Extra: Middlewares

Conceito de Middlewares no Express

O que são Middlewares?

Middlewares são funções no Express.js que têm acesso ao objeto de solicitação (request), ao objeto de resposta (response) e à próxima função middleware no ciclo de requisição-resposta de uma aplicação. Elas podem executar qualquer tipo de código, modificar os objetos de requisição e resposta, encerrar o ciclo de requisição-resposta, ou até mesmo chamar o próximo middleware na pilha.

A estrutura básica de uma função middleware é a seguinte:

```
function middleware(req, res, next) {
    // Lógica do middleware
    next() // Chama o próximo middleware na fila
}
```

- `req`: O objeto de solicitação que contém informações sobre a solicitação HTTP.
- `res`: O objeto de resposta que será enviado de volta ao cliente.
- `next`: Uma função que é chamada para passar o controle ao próximo middleware.

Importância dos Middlewares

Os middlewares são fundamentais no Express por várias razões:

1. **Modularização**: Permitem a divisão do código em funções reutilizáveis, facilitando a manutenção e organização.
2. **Fluxo de Processamento**: Controlam o fluxo de processamento das requisições, possibilitando a execução de operações específicas antes que a resposta seja enviada ao cliente.

3. **Manipulação de Erros**: Facilitam o tratamento de erros de forma centralizada, melhorando a robustez da aplicação.
4. **Autenticação e Autorização**: São amplamente utilizados para verificar se o usuário tem permissão para acessar certos recursos.
5. **Validação de Dados**: Permitem validar e sanitizar os dados de entrada antes de prosseguir com a lógica principal da aplicação.

Tipos de Middlewares

No Express, existem diferentes tipos de middlewares, cada um com sua finalidade específica:

1. **Middlewares de Aplicação**: Afetam toda a aplicação ou um conjunto específico de rotas.
2. **Middlewares de Rota**: Executados apenas em rotas específicas.
3. **Middlewares Incorporados**: Middlewares que vêm embutidos no Express, como `express.static()` para servir arquivos estáticos.
4. **Middlewares de Tratamento de Erros**: Utilizados para capturar e processar erros em uma aplicação Express.
5. **Middlewares de Terceiros**: Pacotes externos que podem ser integrados ao Express, como `body-parser`, `morgan` para logs, etc.

Exemplo Prático de Middleware

A seguir, um exemplo simples para ilustrar o uso de middlewares em uma aplicação Express:

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

// Middleware de aplicação
app.use((req, res, next) => {
    console.log(`${req.method} ${req.url}`)
    next() // Passa o controle para o próximo middleware
})

// Middleware específico para uma rota
app.get('/usuario', (req, res, next) => {
```

```
    res.send('Rota de usuário')
})

// Middleware de tratamento de erros
app.use((err, req, res, next) => {
    console.error(err.stack)
    res.status(500).send('Algo deu errado!')
})

const PORT = 3000
app.listen(PORT, () => {
    console.log(`Servidor rodando na porta ${PORT}`)
})
```

Explicando o Código

1. **Middleware de Aplicação**:
    - Registra um middleware que loga o método e a URL de cada requisição. Esse middleware será executado para todas as rotas da aplicação.
    - A função `next()` é chamada para passar o controle para o próximo middleware, garantindo que a requisição continue sendo processada.
2. **Middleware de Rota**:
    - Um middleware específico para a rota `/usuario`. Quando a rota `/usuario` é acessada, a função middleware responde com a mensagem 'Rota de usuário'.
3. **Middleware de Tratamento de Erros**:
    - Esse middleware captura erros que ocorrem em qualquer parte do ciclo de requisição-resposta. Se um erro ocorrer, ele será registrado no console, e uma mensagem de erro será enviada ao cliente.

Uso de Middlewares Incorporados e de Terceiros

Além de criar seus próprios middlewares, o Express oferece middlewares incorporados para tarefas comuns, como servir arquivos estáticos ou tratar dados de formulários. Por exemplo:

- `express.static()`: Serve arquivos estáticos, como CSS, JavaScript, imagens, etc.
- `express.json()`: Analisa os dados JSON no corpo da requisição.
- `express.urlencoded()`: Analisa os dados de formulários URL-encoded.

Além disso, middlewares de terceiros, como `morgan` para logs de requisições e `cors` para habilitar CORS, são frequentemente utilizados em aplicações Express.

Exemplo com Middleware de Terceiros: `morgan`

```
const morgan = require('morgan')

// Adiciona o middleware 'morgan' para logar todas as requi
sições
app.use(morgan('dev'))
```

Conclusão

Os middlewares são o coração de uma aplicação Express. Eles permitem a criação de fluxos de processamento modularizados, facilitando a manipulação de requisições e respostas de forma eficiente. Entender como os middlewares funcionam e como utilizá-los corretamente é essencial para construir aplicações web robustas e escaláveis com Express.