Web-www.webisoftech.com
E-mail- enquiry@webisoftech.com
Phone- +91 9766962674

Express.js Middleware

Express.js Middleware are different types of functions that are invoked by the Express.js routing layer before the final request handler. As the name specified, Middleware appears in the middle between an initial request and final intended route. In stack, middleware functions are always invoked in the order in which they are added.

Middleware is commonly used to perform tasks like body parsing for URL-encoded or JSON requests, cookie parsing for basic cookie handling, or even building JavaScript modules on the fly.

//.....

What is a Middleware function

Middleware functions are the functions that access to the request and response object (req, res) in request-response cycle.

A middleware function can perform the following tasks:

1. It can execute any code.

2. It can make changes to the request and the response objects.

3. It can end the request-response cycle.

4. It can call the next middleware function in the stack.

//.....

Express.js Middleware

Following is a list of possibly used middleware in Express.js app:

1. Application-level middleware

2. Router-level middleware

3. Error-handling middleware

4. Built-in middleware

5. Third-party middleware

Let's take an example to understand what middleware is and how it works.

Let's take the most basic Express.js app:

File: simple_express.js

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Welcome to JavaTpoint!');
});
app.get('/help', function(req, res) {
  res.send('How can I help You?');
});
var server = app.listen(8000, function () {
  var host = server.address().address
  var port = server.address().port
console.log("Example app listening at http://%s:%s", host, port)
})
```

You see that server is listening.

Now, you can see the result generated by server on the local host http://127.0.0.1:8000

Output:

Let's see the next page: http://127.0.0.1:8000/help

Output:

Note: You see that the command prompt is not changed. Means, it is not showing any record of the GET request although a GET request is processed in the http://127.0.0.1:8000/help page.

//.....

Web-www.webisoftech.com
E-mail- enquiry@webisoftech.com
Phone- +91 9766962674

Use of Express.js Middleware

If you want to record every time you a get a request then you can use a middleware.

See this example:

File: simple_middleware.js

```
var express = require('express');
var app = express();
app.use(function(req, res, next) {
  console.log('%s %s', req.method, req.url);
  next();
});
app.get('/', function(req, res, next) {
  res.send('Welcome to JavaTpoint!');
});
app.get('/help', function(req, res, next) {
  res.send('How can I help you?');
});
var server = app.listen(8000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

//.....

You see that server is listening.

Now, you can see the result generated by server on the local host http://127.0.0.1:8000

You can see that output is same but command prompt is displaying a GET result.

Go to http://127.0.0.1:8000/help

As many times as you reload the page, the command prompt will be updated.

Web-www.webisoftech.com
E-mail- enquiry@webisoftech.com
Phone- +91 9766962674

Note: In the above example next() middleware is used.

Middleware example explanation

1. In the above middleware example a new function is used to invoke with every request via app.use().

2. Middleware is a function, just like route handlers and invoked also in the similar manner.

3. You can add more middlewares above or below using the same API.