What is app.use in Express?

Express is a routing and middleware web framework that has minimal functionality of its own: An Express application is essentially a series of middleware function.

**use** is a method to configure middleware and the middleware is used by the routes of the Express HTTP server object.

To setup your middleware, you can invoke **app.use**(<specific\_middleware\_layer\_here>) for every middleware layer that you want to add (it can be generic to all paths, or triggered only on specific path(s) your server handles), and it will add onto your Express middleware stack. Middleware layers can be added one by one in multiple invocations of **use**, or even all at once in series with one invocation. See use documentation for more details.

**middleware stack (app.stack) looks like as follows:**

stack:

[ { route: '', handle: [Function] },

{ route: '', handle: [Function: static] },

{ route: '', handle: [Function: bodyParser] },

{ route: '', handle: [Function: cookieParser] },

{ route: '', handle: [Function: session] },

{ route: '', handle: [Function: methodOverride] },

{ route: '', handle: [Function] },

{ route: '', handle: [Function] } ]

I called app.use(express.bodyParser()), app.use(express.cookieParser()), etc, which added these express middleware 'layers' to the middleware stack. Notice that the routes are blank, meaning that when I added those middleware layers I specified that they be triggered on any route.

Each layer is essentially adding a function that specifically handles user request and Middleware is a section of Layers.

Example 01:

**app.use()** used to Mounts the middleware function or mount to a specified path,the middleware function is executed when the base path matches.

**For example: if you are using app.use() in indexRouter.js , like this:**

//indexRouter.js

var adsRouter = require('./adsRouter.js');

module.exports = function(app) {

app.use('/ads', adsRouter);

}

In the above code app.use() mount the path on '/ads' to adsRouter.js.

Now in adsRouter.js

// adsRouter.js

var router = require('express').Router();

var controllerIndex = require('../controller/index');

router.post('/show', controllerIndex.ads.showAd);

module.exports = router;

in adsRouter.js, the path will be like this for ads- '/ads/show', and then it will work according to controllerIndex.ads.showAd().

app.use([path],callback,[callback]) : we can add a callback on the same.

app.use('/test', function(req, res, next) {

// write your callback code here.

});

Example 02:

app.use() acts as a middleware in express apps. Unlike app.get() and app.post() or so, you actually can use app.use() without specifying the request URL. In such a case what it does is, it gets executed every time no matter what URL's been hit.

Example 03:

Express servers themselves are a stack of middlewares.

// express

var app = express();

// middleware

var stack = middleware();

Then you can add layers to the middleware stack by calling .use

// express

app.use(express.static(..));

// middleware

stack.use(function(data, next) {

next();

});

A layer in the middleware stack is a function, which takes n parameters (2 for express, req & res) and a next function.

Middleware expects the layer to do some computation, augment the parameters and then call next.

A stack doesn't do anything unless you handle it. Express will handle the stack every time an incoming HTTP request is caught on the server. With middleware you handle the stack manually.

// express, you need to do nothing

// middleware

stack.handle(someData);

//