JavaScript Advanced - Express JS

Created Date : 04/03/2022
Created By : GOVE ACADEMY

Version History

S. No	Revision Date	Version	Modifications	Modified By

Table of Contents

1 . Express JS

- Express.js is a web framework for Node.js that simplifies the process of building web applications and APIs
- It provides a set of tools and functions to handle HTTP requests
- It define routes for handling different types of HTTP requests (like GET, POST, PUT, DELETE)

Some of the key features of Express.js include:

- **Routing**: Express.js provides a simple and intuitive API for defining routes in your application.
- **Middleware**: Middleware functions can be used to perform tasks like authentication, logging, and error handling.
- **Error Handling**: Express.js provides robust error handling mechanisms, allowing you to catch and handle errors that occur in your application.
- **Database Integration**: Express.js makes it easy to integrate your application with databases like MongoDB, MySQL, and PostgreSQL, among others.
- **Security**: Express.js provides a range of security features to help protect your application from common vulnerabilities, including cross-site scripting (XSS), cross-site request forgery (CSRF), and more.

Get API using express without Authentication

Syntax:

- Import the express module

```
//import modules here
const express = require('express')
```

- create an instance of the Express application

```
const app = express()
```

- Routes

```
app.get('/',(req,res)=>{
    res.send("Hello World-1")
})
```

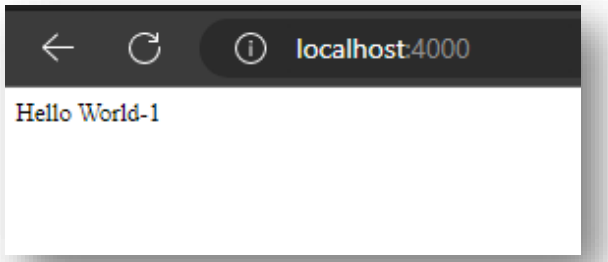
- Start the Server

```
app.listen(4000,()=>{
    console.log("Listening port : ")
})
```

Terminal :

```
023\DEVI-KOUSALYA\JAVASCRIPT-ADVANCED\js-advanced> node main.js
Port is running in 4000
```

Browser



Get API using express without Authentication

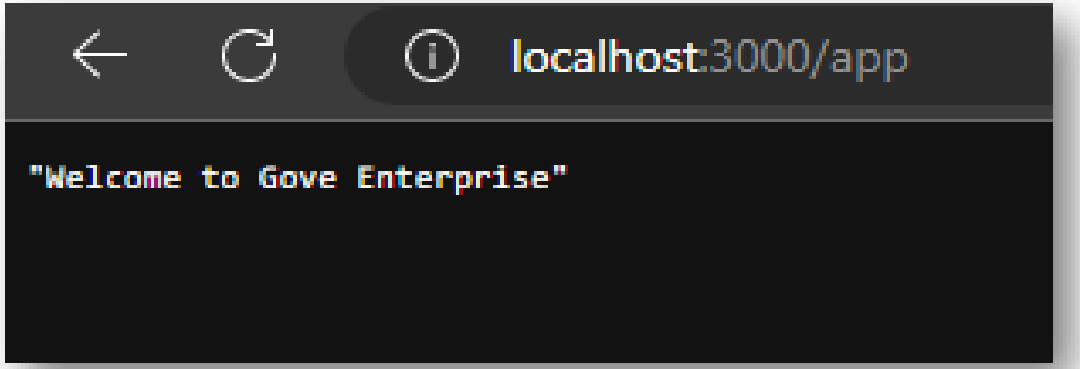
Express API-Get Method

app.js > ...

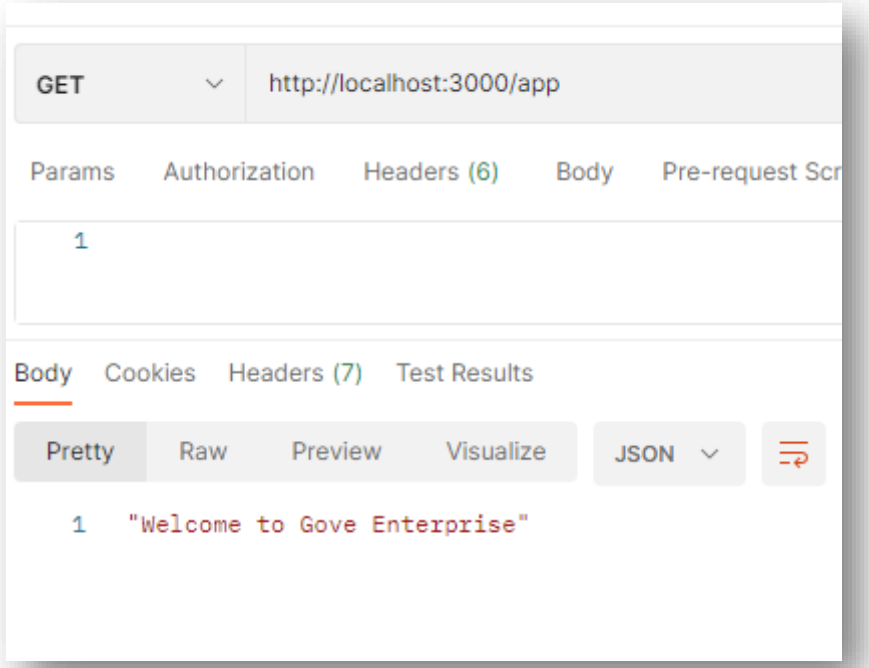
```
/**
 * CreatedBy : Devi Kousalya
 * CreatedDate : 04 April 2023
 * Description : This file contains the  Get API
 *               using express without Authentication
 */
const express= require('express')
const app = express()
app.get('/app',(req,res)=>{
    res.json("Welcome to Gove Enterprise")
})

app.listen(3000,()=>{
    console.log("Listening port is: ")
})
```

Testing Get API using express without Authentication

- In Browser



- In Postman



Securing the API with basic Authentication

- Securing an API with basic authentication is a common way to restrict access to the API and ensure that only authorized users can access it.
- By using basic authentication, the server can verify the user's identity and allow or deny access to the API accordingly.
- It is very useful to restrict access to sensitive information or functionality, such as user data or administrative tasks.

- Import the express module

```
JS getAPI-A.js > ...
  //import required Module
  const express = require('express');
  const basicAuth = require('basic-auth');
```

- Create an instance of the Express application

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

- Middleware Function for Basic Authentication

```
const auth = (req, res, next) => {
    const user = basicAuth(req);

    if (!user || user.name !== 'myuser' || user.pass !== 'mypassword') {
        res.status(401).send('Authentication required.');
        return;
    }

    next();
};
```

- Route for Getting Data with Basic Authentication

```
app.get('/api/data', auth, (req, res) => {
    res.json(data);
});
```

- Start the Server

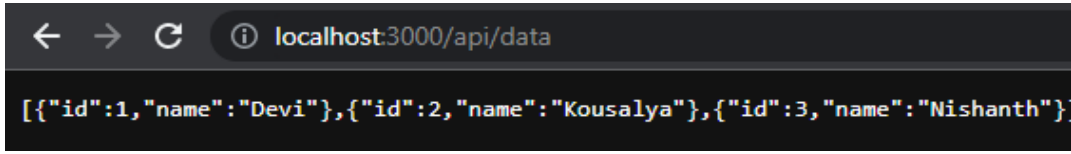
```
// Start the server
app.listen(3000, () => console.log('Lisenting Potrt : 3000'));
```

Get API using express with Authentication

- Express API-Get Method

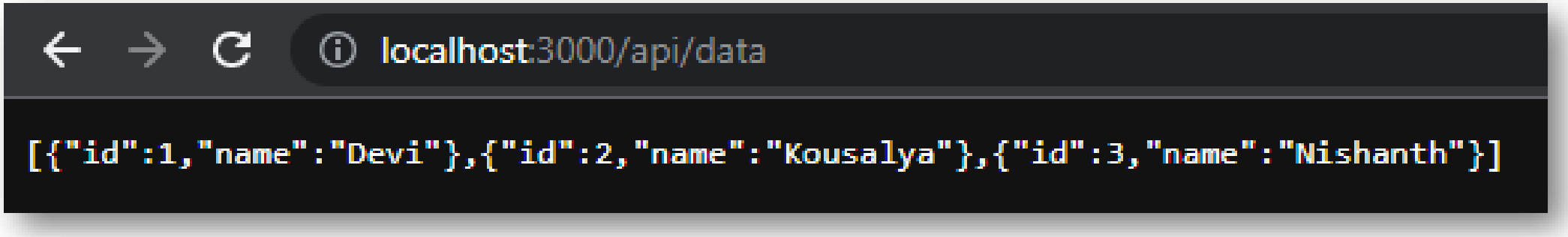
```
/**
 * CreatedBy : Devi Kousalya
 * Created Date : April 24 2023
 * Description : This file contains the  GET APU using Express Without Authentication
 */
//import required Module
const express = require('express');
const basicAuth = require('basic-auth');
const app = express();
const data = [
    { id: 1, name: 'Devi' },
    { id: 2, name: 'Kousalya' },
    { id: 3, name: 'Nishanth' }
];
//middleware function for Basic Authentication
const auth = (req, res, next) => {
    const user = basicAuth(req);
    if (!user || user.name !== 'myuser' || user.pass !== 'mypassword') {
        res.status(401).send('Authentication required.');
        return;
    }
    next();
};
//  route for getting data with Basic Authentication
app.get('/api/data', auth, (req, res) => {
    res.json(data);
});
// Start the server
app.listen(3000, () => console.log('Lisenting Potrt :  3000'));
```

```
Lisenting Potrt :  3000
```



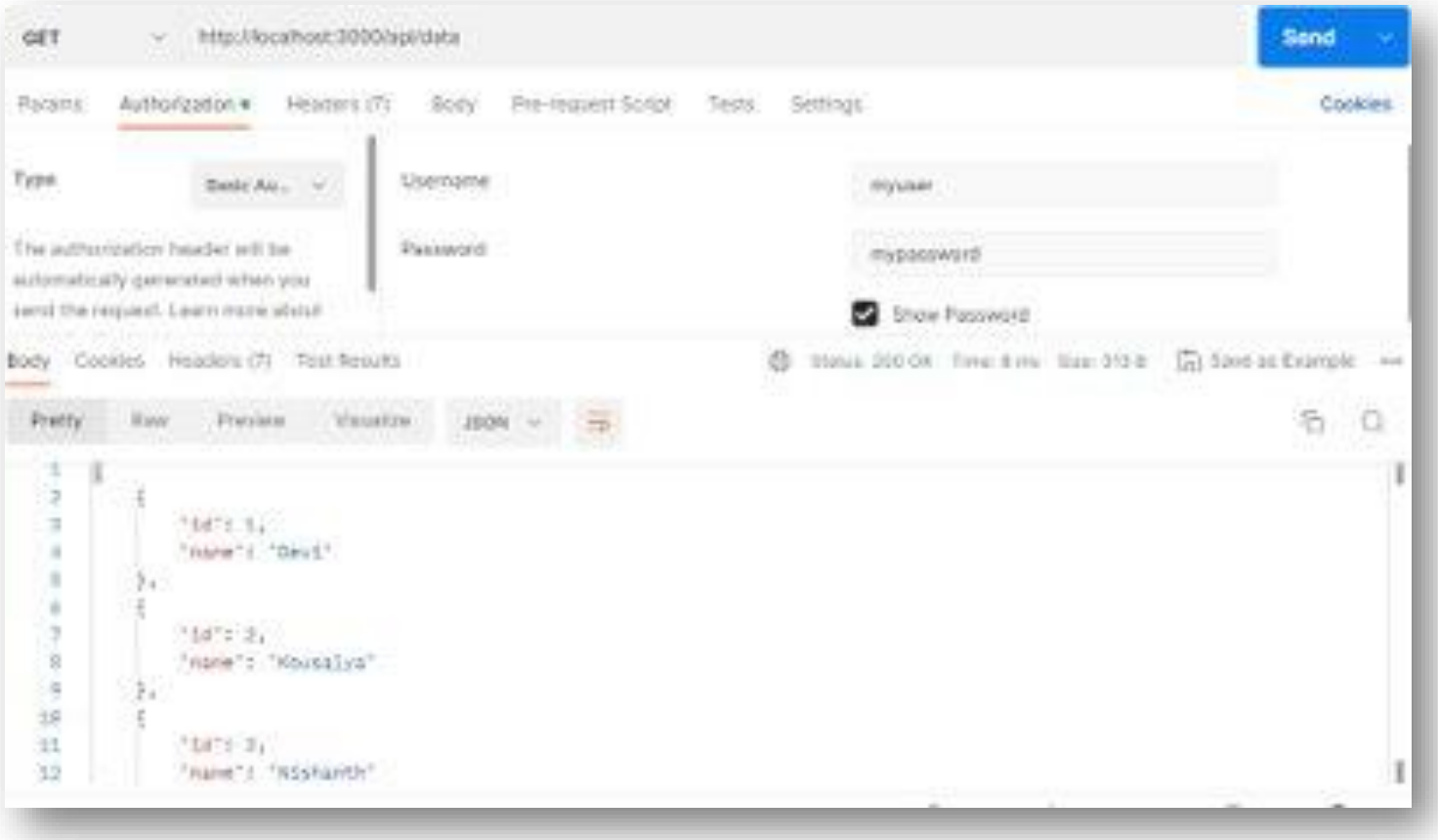
Testing Get API using express with Authentication

- In Browser



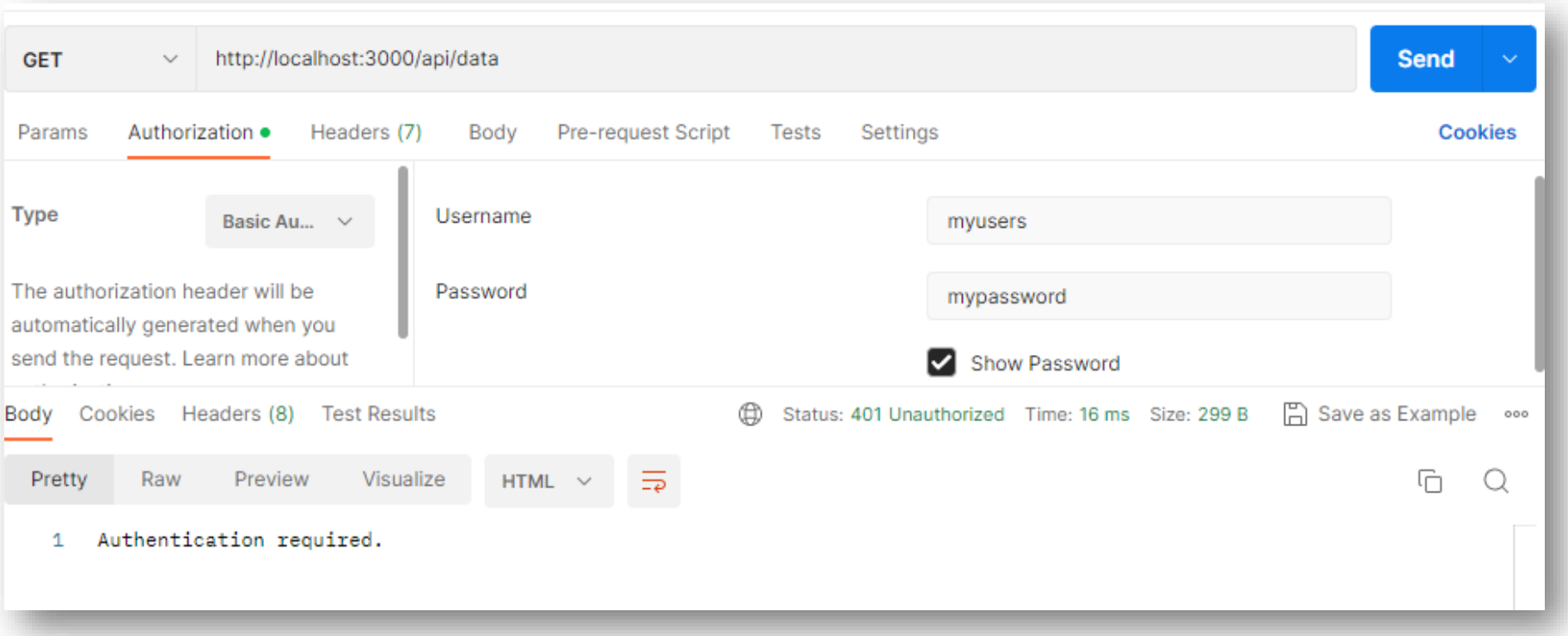
Testing Get API In Postman using Incorrect Password

- Correct Password



Testing Get API In Postman using Incorrect Password

- In-Correct Password

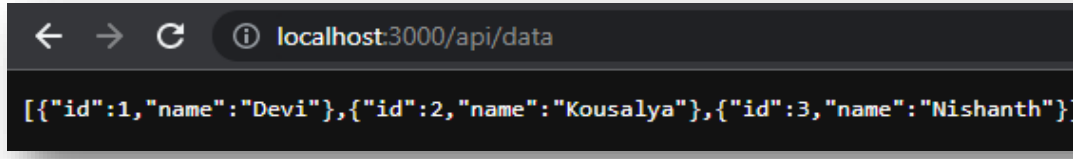


Get API using express with Authentication

- Express API-Get Method

```
/**
 * CreatedBy : Devi Kousalya
 * Created Date : April 24 2023
 * Description : This file contains the  GET APU using Express Without Authentication
 */
//import required Module
const express = require('express');
const basicAuth = require('basic-auth');
const app = express();
const data = [
    { id: 1, name: 'Devi' },
    { id: 2, name: 'Kousalya' },
    { id: 3, name: 'Nishanth' }
];
//middleware function for Basic Authentication
const auth = (req, res, next) => {
    const user = basicAuth(req);
    if (!user || user.name !== 'myuser' || user.pass !== 'mypassword') {
        res.status(401).send('Authentication required.');
        return;
    }
    next();
};
//  route for getting data with Basic Authentication
app.get('/api/data', auth, (req, res) => {
    res.json(data);
});
// Start the server
app.listen(3000, () => console.log('Lisenting Potrt :  3000'));
```

```
Lisenting Potrt :  3000
```



Post API using express without Authentication

Post Api without Authentication

Authentication: API authentication is a combination of technology and process that proves or verifies the identities of users who want access to an API.

```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

// Use body-parser middleware to parse incoming JSON data
app.use(bodyParser.json());

const data = [
    { id: 1, name: 'Devi' },
    { id: 2, name: 'Kousalya' },
    { id: 3, name: 'Nishanth' }
];

// Route for getting data
app.get('/api/userdata', (req, res) => {
    res.json(data);
});

// Route for creating data
app.post('/api/data', (req, res) => {
    // Get the data from the request body
    const newData = req.body;

    // Generate a new ID for the data
    const newId = data.length + 1;

    // Add the new data to the data array
    data.push({ id: newId, ...newData });

    // Send a response with the newly created data
    res.json(data[data.length - 1]);
});

// Start the server
app.listen(3010, () => console.log('Listening Port :  3010'));
```

Post API using express with Authentication

Post Api with Authentication

```
const express = require('express');
const basicAuth = require('basic-auth');
const bodyParser = require('body-parser');

const app = express();

// Use body-parser middleware to parse incoming JSON data
app.use(bodyParser.json());

const data = [
    { id: 1, name: 'Devi' },
    { id: 2, name: 'Kousalya' },
    { id: 3, name: 'Nishanth' }
];

//middleware function for Basic Authentication
const auth = (req, res, next) => {
    const user = basicAuth(req);
    if (!user || user.name !== 'myuser' || user.pass !== 'mypassword') {
        res.status(401).send('Authentication required.');
        return;
    }
    next();
};

// Route for getting data with Basic Authentication
app.get('/api/userdata', auth, (req, res) => {
    res.json(data);
});
```

```
20  // Route for creating data
21  app.post('/api/data', (req, res) => {
22      // Get the data from the request body
23      const newData = req.body;
24
25      // Generate a new ID for the data
26      const newId = data.length + 1;
27
28      // Add the new data to the data array
29      data.push({ id: newId, ...newData });
30
31      // Send a response with the newly created data
32      res.json(data[data.length - 1]);
33  });
34
35  // Start the server
36  app.listen(3010, () => console.log('Listening Port :  3010'));
```

How to get the data that was passed in the request body

This is how we get data that is passed in body and print it in the console

```
const express = require('express');
const basicAuth = require('basic-auth');
const bodyParser = require('body-parser');

const app = express();

// Use body-parser middleware to parse incoming JSON data
app.use(bodyParser.json());

const data = [
    { id: 1, name: 'Devi' },
    { id: 2, name: 'Kousalya' },
    { id: 3, name: 'Nishanth' }
];

//middleware function for Basic Authentication
const auth = (req, res, next) => {
    const user = basicAuth(req);
    if (!user || user.name !== 'myuser' || user.pass !== 'mypassword') {
        res.status(401).send('Authentication required.');
        return;
    }
    next();
};

// Route for getting data with Basic Authentication
app.get('/api/userdata', auth, (req, res) => {
    res.json(data);
});
```

```
// Route for getting data with Basic Authentication
app.get('/api/userdata', auth, (req, res) => {
    res.json(data);
});

// Route for creating data with Basic Authentication
app.post('/api/data/auth', auth, (req, res) => {
    // Get the data from the request body
    const newData = req.body;
    console.log("request",req.body.name)

    // Generate a new ID for the data
    const newId = data.length + 1;

    // Add the new data to the data array
    data.push({ id: newId, ...newData });

    // Send a response with the newly created data
    res.json(data[data.length - 1]);
});

// Start the server
app.listen(3005, () => console.log('Listening Port :  3005'));
```

Data that is passed in the postman

```
{
    "name": "Nishanth46",
    "email": "nishanth@gmail.com"
}
```

```
PS D:\express> node postAPI-Auth.js
Listening Port :  3005
request Nishanth46
```

How to Pass headers in the API

HTTP Headers are an important part of the API request and response as they **represent the meta-data associated with the API request and response**. Headers carry information for: Request and Response Body. Request Authorization. Response Caching.

```
const express = require('express');
const basicAuth = require('basic-auth');
const bodyParser = require('body-parser');

const app = express();

// Use body-parser middleware to parse incoming JSON data
app.use(bodyParser.json());

const data = [];

// Middleware function to require headers
const requireHeaders = (req, res, next) => {
    const requiredHeaders = ['x-header'];
    const missingHeaders = [];

    for (let header of requiredHeaders) {

        // console.log('%c', 'color: #ff0000', requiredHeaders,header,);
        if (!req.headers[header]) {
            missingHeaders.push(header);
        }
    }

    if (missingHeaders.length > 0) {
        res.status(400).send(`Missing required headers: ${missingHeaders.join(', ')}`);
    } else {
        next();
    }
};

```

```
//middleware function for Basic Authentication
const auth = (req, res, next) => {
    const user = basicAuth(req);
    if (!user || user.name !== 'myuser' || user.pass !== 'mypassword') {
        res.status(401).send('Authentication required.');
        return;
    }
    next();
};

// Route for getting data with Basic Authentication and required headers
app.get('/api/userdata', auth, requireHeaders, (req, res) => {
    res.set('X-Header', 'Custom');
    res.json(data);
});

// Route for creating data with Basic Authentication and required headers
app.post('/api/head', auth, requireHeaders, (req, res) => {
    // Get the data from the request body
    const newData = req.body;

    // Generate a new ID for the data
    const newId = data.length + 1;

    // Add the new data to the data array
    data.push({ id: newId, ...newData });

    // Set the headers
    res.set('X-Header', 'Custom');

    // Send a response with the newly created data
    res.json(data[data.length - 1]);
```

```
    console.log('%c%s', 'color: #00e600', req.headers);
});

// Start the server
app.listen(3007, () => console.log('Listening Port :  3007'));
```

Headers that is passed in the postman

	Key	Value
☑	X-Header	Custom
	Key	Value

```
PS D:\express> node postAPI-Head.js
Listening Port :  3007
{
  'x-header': 'Custom',
  'content-type': 'application/json',
  authorization: 'Basic bXl1c2VyOm15cGFzc3dvcmQ=',
  'user-agent': 'PostmanRuntime/7.32.2',
  accept: '*/*',
  'cache-control': 'no-cache',
  'postman-token': 'db77f2ec-4717-42a1-86ff-031d965e07ef',
  host: 'localhost:3007',
  'accept-encoding': 'gzip, deflate, br',
  connection: 'keep-alive',
  'content-length': '66'
}
```

Arun

GET & Post API with path parameter- Path parameter <Postman Testing>
Testing the GET & Post API with one path parameter
Testing the GET & Post API with Multiple path parameter - Arun

Petchi Kumar

GET & POST API with Query parameter
Testing the GET & Post API with one query parameter
Testing the GET & Post API with Multiple query parameter

GET & POST Method using Query Parameter

Query parameter :

- A query parameter, also known as a query string parameter, is a part of a URL used to pass information to a web server as a key-value pair.

- It is located after the "?" symbol in a URL and is composed of one or more key-value pairs separated by the "&" symbol

Example:

http://localhost:3000/hello?name=nisanth

GET Method query parameter :

- In Get method , request to get a single data using the query parameter.

- The request parameter passed with authorization and headers for this method.

Aswin

PATCH & DELETE API with path parameter
Testing the PATCH & DELETE API with one path parameter
Testing the PATCH & DELETE API with Multiple path parameter

Nishanth

PATCH & DELETE API with path parameter
Testing the PATCH & DELETE API with one query parameter
Testing the PATCH & DELETE API with Multiple query parameter
Difference the method names(Post, Delete)

Router in Express Json
Router with Single File
Router with Multiple File
Testing with router with single & multiple files

Router in Express JSON