Express.js

Tutorial

Express Js

what is express js

Express.js is a minimalist web application framework for Node.js, designed to simplify building web applications and APIs. It provides a layer of fundamental web application features, making it easier to handle server-side logic, routing, and HTTP requests.

Key features of Express.js

•Routing system: Express makes it simple to define how an application responds to specific client requests (e.g., `GET`, `POST`) to different endpoints or URLs.

•Middleware support: Middleware functions are the backbone of Express, providing a flexible way to add functionality at different stages of the request-response cycle. This can include tasks like authentication, logging, or parsing request bodies.

Express Js

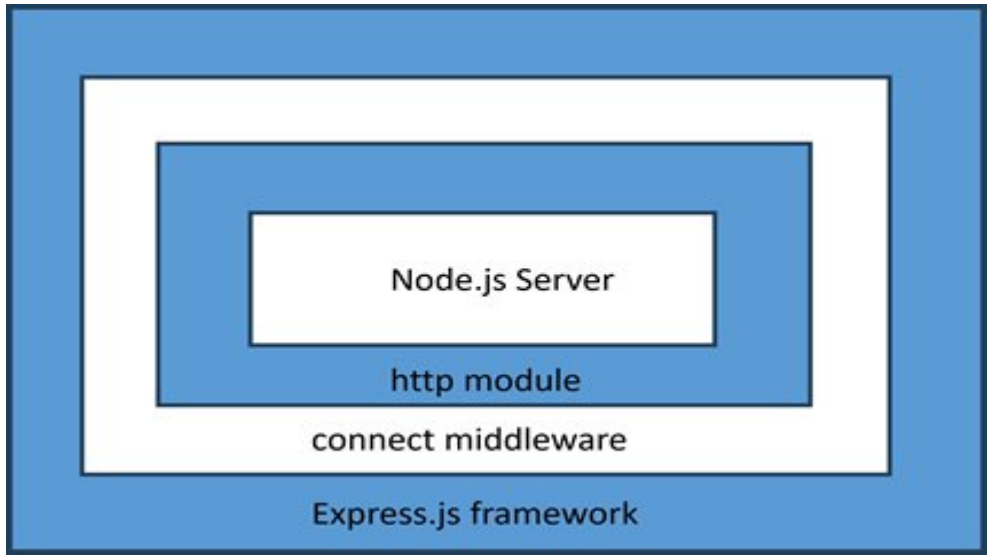
- Minimal and flexible: It provides a thin layer over Node.js's core features, allowing developers to structure and customize their applications without forcing a rigid architecture.
- Fast and lightweight: Its minimalistic design means it's not burdened with unnecessary features, resulting in high performance and low overhead.
- API development: Express is widely used for building robust RESTful APIs, providing an easy-to-use set of HTTP utility methods.
- Static file serving: The framework includes built-in middleware (`express.static`) to serve static assets like images, CSS, and JavaScript files.
- Integration with templating engines: Express supports popular templating engines like Pug (formerly Jade) and EJS to help generate dynamic HTML content on the server side.

Express.js vs. Node.js

Express.js is a framework that runs *on top of* Node.js, not a replacement for it. You can think of the relationship as:

•Node.js: The core JavaScript runtime environment for executing server-side code.

•Express.js: The toolset that simplifies and enhances Node.js's built-in functionality for building web applications



Express Js

What is Express.js used for?

Express.js is a popular choice for building various backend applications, including:

•**Single-Page Applications (SPAs):** Provides the API backend that modern frontend frameworks like React or Angular can consume.

•**RESTful APIs:** Quickly creates powerful APIs that handle communication between different services or applications.

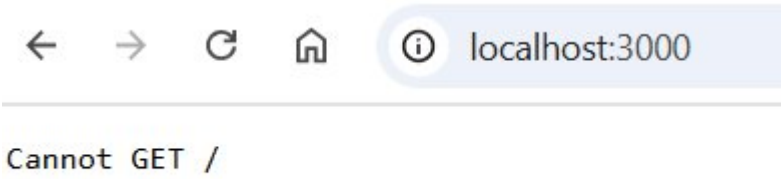
•**Real-time applications:** Can be used with other libraries like Socket.io to build real-time web features such as live chat.

•**Full-stack development:** It is a key component of popular stacks like MEAN (MongoDB, Express, Angular, Node.js) and MERN (MongoDB, Express, React, Node.js).

Express Js

```
npm init
npm i nodemon -D
npm i express
```

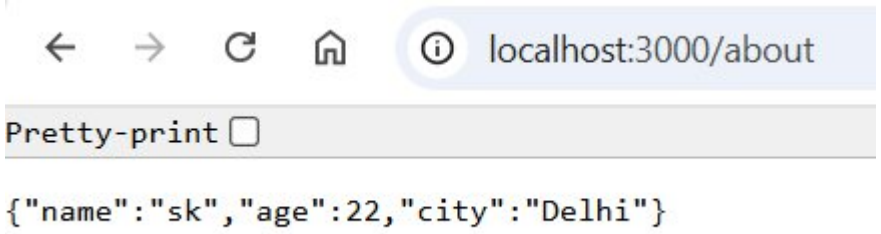
```
//create a server with express
const express = require('express');
const app = express();
const port = 3000;

app.listen(3000, () => {
    console.log(`Example app listening on port ${port}`);
});
```




```
app.get('/', (req, res) => {
    res.send('Hello World!');
});
```

Express Js

```
app.get('/about', (req, res) => {
    res.json({
        name: "sk",
        age: 22,
        city: "Delhi"
    });
}
);
```



Express Js

Install the POSTMAN to test API calls

--postman.com

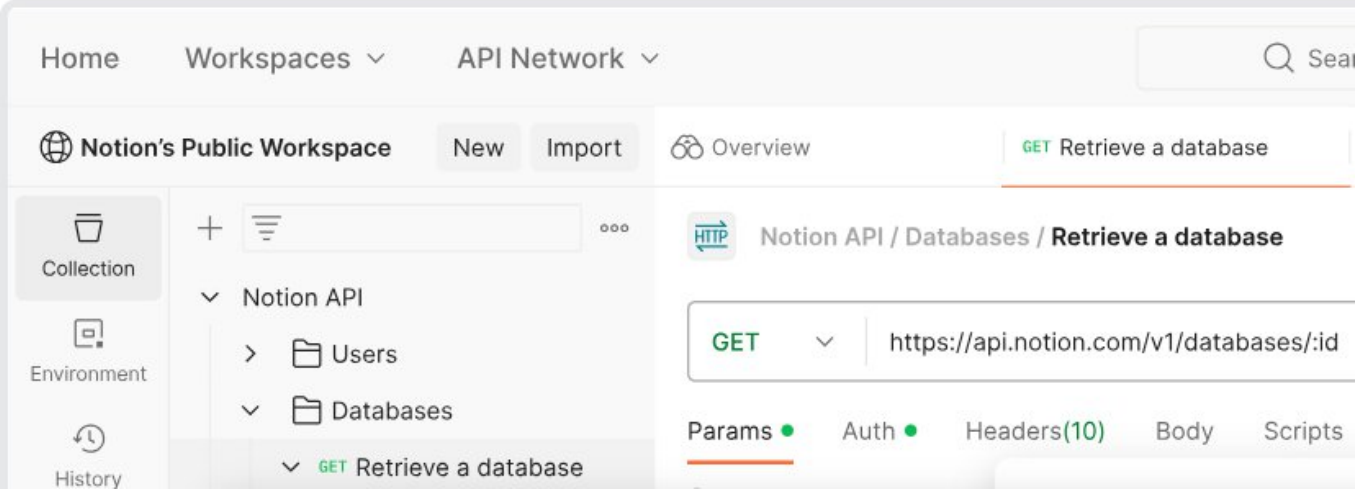
Download Postman

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

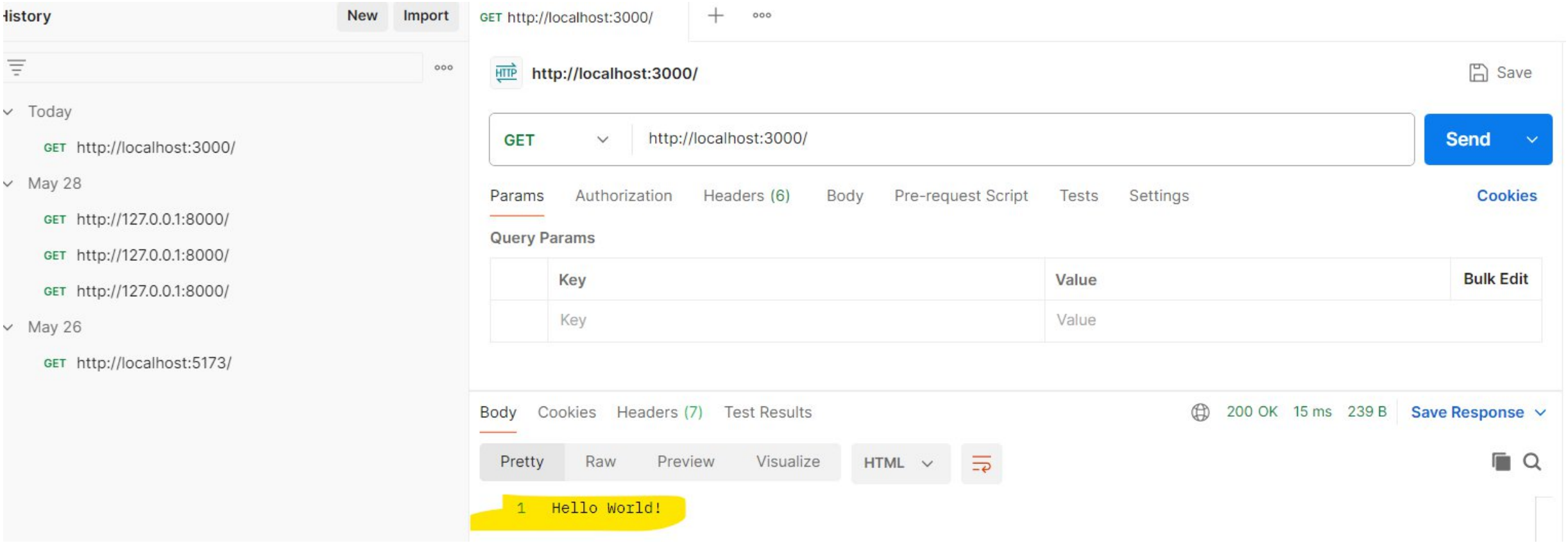
The Postman app

Download the app to get started with the Postman API Platform.

Windows 64-bit



Express Js


History
New
Import
Today
GET http://localhost:3000/
May 28
GET http://127.0.0.1:8000/
GET http://127.0.0.1:8000/
GET http://127.0.0.1:8000/
May 26
GET http://localhost:5173/
GET http://localhost:3000/
http://localhost:3000/
Save
GET
http://localhost:3000/
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Bulk Edit
Key
Value
Body
Cookies
Headers (7)
Test Results
200 OK
15 ms
239 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 Hello World!

Express Js

http://localhost:3000/about

Save

GET http://localhost:3000/about Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

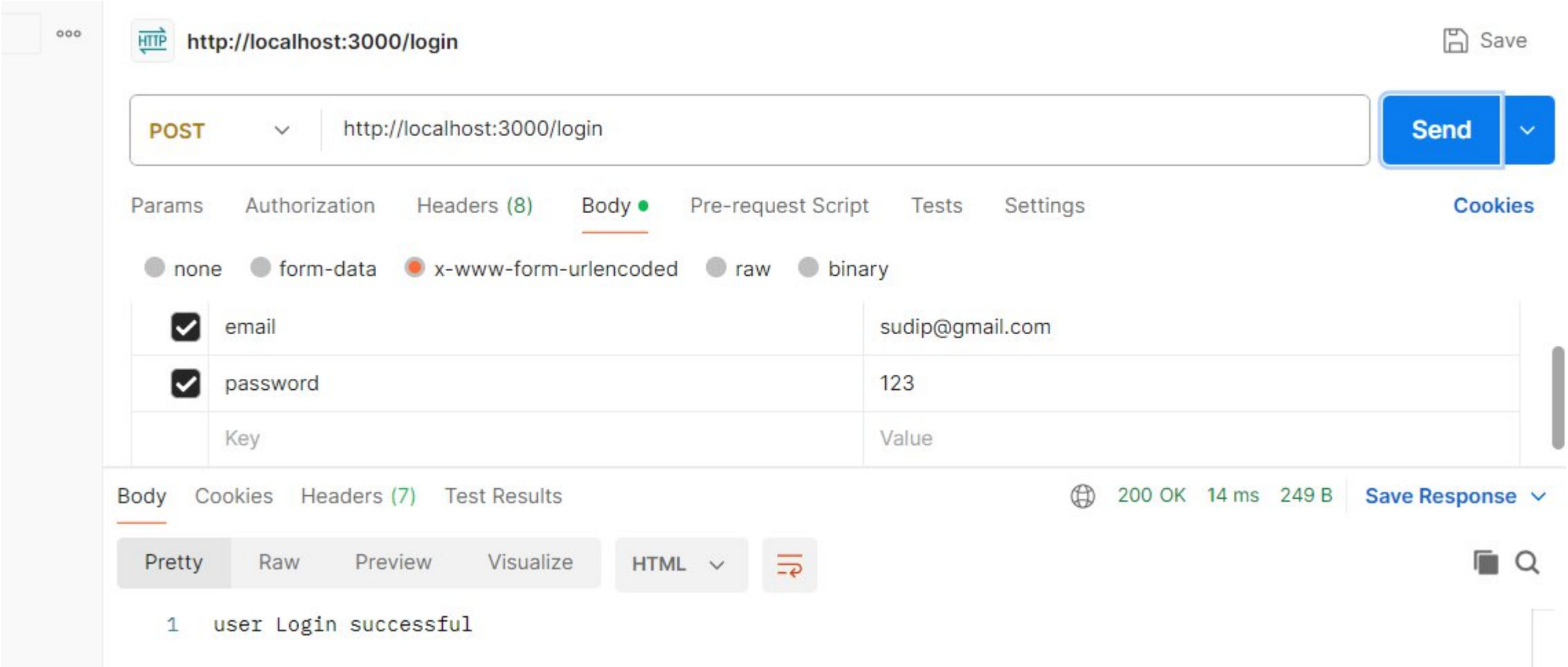
Body Cookies Headers (7) Test Results 200 OK 4 ms 272 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "name": "sk",
    "age": 22,
    "city": "Delhi"
}
```

Express Js

```
app.post('/login', (req, res) => {
    console.log('Login Request Received');
    console.log(req.body); // Assuming body-parser
middleware is used

    res.send('user Login successful');
});
```



Express Js

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Example app listening on port 3000
Login Request Received
undefined
```

npmjs.com/package/body-parser

Pro Teams Pricing Documentation

npm Search packages

body-parser DT

2.2.0 • Public • Published 6 months ago

```
// parse application/x-www-form-urlencoded
app.use(express.urlencoded())

// parse application/json
app.use(express.json())
```

npmjs.com/package/body-parser

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

const app = express()

// parse application/x-www-form-urlencoded
app.use(bodyParser.urlencoded())

// parse application/json
app.use(bodyParser.json())

app.use(function (req, res) {
  res.setHeader('Content-Type', 'text/plain')
  res.write('you posted:\n')
  res.end(String(JSON.stringify(req.body, null, 2)))
})
```

```
[nodemon] starting `node index.js`
Example app listening on port 3000
Login Request Received
{ email: 'sudip@gmail.com', password: '123' }
```

Express Js

```
app.post('/login', (req, res) => {
   console.log('Login Request Received');
   console.log(req.body.email); // Assuming body-
parser middleware is used
   console.log(req.body.password); // Assuming body-
parser middleware is used
   res.send('user Login successful');
});
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Example app listening on port 3000
Login Request Received
sudip@gmail.com
123
```

Middleware in express

In Express.js, middleware refers to 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. These functions are executed in the order they are defined and can perform various tasks:

- **Execute any code:**
  Perform operations like logging, data processing, or conditional logic.
- **Modify the request and response objects:**
  Add properties, change headers, or alter the body of the request or response.
- **End the request-response cycle:**
  Send a response to the client and prevent further middleware from executing. This is typically done by route handlers or error-handling middleware.
- **Invoke the next middleware function:**
  Call `next()` to pass control to the next middleware in the stack. If `next()` is not called, the request will be left hanging unless a response is sent.

Express Js

Types of Middleware:

- **Application-level middleware:**
  Applied to all requests using `app.use()`.
- **Router-level middleware:**
  Applied to specific routes or a group of routes using `router.use()` or directly within route definitions.
- **Built-in middleware:**
  Provided by Express itself, such as `express.static` for serving static files.
- **Third-party middleware:**
  Libraries developed by others and installed via npm, like `body-parser` for parsing request bodies or `morgan` for logging.
- **Error-handling middleware:**
  Special middleware functions with four arguments (`err, req, res, next`) designed to catch and handle errors.

Express Js

```
//middleware
app.use((req, res, next) => {
    console.log('Hello from Middleware');

});
```

```
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Example app listening on port 3000
Hello from Middleware
```

```
//middleware
app.use((req, res, next) => {
    console.log('Hello from Middleware');
    next();
});
//routes
app.get('/', (req, res,next) => {
    res.send('Hello World!');
}
);
```

GET http://localhost:3000/

Params Authorization Headers (8) Body Pre-request Script

none form-data x-www-form-urlencoded raw bina

☑	email
☑	password
	Key

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize HTML

```
1  Hello World!
```

Express Js

Create a custom logger-

```
//middleware
app.use((req, res, next) => {
    console.log(req.method);
    console.log(req.protocol);
    console.log(req.get('host'));
    console.log(req.originalUrl);
    next();
});
```

```
[nodemon] starting `node index.js`
Example app listening on port 3000
POST
http
localhost:3000
/login
Login Request Received
sudip@gmail.com
123
```

Thank you!!