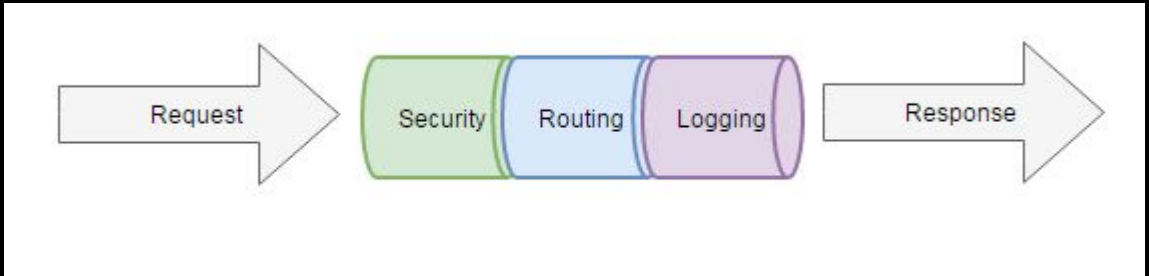
BCDV 1041
Express & Middleware

2023 Fall
week 01 - class 03

Middleware

- Code that sits between two layers of software
- With Express, the middleware is sitting between the request and response



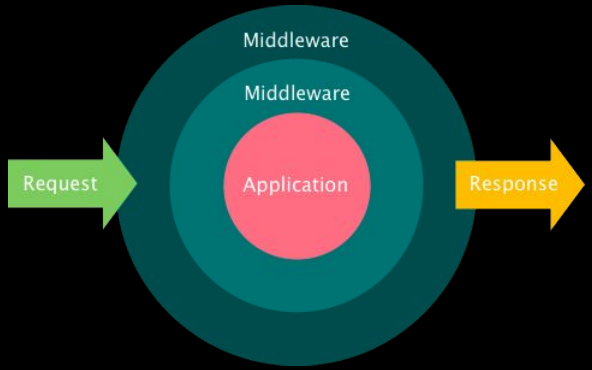
Middleware

- Middleware is a pipeline of code that gets called before your request handler

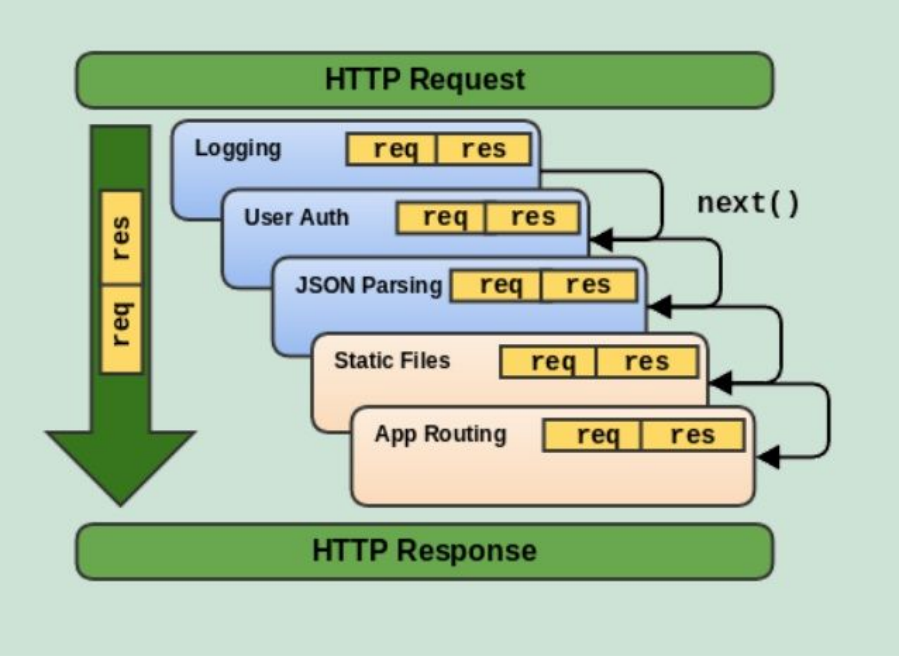
- Express applications are basically a bunch of middleware calls

Middleware can:

- Execute any code
- Make changes to the request and the response objects
- End the request-response cycle
- Call the next middleware in the stack



Express Middleware



- Middleware is a function with access to the **request object (req)** and the **response object (res)**
- Also, has access to the **next middleware object (next)** in line in request-response cycle of Express application

Express Middleware modules

- Some useful Express maintained middleware and 3rd party middleware can be found here:
  - https://expressjs.com/en/resources/middleware.html

- Some popular middleware include:
  - CookieParser - parse cookie header
  - BodyParser - parse the HTTP request body (prior to Express v4.1.6)
  - Passport - simple, unobtrusive authentication for Node.js

Serving Static Files with Express

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

// Serve static files from the "public" directory
app.use(express.static('public'));

// Define a route for the homepage
app.get('/', (req, res) => {
  res.send('Welcome to the homepage!');
});

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

- To serve static files such as images, CSS files and JavaScript files use the **express.static** built-in middleware function in Express
- The function signature is **express.static**(root, [options]) where root is the root directory from which the serve the static assets.
- For example, to serve images, CSS files and JavaScript from the public directory use.

Express - Built in Middleware - express.urlencoded()

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

const app = express();

// Middleware to parse URL-encoded data
app.use(express.urlencoded({ extended: true }));

// Define a route to handle a POST request
app.post('/submit', (req, res) => {
  // Access the parsed data from the request body
  const { username, email } = req.body;

  // Do something with the data (e.g., save it to a database)
  // For simplicity, we'll just send a response back
  res.send(`Received data: Username - ${username}, Email - ${email}`);
});
```

- The `express.urlencoded()` middleware is built into Express.js and is used to parse incoming request data with a content type of `application/x-www-form-urlencoded`. This middleware is often used when dealing with form submissions from HTML forms.

Express - Custom Middleware

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

// Custom Logger Middleware
const customLogger = (req, res, next) => {
  const timestamp = new Date().toLocaleString();
  const method = req.method;
  const url = req.url;

  console.log(`[${timestamp}] ${method} request to ${url}`);

  next(); // Call the next middleware in the chain
};

// Use the custom logger middleware for all routes
app.use(customLogger);

// Route handlers
app.get('/', (req, res) => {
  res.send('Hello, Express!');
```

Custom middleware in Express.js allows you to add your own functionality to the request/response handling process. It is a function that has access to the request (`req`), response (`res`), and `next` function.

Video Express & Middleware

Express - Built in Middleware - express.json()

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

// Middleware to parse JSON request bodies
app.use(express.json());

app.post('/api/posts', (req, res) => {
  // Access request body data using req.body
  const postData = req.body;
  // Process the data as needed
  res.json({ message: 'Post received', data: postData });
});

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

- Starting from Express.js version 4.16 and later, the `body-parser` middleware is no longer required as it is included in the Express.js core.
- Prior to version 4.16, developers had to explicitly install and use `body-parser` to parse the request body for data such as JSON, form data, and other content types. However, Express.js now includes this functionality by default, making it easier for developers to work with request bodies.
- With the built-in body parsing, you can access request body data using the `req.body` object without needing to include and configure `body-parser` separately.

Express Router & Generator

Express Router

- Use the express.Router class to create modular, mountable route handlers.
- A Router instance is a complete middleware and routing system; for this reason, it is often referred to as a "mini-app".

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

// middleware that is specific to this router
router.use(function timeLog (req, res, next) {
  console.log('Time: ', Date.now())
  next()
})
```

express.Router

- Load the module in the main app.js, the app will now be able to handle /birds and /birds/about

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

// define the home page route
router.get('/', function (req, res) {
  res.send('Birds home page')
})
// define the about route
router.get('/about', function (req, res) {
  res.send('About birds')
})

module.exports = router
```

```
var birds = require('./birds')

// ...

app.use('/birds', birds)
```

Express application generator

- Use the application generator tool, express-generator, to quickly create an application skeleton.

- The express-generator package installs the express command-line tool. Use the following command to do so

```
$ npm install express-generator -g
```

Using template engines with Express

- A template engine enables you to use static template files in your application. At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client.

- Some popular template engines that work with Express are Pug, Mustache, and EJS. The Express application generator uses Jade as its default, but it also supports several others.

```
$ npm install pug --save
```