Aula 07: Validação de Dados com Express

1. Introdução à Validação de Dados

A **validação de dados** é crucial no desenvolvimento de aplicações web monolíticas. Ela garante que os dados recebidos de formulários HTML estejam no formato correto e sigam as regras definidas pelo sistema. Sem validação, a aplicação pode estar sujeita a falhas e comportamentos inesperados.

No Express, a validação pode ser feita manualmente ou com o uso de bibliotecas de validação externas.

2. Configuração de Aplicação Monolítica para Receber Dados via Formulários

Quando trabalhamos com formulários HTML, os dados são enviados ao servidor usando o método `POST` no formato **application/x-www-form-urlencoded**. Para lidar com esse tipo de dados, usamos o middleware `express.urlencoded()`.

Exemplo de Configuração

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

// Middleware para tratar dados de formulários (URL-encode
d)
app.use(express.urlencoded({ extended: true }));

app.listen(3000, () => {
    console.log('Servidor rodando na porta 3000');
});
```

O middleware `express.urlencoded()` permite ao Express interpretar dados enviados em formulários HTML.

---

3. Validação Manual de Formulários

Agora, vamos implementar a validação dos dados enviados através de formulários. Suponha que temos um formulário de cadastro de usuário com campos como **nome**, **email** e **idade**.

Exemplo de Validação Manual de Dados de Formulários

```
<!-- Formulário HTML para envio de dados -->
<form action="/cadastro" method="POST">
    <label for="nome">Nome:</label>
    <input type="text" name="nome" id="nome" required>

    <label for="email">Email:</label>
    <input type="email" name="email" id="email" required>

    <label for="idade">Idade:</label>
    <input type="number" name="idade" id="idade" required>

    <button type="submit">Cadastrar</button>
</form>
```

No lado do servidor, podemos validar esses dados com uma lógica simples:

```
app.post('/cadastro', (req, res) => {
    const { nome, email, idade } = req.body;

    // Validação simples
    if (!nome || nome.trim() === '') {
        return res.status(400).send('Nome é obrigatório');
    }

    const emailRegex = /\S+@\S+\.\S+/;
```

```
    if (!email || !emailRegex.test(email)) {
        return res.status(400).send('Email inválido');
    }

    if (!idade || isNaN(idade) || idade < 18) {
        return res.status(400).send('Idade deve ser um núme
ro maior ou igual a 18');
    }

    res.send('Cadastro realizado com sucesso!');
});
```

Neste exemplo:

- O servidor verifica se o **nome** foi enviado e se não está vazio.
- O **email** é validado com uma expressão regular.
- A **idade** é validada para garantir que o valor seja numérico e maior ou igual a 18.

---

4. Usando Middleware para Validação de Dados de Formulários

Assim como na validação manual, podemos mover essa lógica de validação para middlewares. Isso permite a reutilização do código de validação em várias rotas.

Exemplo de Validação com Middleware

```
// Middleware de validação de cadastro
function validarCadastro(req, res, next) {
    const { nome, email, idade } = req.body;

    if (!nome || nome.trim() === '') {
        return res.status(400).send('Nome é obrigatório');
    }

    const emailRegex = /\S+@\S+\.\S+/;
    if (!email || !emailRegex.test(email)) {
```

```
        return res.status(400).send('Email inválido');
    }

    if (!idade || isNaN(idade) || idade < 18) {
        return res.status(400).send('Idade deve ser um núme
ro maior ou igual a 18');
    }

    next();
}

app.post('/cadastro', validarCadastro, (req, res) => {
    res.send('Cadastro realizado com sucesso!');
});
```

Aqui, a função `validarCadastro` é reutilizada como middleware, permitindo a separação da lógica de validação da lógica de negócio da rota.

---

5. Validação com Zod para Formulários

O **Zod** é uma biblioteca de validação que permite a definição de esquemas de dados de forma declarativa. Vamos usá-lo para validar dados de formulários HTML.

Instalando o Zod

Para usar o Zod, primeiro precisamos instalá-lo no projeto:

```
npm install zod
```

Exemplo de Validação com Zod

```
const { z } = require('zod');

// Definindo um esquema de validação para dados de cadastro
```

```
const cadastroSchema = z.object({
    nome: z.string().min(1, { message: 'Nome é obrigatório'
}),
    email: z.string().email({ message: 'Email inválido' }),
    idade: z.coerce.number().min(18, { message: 'Idade deve
ser maior ou igual a 18' })
});

// Middleware de validação usando Zod
function validarComZod(req, res, next) {
    const validacao = cadastroSchema.safeParse(req.body);

    if (!validacao.success) {
        // Envia os erros de validação
        return res.status(400).send(validacao.error.issues.
map(erro => erro.message).join(', '));
    }

    next();
}

app.post('/cadastro', validarComZod, (req, res) => {
    res.send('Cadastro realizado com sucesso!');
});
```

Explicação do Código

- Usamos o método `z.coerce.number()` para garantir que o campo idade seja convertido em número, pois os dados enviados via formulários vêm como strings.
- O método `safeParse` do Zod retorna um objeto que indica se a validação foi bem-sucedida ou não. Em caso de falha, retornamos os erros de validação.

---

6. Exercícios Práticos

1. **Exercício 1 (Fácil)**:

Crie uma rota POST `/contato` que receba um nome, um email e uma mensagem através de um formulário HTML. Valide que o nome e o email sejam obrigatórios, e que o email siga o formato correto. Implemente essa validação manualmente.

2. **Exercício 2 (Médio)**:

   Crie uma rota POST `/encomenda` que receba um nome, endereço e quantidade de itens. Valide que:

   - O nome e o endereço sejam obrigatórios.
   - A quantidade seja um número inteiro e maior que zero.
   - Utilize o Zod para a validação.

3. **Exercício 3 (Difícil)**:

   Crie uma rota POST `/inscricao` que receba dados de inscrição de evento (nome, email, idade, número de ingressos). Valide que:

   - O nome e o email sejam obrigatórios.
   - O email siga o formato correto.
   - A idade seja maior ou igual a 18.
   - O número de ingressos seja maior ou igual a 1 e menor ou igual a 10.
   - Utilize o Zod para a validação.

Conclusão

A validação de dados no backend é essencial para garantir que os dados recebidos via formulários estejam corretos e seguros. Embora seja possível realizar essa validação manualmente, usar uma biblioteca como o **Zod** pode simplificar e padronizar o processo. Em aplicações monolíticas que utilizam formulários HTML, o middleware `express.urlencoded()` permite processar dados no formato `application/x-www-form-urlencoded`, e as técnicas de validação podem ser facilmente integradas a qualquer fluxo de dados de entrada.