Rutas y Middlewares

¿Qué son las rutas en Express.js?

Las rutas son puntos de acceso que una aplicación web ofrece para procesar solicitudes HTTP específicas, como GET, POST, PUT o DELETE. En Express.js, una ruta incluye:

- Vía de acceso de ruta: URL específica donde se accede.
- Método HTTP: GET, POST, etc.
- Manejador de ruta: Función que maneja la solicitud y genera una respuesta.

La ruta tiene la siguiente estructura:

```
app.metodo(via-de-acceso, manejador)
```

Ejemplos:

Responde con “Hello World” en la página inicial.

```
app.get('/', function (req, res) {
  res.send('Hello World!');
});
```

Responde con “Got a POST request” a una solicitud POST con la ruta raíz (/).

```
app.post('/', (req, res) => {
  res.send('Got a POST request');
});
```

Responde con “¡Bienvenido a mi aplicación!” a cualquier solicitud GET con ruta especifica a /welcome.

```
app.get('/welcome', (req, res) => {
    res.send('¡Bienvenido a mi aplicación!');
});
```

Rutas Dinámicas

Las rutas dinámicas permiten manejar URLs que cambian según ciertos parámetros. Para crear una ruta dinámica se usa el operador **:** seguido por el parámetro dinámico. Por ejemplo:

```
app.get('/user/:id', (req, res) => {
    const userId = req.params.id;  // Captura el parámetro de la URL
    res.send(`Usuario con ID: ${userId}`);
});
```

:id indica el parámetro dinámico, en este caso la ID del usuario y req.params.id accede al valor dado por la URL.

¿Qué son los Middlewares?

Un middleware es una función que se ejecuta entre dos partes de la aplicación. Comúnmente antes que llegue al manejador de rutas o antes de que el cliente reciba la respuesta. Estas funciones tienen acceso a la solicitud (req) y la respuesta (res) al igual que la siguiente función de middleware (next). Las funciones middleware pueden realizar lo siguiente:

- Puede modificar el req o res.
- Ejecutar un bloque de código.
- Finalizar el ciclo de solicitud-respuesta.
- Invocar la siguiente función middleware con next().

Middlewares en Express.js

Una aplicación con Express.js puede utilizar diferentes tipos de middleware. Entre estos podemos encontrar los siguientes:

1. Middleware incorporado

Middleware que esta incorporado en el propio Express sin necesidad de instalar e importar módulos externos. Por ejemplo, la función express.static, esta función es responsable por el servicio de activos estáticos de una aplicación. Esta función cuenta con la siguiente estructura:

```
app.use(express.static('raiz', opciones));
```

Donde *raíz* se refiere al directorio desde el que se realiza el servicio de activos estáticos, y *opciones* es un objeto, que puede no ser incluido, que contiene algunas propiedades como dotfiles, etag, index, etc.

2. Middleware de terceros

El middleware de terceros se refiere a módulos/bibliotecas que se pueden integrar a una aplicación para añadir nuevas funcionalidades. El paquete de body-parser que se vio anteriormente en este curso es un middleware de tercero.

3. Middleware de nivel de aplicación

Este tipo de middleware esta directamente vinculado a una instancia de aplicación utilizando app.use() para correrlo para cada solicitud o app.metodo() para correrlo para algún método en específico. Funciona para la ruta raíz (/) o rutas específicas (/ruta), dependiendo de la vía de acceso, y se utiliza para procesar o manipular solicitudes y respuestas. Por ejemplo:

```
app.use('/',(req, res, next) => {
   console.log('Middleware 1');
   next(); // Pasa al siguiende middleware en la pila
});
```

Este middleware se corre para cada solicitud que tiene como vía de acceso a la ruta raíz y luego pasara a la siguiente función.

Middlewares Personalizados

Un middleware personalizado es una función que es creada con el propósito de manejar tareas especificas para cumplir ciertos requisitos de la aplicación. Esta función cuenta con un mínimo de tres parámetros, la solicitud, la respuesta, la siguiente función. Usualmente cuentan con la siguiente estructura:

```
const nombre_funcion = (req, res, next) => {
    //Bloque de codigo
    next(); // Pasa a la siguiente funcion middleware
};
```

Ahora, tome el la siguiendo función middleware personalizada.

```
const logger = (req, res, next) => {
    console.log(`Request Method: ${req.method}, URL: ${req.url}`);
    next(); // Pasa a la siguiente funcion middleware
};
```

Esta función se puede utilizar de diferentes maneras:

1. Aplicarla globalmente con app.use().

```
app.use(logger);
```

2. Aplicarla para una ruta en específico.

```
app.use('/secure', logger);
```

3. Aplicarla para una ruta y un método en especifico

```
app.get('/login', logger, (req, res) => {
    res.send('Main');
});
```