

MORE ON EXPRESS AND NODE.JS

ESTR2106 2022-23 Term 1

Building Web Applications

Dr. Chuck-jee Chau chuckjee@cse.cuhk.edu.hk

OUTLINE

- Writing Middleware
- Error Middleware
- Third-party Middleware
- Node.js accessing filesystem and OS

MIDDLEWARE

• Middleware is a function in the form

```
function (req, res, next) { ... }

When an
Express app receives a request middleware middleware middleware
middleware
```

- An Express application is essentially a series of middleware calls
- Routing defining how middleware(s) are used to handle a request

MIDDLEWARE

- An Express middleware can do these:
 - Execute any code
 - Make changes to req and/or res objects
 - End the request-response cycle
 - Call the next middleware in the stack
- Here is a call to a very simple middleware...

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

Middleware to send a response, without calling further middleware

WRITING MIDDLEWARE

- As a callback, middleware can be declared as a separate function
 - Note: Order of loading is important!

WRITING MIDDLEWARE

- With app.use(path, callback), a path can be specified so that the middleware is only applicable for certain requests
 - e.g., app.use('/abc', f1) will limit the middleware f1 to be used for any request for path /abc
- More middleware can be specified for one path
 - A list of callbacks, separated by comma:
 app.use(path, f1, f2, f3)
 - Or with an array of callbacks:
 app.use(path, [f1, f2, f3])

ERROR MIDDLEWARE

- An error-handling middleware has four arguments (err, req, res, next)
 - A required signature to be recognized by Express
- The error-handling middleware should be defined last, after *all* other middleware and routes

THIRD-PARTY MIDDLEWARE

- Some more middleware are commonly used:
 - See: https://expressjs.com/en/resources/middleware.html
- cors is a package for enabling CORS with options
 - See: http://expressjs.com/en/resources/middleware/cors.html

```
const cors = require('cors');
// enabling cors for ALL route
app.use(cors());
// enabling cors with pre-flight request: special HTTP methods
app.option(path, cors());
```

- CORS Pre-flight: a special HTTP OPTION step is needed to agree on extra settings, such as HTTP methods to allow
 - See: https://www.baeldung.com/cs/cors-preflight-requests

NODE.JS ACCESSING FILESYSTEM

- Unlike browsers, Node.js allows scripts to access the filesystem
 - Many filesystem actions are available, e.g., chmod, utime, rmdir, ...
 - See: <u>https://nodejs.org/api/fs.html#fsappendfilepath-data-options-callback</u>

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

app.all('/*', (req,res)=> {
            filePath = __dirname + '/log.txt';
            let s = req.originalUrl; // the full path in the request
            fs.appendFile(filePath, s, (err) => {if (err) console.log(err)});
            res.send("Hi")
});

const server = app.listen(3000);
```

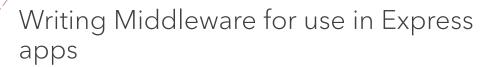
NODE.JS ACCESSING OS

- Node.js can access system-related utility methods and properties
 - Some useful information is available, e.g., cpu, loadavg, ...
 - See: https://nodejs.org/api/os.html

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

app.get('/cpu', (req,res) => res.send(os.cpus()));

const server = app.listen(3000);
```



http://expressjs.com/en/guide/writing-middleware.html

Node.js documentation

https://nodejs.org/api

READ FURTHER...