
- What is Express JS
- HTTP Basics, Verbs, Status Codes
- Handling HTTP Routes
- Navigation Route options
- Middleware

Session 1

- Day 3 Recap
    - OS Module
    - Working with File System
    - REST (REpresentational State Transfer)
    - Creating Web Server

Session 1 contd….

- Web Server
- REST APIs with inbuilt http module

Session 2

- Express Framework
- API Implementation
  - GET
  - PUT
  - POST
  - PATCH
  - DELETE

Session 3

- Structuring
    - Controllers
    - Services
    - Database
    - Routers
    - Config

Session 4

- Middleware
- Types of Middleware
- Custom Middleware

Express Framework

- Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for building web and mobile applications.
- It is often referred to as the de facto standard server framework for Node.js. Express.js simplifies the process of creating server-side applications by providing a range of tools and middleware to handle common web development tasks.

Key Features of Express.js

- **Routing:**
    - Express.js offers a simple and flexible routing system for handling different HTTP requests at various URL paths.
- **Middleware:**
    - Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application’s request-response cycle. Middleware can execute any code, make changes to the request and response objects, end the request-response cycle, and call the next middleware function.

Key Features of Express.js contd…

- **HTTP Helpers:**
  - Express.js provides numerous HTTP utility methods and middleware for creating robust APIs and handling HTTP requests and responses
- **View System:**
  - It supports various template engines, allowing you to dynamically generate HTML pages..

Key Features of Express.js contd…

- **Static File Serving:**
  - Express.js can serve static files, such as images, CSS files, and JavaScript files.
- **Error Handling:**
  - It includes a straightforward way to handle errors, which is essential for building reliable applications.

HTTP Basics, Verbs, Status Codes

- **HTTP (Hypertext Transfer Protocol)** is the foundational protocol for transferring data over the web. It operates as a request-response protocol where a client (usually a web browser) sends a request to a server, and the server responds with the requested resource or information.
- HTTP is an extensible protocol that relies on concepts like resources and Uniform Resource Identifiers (URIs), simple message structure, and client-server communication flow.

HTTP Verbs (Methods)

- **GET**
  - **Description**: Retrieves data from the server.
  - **Usage**: Used to request data from a specified resource.
  - **Example**: GET /users retrieves a list of users.
- **POST**
  - **Description**: Submits data to be processed to a specified resource.
  - **Usage**: Used to send data to the server to create or update a resource.
  - **Example**: POST /users with user data to create a new user.

HTTP Verbs (Methods)

- **PUT**
  - **Description**: Updates a resource or creates a new resource if it does not exist.
  - **Usage**: Used to update a resource with new data.
  - **Example**: PUT /users/1 updates the user with ID 1.
- **PATCH**
  - **Description**: Partially updates a resource.
  - **Usage**: Used to make partial updates to a resource.
  - **Example**: PATCH /users/1 with data to update certain fields of the user with ID 1.

HTTP Verbs (Methods) contd…

- **DELETE**
  - **Description**: Deletes a resource from the server.
  - **Usage**: Used to remove a specified resource.
  - **Example**: DELETE /users/1 removes the user with ID 1.
- **OPTIONS**
  - **Description**: Describes the communication options for the target resource.
  - **Usage**: Used to retrieve the allowed methods for a resource.
  - **Example**: OPTIONS /users checks which HTTP methods are allowed for /users.

HTTP Status Codes

HTTP status codes are issued by the server to indicate the outcome of the request. They are divided into several categories:

1xx (Informational)

- **100 Continue**: The server has received the request headers and the client should proceed to send the request body.
- **101 Switching Protocols**: The server is switching protocols as requested by the client.

2xx (Successful)

- **200 OK**: The request was successful, and the server responded with the requested data.
- **201 Created**: The request was successful and a new resource was created.
- **204 No Content**: The request was successful but there is no content to send in the response.

HTTP Status Codes contd…

3xx (Redirection)

- **301 Moved Permanently**: The requested resource has been permanently moved to a new URL.
- **302 Found**: The requested resource is temporarily located at a different URL.
- **304 Not Modified**: The resource has not been modified since the last request.

4xx (Client Error)

- **400 Bad Request**: The request could not be understood by the server due to malformed syntax.
- **401 Unauthorized**: The request requires user authentication.
- **403 Forbidden**: The server understood the request, but refuses to authorize it.
- **404 Not Found**: The requested resource could not be found on the server.

HTTP Status Codes contd…

5xx (Server Error)

- **500 Internal Server Error**: The server encountered an unexpected condition that prevented it from fulfilling the request.
- **502 Bad Gateway**: The server received an invalid response from the upstream server.
- **503 Service Unavailable**: The server is currently unable to handle the request due to temporary overloading or maintenance.

HTTP Routes

- **Route**: A route defines the path and method (GET, POST, etc.) that the server should respond to.
- **Route Handler**: A function that is executed when a request matches a route.

Middleware

What is Middleware?

Middleware functions are executed sequentially in the order they are added to the middleware stack. Each middleware function can perform operations such as:

- **Processing**: Read or modify req (request) and res (response) objects.
- **Ending**: Send a response to the client and end the request-response cycle.
- **Passing Control**: Call next() to pass control to the next middleware function.

Types of Middleware

- **Application-Level Middleware**: Applied globally to the entire app.
- **Router-Level Middleware**: Applied to specific routes.
- **Built-In Middleware**: Provided by Express for common tasks.
- **Third-Party Middleware**: External libraries for additional features.
- **Error-Handling Middleware**: Catches and handles errors.