SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC-UGC with 'A' Grade

Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

Course Code and Name : 19TS601 FULL STACK DEVELOPMENT

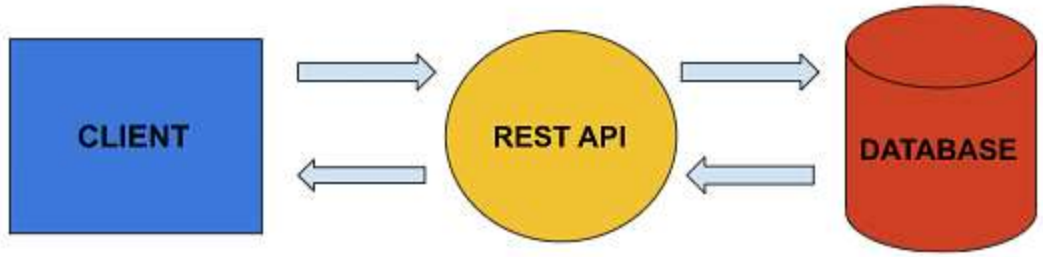
Unit 3 : NODEJS AND EXPRESS

Topic : EXPRESS REST API

Express REST API

- Restful API is very popular and commonly used to create APIs for web-based applications.
- Express is a back-end web application framework of node js, and with the help of express, we can create an API very easily.
- REST API is the standard way to send and receive data for web services.
- A client sends a req which first goes to the rest API and then to the database to get or put the data after that, it will again go to the rest API and then to the client.

- Using an API is just like using a website in a browser, but instead of clicking on buttons, we write code to req data from the server. It's incredibly adaptable and can handle multiple types of requests



What Is REST API?

- REST (Representational state transfer) is a popular architecture that is used to create web services.
- API (Application Programming Interface is a code that allows two software programs to communicate with each other.
- REST API is a software that allows two apps to communicate with one another over the internet and through numerous devices.

HTTP Request Types

- HTTP Requests are simply messages that are sent by the client to do some tasks on the server
- GET - Get command is used to request data from the server, but mainly this method is used to read data
- PATCH - This command is used to update, change or replace the data
- POST - The post method is used to create new or to edit already existing data
- Delete - This delete command is used to delete the data completely from the server

API Using Express and Its Architecture

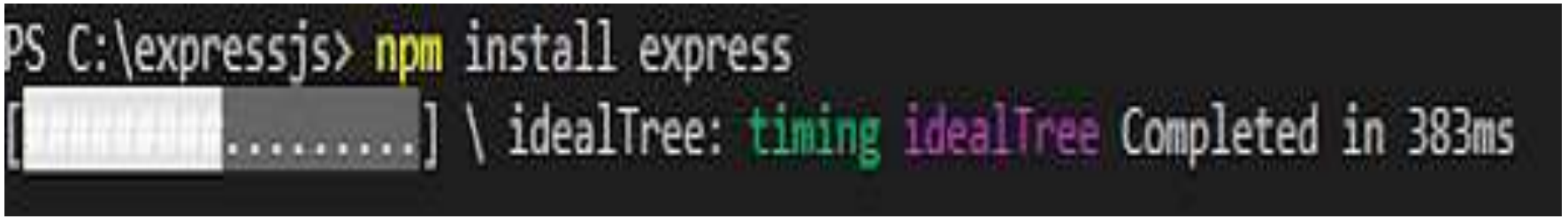
- A request is sent by the client in the form of a JSON file, and with the help of an HTTP request which gets a patch post and delete, it will go to the server first then, the server sends back the response to the client in the form of a message to tell what happened to your request.
- Step 1: First, open your editor. Now open your terminal and write a command npm init -y

```
PS C:\expressjs> npm init -y
Wrote to C:\expressjs\package.json:

{
  "name": "expressjs",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
```

- This command will create a JSON file.

- npm install express



- This will install dependencies in the package.json file
- Step 3: Now we will create a new file to write all our code index.js
- Write a few lines of code in the first line. We will import the express packages

- Step 4

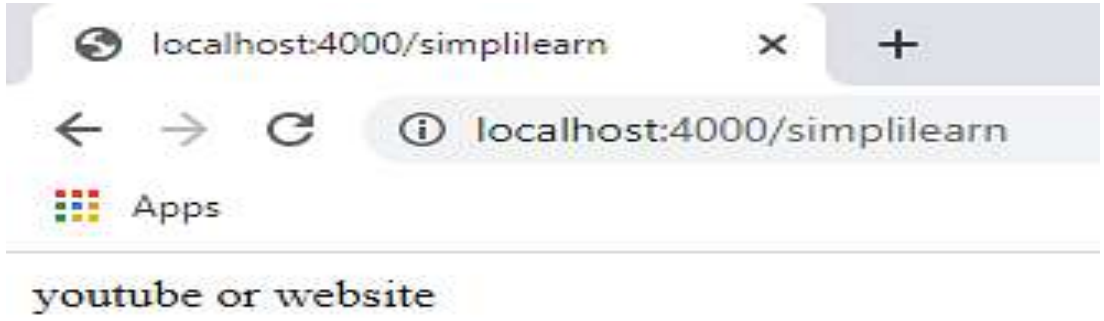
index.js

```
index.js > app.get('/simplilearn') callback
1   const app = require('express')();
2
3   const PORT = 4000;
4
5   app.listen(
6       PORT,
7       () => console.log(`its running on http://localhost:${PORT}`)
8   );
9
10  app.get('/simplilearn', (req,res) => {
11      res.send("youtube or website")
12  });
13
```

- This app.listen will respond to the server to listen on a specific port which we have defined already as a variable 4000
- Step 5: Then we will write a second argument as a callback to tell the API is working, now lets run this code to check our API is working or node
- To run this code, we will write a command node.

```
PS C:\expressjs> node .
its running on http://localhost:4000
```

- Step 6: As our code is running fine, let's open our search engine and write localhost:4000, this will not run,
- Step 7: Now we need an add the second argument here
- localhost:4000/simplilearn



- API is running Successfully

REST

- REST (REpresentational State Transfer) is an architectural style for developing web services and systems that can easily communicate with each other.
- REST is popular due to its simplicity and the fact that it builds upon existing systems and features of the internet's HTTP to achieve its objectives, as opposed to creating new standards, frameworks and technologies.
- It is popularly believed that REST is a protocol or standard. However, it is neither.
- REST is an architectural style that is commonly adopted for building web-based application programming interfaces (APIs).

REST

- In this architectural style, systems interact through operations on resources.
- Resources include all data and functionality.
- They are accessed using Uniform Resource Identifiers (URIs) and acted upon using simple operations.
- Systems, service interfaces or APIs that comply with the REST architectural style are called RESTful systems (or RESTful APIs).
- Typically, these applications are lightweight, fast, reliable, scalable and portable.

REST constraints

- For a system to be RESTful, it must satisfy five mandatory constraints:
- It must have a **uniform interface** to simplify system architecture and improve the visibility of interactions between system components.
- It must incorporate the **client-server** design pattern, allowing for the separation of concerns and for the client and server implementations to be done independently.
- It must be **stateless**, meaning that the server and client don't need to know anything about each other's state so they can both understand the messages received from each other without having to see previous messages.

- It must be **cacheable**, meaning a response should label itself as cacheable or noncacheable, and copies of frequently accessed data must be stored in multiple caches along the request-response path.
- It must be **layered** to constrain component behavior, to remove the need to edit code (on the client or server) and to improve the web app's security.

REST and the HTTP methods

- In a REST system, numerous resource methods are used for resource interactions and to enable resource state transitions. These methods are also known as HTTP verbs.
- The default operation of HTTP is GET, used when retrieving a resource or set of resources from the server by specifying the resource ID.
- In addition to GET, HTTP also defines several other request methods, including PUT (update a resource by ID), POST (create a new resource) and DELETE (remove a resource by ID).
- The REST philosophy asserts that to delete something on the server, you would simply use the URL for the resource and specify the DELETE method of HTTP

- For saving data to the server, a URL and the PUT method would be used.
- For operations that are more involved than simply saving, reading or deleting information, the POST method of HTTP can be used.
- **Web APIs.** RESTful services employ effective HTTP mechanisms for caching to reduce latency and the load on servers.

- **Separation of client and server.** By providing many endpoints, a REST API makes it easy to create complex queries that can meet specific deployment needs.
- Also, different clients hit the same REST endpoints and receive the same responses if they use a REST interface, improving reliability and performance.
- **Resilience.** In a REST system, the failure of a single connector or component does not result in the entire system collapsing.

Disadvantages of REST

- **Design limitations.** There are some limitations of the REST architecture design. These include multiplexing several requests over a single TCP connection, having different resource requests for each resource file, server request uploads and long HTTP request headers, which cause delays in webpage loading. Also, the freedom that REST provides regarding design decisions can make REST APIs harder to maintain.
- **Stateless applications.** Because the server does not store state-based information between request-response cycles, the client must perform state management tasks, which makes it difficult to implement server updates without using client-side polling or other types of webhooks that send data and executable commands between apps.

- **Definition.** REST lacks a clear reference implementation or a definitive standard to determine whether a design can be defined as RESTful or whether a web API conforms to REST-based principles.
- **Data overfetching/underfetching.** RESTful services frequently return large amounts of unusable data along with relevant information -- typically the result of multiple server queries -- increasing the time it takes for a client to return all the required data.

- REST makes it easy to implement network-based web services by using the basic construct of the network protocol itself, which in terms of the internet is HTTP.
- There's no need for developers to work with custom protocols for client-server message exchanges. That's important, as REST and its principles are intended to apply not just to the internet, but to apply to all protocols, including WebDAV and FTP.
- That said, REST is mainly a good fit for CRUD (create, read, update, delete) applications where the five HTTP methods (GET, PUT, POST, PATCH, DELETE) can be easily used. For other types of applications or use cases, REST might not be the best choice.

Concept of RESTful APIs in Express

- RESTful APIs are a popular way of creating web applications that exchange data over the internet in a standardized manner.
- These APIs follow specific principles such as using resource-based URLs and HTTP methods to perform various operations like creating, reading, updating, and deleting data.
- ExpressJS is a powerful framework that allows developers to easily create routes, handle requests, and send responses, making it a versatile choice for building APIs that are robust and scalable.

- REST Architecture: REST, which stands for Representational State Transfer, is an architectural style for designing networked applications.
- Resource-Based: RESTful APIs in ExpressJS are designed around resources, which are identified by unique URLs.
- These resources represent the entities (e.g., users, products, articles) that the API manipulates.

- HTTP Methods: RESTful APIs use standard HTTP methods to perform CRUD (Create, Read, Update, Delete) operations on resources:
- GET: Retrieves data from a resource.
- POST: Creates a new resource.
- PUT: Updates an existing resource.
- DELETE: Deletes a resource.

- **Statelessness**: RESTful APIs are stateless, meaning that each request from a client contains all the information needed to process the request. Servers do not maintain a session state between requests.
- **Uniform Interface**: RESTful APIs have a uniform interface, which simplifies communication between clients and servers. This interface typically involves the use of standard HTTP methods, resource identifiers (URLs), and representations (e.g., JSON, XML).

- **ExpressJS and Routing**: In ExpressJS, you define routes to handle incoming requests for specific resources and HTTP methods. Each route specifies a callback function to process the request and send an appropriate response.
- **Middleware Integration**: ExpressJS middleware can be used to handle tasks such as request validation, authentication, and response formatting, enhancing the functionality and security of RESTful APIs.
- **Response Codes**: RESTful APIs use standard HTTP status codes to indicate the success or failure of a request. Common status codes include **200 (OK), 201 (Created), 400 (Bad Request), 404 (Not Found)**, and **500 (Internal Server Error)**.

- Install the necessary package in your application using the following command.
- npm install express
- *// server.js*
- **const** express = require('express');
- **const** app = express();
- **const** PORT = 3000;

```
// To define the sample data
let books = [
    {
        id: 1,
        title: 'The Great Gatsby',
        author: 'F. Scott Fitzgerald'
    },
    {
        id: 2,
        title: 'To Kill a Mockingbird',
        author: 'Harper Lee'
    },
];
```

```
// Define routes for handling GET requests
app.get('/api/books',
    (req, res) =&gt; {
        res.json(books);
    });
```

```
app.get('/api/books/:id',
    (req, res) =&gt; {
        const id =
            parseInt(req.params.id);
        const book =
            books.find(book =&gt; book.id === id);
        if (book) {
            res.json(book);
        } else {
            res.status(404)
                .json({ message: 'Book not found' });
        }
    });
```

```
app.listen(PORT,
    () =&gt; {
        console.log(`Server is running on port ${PORT}`);
    });
```

Start your application using the following command.

```
node server.js
```

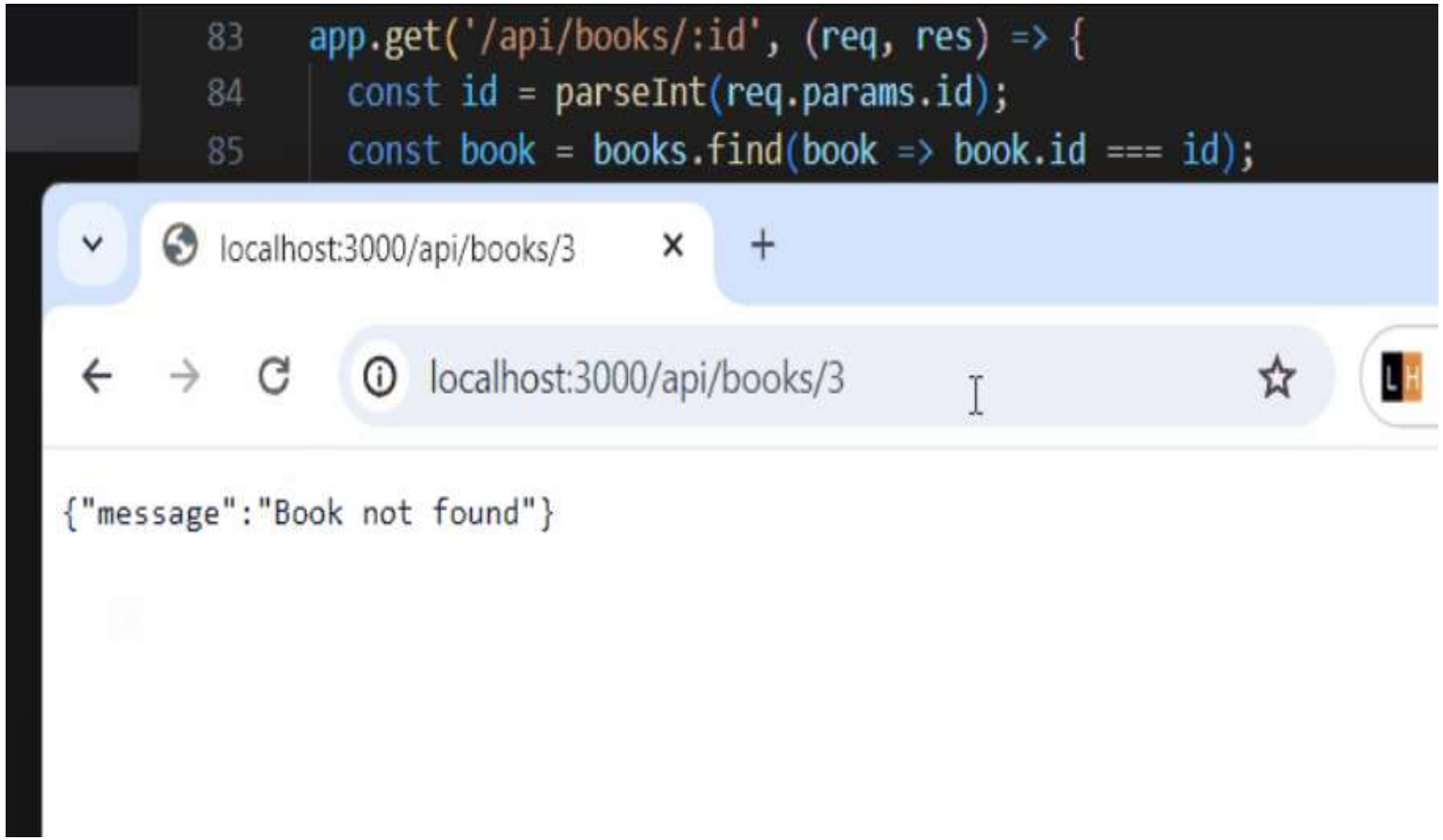
OUTPUT

```
83  app.get('/api/books/:id', (req, res) => {
84    const id = parseInt(req.params.id);
85    const book = books.find(book => book.id === id);
```

localhost:3000/api/books/3

localhost:3000/api/books/3

```
{"message":"Book not found"}
```


83 app.get('/api/books/:id', (req, res) => {
84 const id = parseInt(req.params.id);
85 const book = books.find(book => book.id === id);
localhost:3000/api/books/3
localhost:3000/api/books/3
{"message":"Book not found"}

- REST APIs enable you to develop all kinds of web applications having all possible CRUD (create, retrieve, update, delete) operations.

HTTP method for the action performed by API.

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE
- HTTP PATCH


Thank
You!