Desenvolvimento Backend

Aula 04 - Criando servidor Web com Express

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula, você conhecerá como criar servidores para a web com a biblioteca Express do Node.js.

Objetivos

- Conhecer como criar operações GET, POST, PUT e DELETE com Express;
- Entender o funcionamento dessas operações e suas aplicações na prática.

Criando servidor web com Express

Link do video da aula: https://youtu.be/VOZdCHZcPl4

Nesta aula, você verá como utilizar o Express. O Express é uma biblioteca que facilita a criação de servidores web com Node.js.

Acompanhe a videoaula para uma explicação detalhada dos exemplos abaixo.

Iniciando com o Express

Para instalar o Express, digite no *prompt* de comando:

```
$ npm install express --save
```

Em seguida, crie um arquivo (index.js), com o código abaixo:

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

app.get('/', (req, res) => {
    res.json({msg: "Hello from Express!"})
})

app.listen(8080, () => {
    console.log('Servidor pronto na porta 8080')
})
```

> Lembre-se de configurar o package.json para executar o arquivo recém-criado, conforme vimos em aulas anteriores.

Feito isso, você terá um servidor web pronto na porta 8080 que responderá a acessos à rota "/". Para acessar, utilize seu navegador no seguinte endereço: http://localhost:8080/

Criando simples API com Express (Parte 1)

Link do video da aula: https://youtu.be/SnUcrIuINgQ

Vamos dar sequência à criação de um servidor web com Express. Para isso, vamos criar um exemplo de uma API (*Application Programming Interface*).

Nesta aula, vamos criar a operação de armazenamento de um aluno através do método POST do HTTP. Por enquanto, os dados serão salvos em memória e não em um banco de dados (veremos mais adiante no curso como utilizar um banco de dados).

Como os dados serão salvos em memória, vamos precisar criar um identificador único para cada aluno salvo. Esse trabalho é normalmente feito pelos próprios bancos de dados, mas neste exemplo faremos nós mesmos.

Para isso, vamos utilizar uma biblioteca chamada uuid que permite criar identificadores únicos universais. Para instalar essa biblioteca execute:

```
$ npm install uuid --save
```

Com tudo instalado, acompanhe a videoaula para explicação detalhada do exemplo abaixo:

```
const express = require('express')
const { v4: uuidv4 } = require('uuid');
const app = express()
app.use(express.json())

const alunos = {}

app.get('/', (req, res) => {
    res.json({msg: "Hello from Express!"})
})

app.post('/alunos', (req, res) => {
    const aluno = req.body
```

```
    const idAluno = uuidv4()
    aluno.id = idAluno
    alunos[idAluno] = aluno
    res.json({msg: "Aluno adicionado com sucesso!"})
})

app.get('/alunos', (req, res) => {
    res.json({alunos: Object.values(alunos)})
})

app.listen(8080, () => {
    console.log('Servidor pronto na porta 8080')
})
```

Como vimos em aulas anteriores, para testar, você pode instalar a extensão do VSCode chamada Rest Client. Com essa extensão, você poderá criar um arquivo de extensão ".http", para simular as requisições.

Abaixo exemplo de arquivo que pode ser utilizado como teste:

```
GET http://localhost:8080/

###

GET http://localhost:8080/alunos

###

POST http://localhost:8080/alunos
Content-type: application/json

{
    "nome": "Pedro"
}
```

Criando simples API com Express (Parte 2)

Link do video da aula: https://youtu.be/JkGrF_AVLRU

Vamos dar continuidade à criação do nosso servidor web adicionando funções para

editar, remover e buscar por determinado identificador.

Acompanhe a videoaula para explicação detalhada do exemplo abaixo:

```
const express = require('express')
const { v4: uuidv4 } = require('uuid');
const app = express()
app.use(express.json())

const alunos = {}

app.get('/', (req, res) => {
    res.json({msg: "Hello from Express!"})
})

app.get('/alunos/:id', (req, res) => {
    res.json({aluno: alunos[req.params.id]})
})

app.put('/alunos', (req, res) => {
    const id = req.query.id
    if (id && alunos[id]){
        const aluno = req.body
        aluno.id = id
        alunos[id] = aluno
        res.json({msg: "Aluno atualizado com sucesso!"})
    }else{
        res.status(400).json({msg: "Aluno não encontrado!"})
    }

})

app.delete('/alunos', (req, res) => {
    const id = req.query.id
    if (id && alunos[id]){
        delete alunos[id]
        res.json({msg: "Aluno deletado com sucesso!"})
    }else{
        res.status(400).json({msg: "Aluno não encontrado!"})
    }
})

app.post('/alunos', (req, res) => {
    const aluno = req.body
```

```
    const idAluno = uuidv4()
    aluno.id = idAluno
    alunos[idAluno] = aluno
    res.json({msg: "Aluno adicionado com sucesso!"})
})

app.get('/alunos', (req, res) => {
    res.json({alunos: Object.values(alunos)})
})

app.listen(8080, () => {
    console.log('Servidor pronto na porta 8080')
})
```

Resumo

Nesta aula, você viu como criar servidores web*com a biblioteca Express. Viu como criar diferentes comportamentos para o mesmo* path *variando o método HTTP. Além disso, viu como obter parâmetros via* query string *e pelo* path* da requisição.