ARYATECHNO

Middleware in Express.js

Topics : Node js
Written on November 30, 2023

Middleware functions have access to the request (`req`) and response (`res`) objects, and they can also call the next middleware function in the stack.

Here are some key points about middleware in Express.js:

Middleware Function: Middleware functions in Express.js are functions that have three parameters: `req` (request object), `res` (response object), and `next` (a function to call the next middleware function in the stack). They can perform tasks such as modifying the request or response objects, ending the request-response cycle, or calling the next middleware function.

Example middleware function:

```
function myMiddleware(req, res, next) {
  // Do something with the request or response
  console.log('Middleware executed');

  // Call the next middleware function
  next();
}
```

Application-Level Middleware: Middleware can be applied at the application level using `app.use()` or for specific routes using `app.use('/path', myMiddleware)`. These middleware functions will be executed for every incoming request.

Example:

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

// Application-level middleware
app.use(myMiddleware);

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

app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

Router-Level Middleware: Middleware can also be applied at the router level using `router.use()`. This allows you to define middleware specific to a set of routes.

Example:

```
const express = require('express');
const router = express.Router();

// Router-level middleware
router.use(myMiddleware);

// Route handler
router.get('/route', (req, res) => {
  res.send('This is a route with middleware');
});

module.exports = router;
```

Error Handling Middleware: Middleware functions with four parameters (`err`, `req`, `res`, `next`) are considered error-handling middleware. They are used to handle errors that occur during the execution of previous middleware or route handlers.

Example:

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send('Something went wrong!');
});
```

Built-in Middleware: Express.js comes with several built-in middleware functions that handle common tasks such as serving static files (`express.static`), parsing incoming request bodies (`express.json`, `express.urlencoded`), and more.

Example:

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

// Built-in middleware to parse JSON bodies
app.use(express.json());

// Route handler
app.post('/api/data', (req, res) => {
  console.log(req.body);
  res.send('Data received');
});
```