CAPÍTULO 4 – Express – rotas, requisições e HTML render

Neste capítulo você vai ver:

- Introdução ao express;

- Criação de um servidor HTTP com Express;

- Criação de rotas;

- Gerenciamento de requisições;

- Renderização de arquivos HTML

Atividade 1 – Express e app "Olá Mundo!"

Essa atividade tem como objetivo:

- Instalar o módulo do Express;

- Criar um servidor e desenvolver uma aplicação simples;

Comandos: app.get | res.send | app.listen

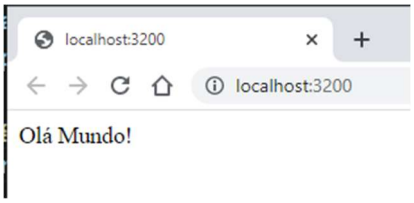
Primeira aplicação com Express

Vamos aprender a instalar e utilizar o modulo do Express criando um servidor HTTP simples.

1. Para iniciar as atividades do capítulo 04, crie um diretório na pasta "NODEJS_EXERCÍCIOS" com o nome "Cap04" e acesse o novo diretório através do comando "cd Cap04".
2. Com o diretório "Cap04" ativo emita o comando "npm i express".
3. Agora crie um módulo com o nome "at01-ExpressHTTP.js" e insira o seguinte código:

```
const express = require('express')
var app = express()//Instância do módulo Express
var porta = '3200' //Variável com o número da porta usada pelo app

//Recebe uma requisição HTTP do tipo get na rota raiz "/"
app.get('/', function (req, res){
    res.send('Olá Mundo!') //Emite a resposta para a requisição
})

//Coloca o servidor em execução para escutar requisições HTTP
app.listen(porta, function(){
    console.log(`Servidor rodando em: http://localhost:${porta}`)
})
```

4. Para realizar o teste execute o script e acesse no navegador a url http://localhost:3200/. Deverá retornar o seguinte resultado:



5. Vamos agora refatorar o código criando novas rotas, passando tags HTML como resposta e utilizando arrow functions. Para isso atualize o escript com o seguinte código:

```
const express = require('express')
var app = express()
var porta = '3200'
app.get('/', (req, res)=>{res.send('<h1>Página Inicial</h1>')})
app.get('/cadastro', (req, res)=>{res.send('<h1>Tela de cadastro</h1>')})
app.get('/usuario', (req, res)=>{res.send('<h1>Tela de usuário</h1>')})
app.get('/consulta', (req, res)=>{res.send('<h1>Tela de consulta</h1>')})
app.listen(porta, function(){console.log(`Servidor rodando em:
http://localhost:${porta}`)})
```

6. Para realizar o teste execute o script e acesse no navegador as seguintes url:

http://localhost:3200/ | http://localhost:3200/cadastro | http://localhost:3200/usuario | http://localhost:3200/consulta

Atividade 2 – Parâmetros obrigatórios e opcionais

Essa atividade tem como objetivo:

- Receber parâmetros obrigatórios e opcionais por meio da url da requisição;

Comandos: req.params

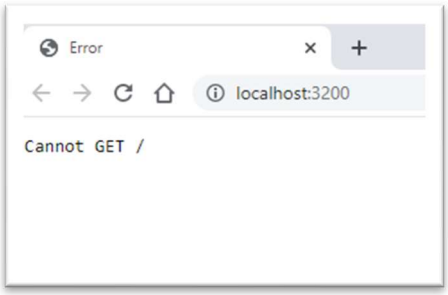
Parâmetros na url da requisição

Vamos ver como receber parâmetros em uma aplicação utilizando o Express.

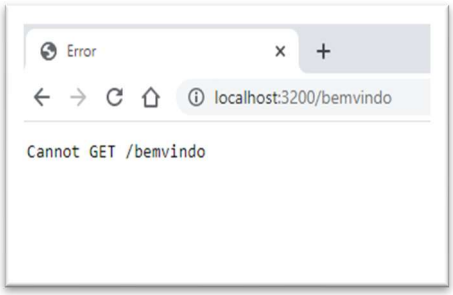
1. Para iniciar a atividade, crie um módulo no diretório Cap04 com o nome “at02-ParametroExpress.js” e insira o seguinte código:

```
const app = require('express')()
var porta = '3200'
//Parâmetros obrigatórios '/:nome'
app.get('/bemvindo/:nome', (req, res)=>{
    let nome = req.params.nome //Acesso ao parâmetro "nome"
    res.send(`<h1>Olá ${nome}, seja bem-vindo!</h1>`)
})
app.listen(porta, ()=>console.log(`Servidor rodando em:
http://localhost:${porta}`))
```

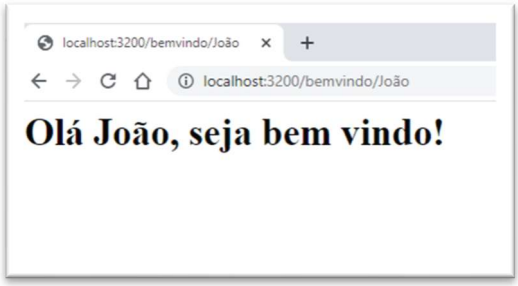
2. execute o script e realize testes no navegador conforme o seguinte:



Neste caso a rota raiz “/” não existe



Neste caso o parâmetro é obrigatório



Neste caso o parâmetro “nome” recebeu o valor “João”

3. Vamos agora refatorar o código e tornar o parâmetro opcional, para isso atualize o código conforme o seguinte:

```
const app = require('express')()
var porta = '3200'
//Parâmetros opcionais acrescentar '?', exemplo '.../:nome?'
app.get('/bemvindo/:nome?', (req, res) => {
    let nome = req.params.nome? req.params.nome : ''
    res.send(`<h1>Olá ${nome}, seja bem-vindo!</h1>`)
})
app.listen(porta, ()=>{console.log(`Servidor rodando em:
http://localhost:${porta}`)})
```

4. Refaça os testes acessando as mesmas URLs e observe os resultados.

Atividade 3 – Receber parâmetros por meio de query parameters

Essa atividade tem como objetivo:

- Aprender como receber parâmetros na URL da requisição com query parameters

Comandos: req.query

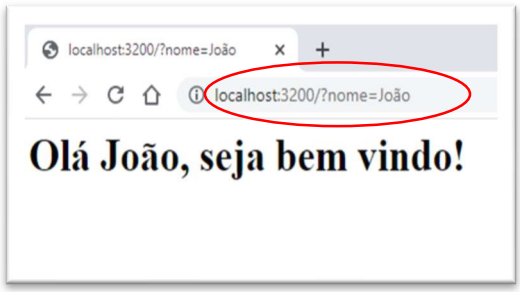
Parâmetros na url da requisição por meio de query parameters

Vamos ver como receber parâmetros na URL por meio de query parameters em uma aplicação do Express.

1. Para iniciar a atividade, crie um módulo no diretório Cap04 com o nome “at03-QueryParams.js” e insira o seguinte código:

```
var app = require('express')()
var porta = '3200'
app.get('/',(req, res)=>{
    //Acessa o Query Parameter na URL da requisição, caso exista
    if (req.query['nome']){
        let nome =req.query['nome']
        res.send(`<h1>Olá ${nome}, seja bem vindo!</h1>`)
    }
    else{
        res.send(`<h1>Informe seu nome</h1>
            <form method="GET">
              <label for="nome">Nome:</label>
              <input type="text" name="nome" placeholder="Informe seu nome" />
              <input type="submit" value="Enviar" />
            </form>`)
    }
})
app.listen(porta, ()=>console.log(`Servidor rodando em:
http://localhost:${porta}`))
```

2. Realize o teste conforme acessando o navegador conforme imagens abaixo:



3. Observe no destaque da imagem que os query parameters são definidos na URL pelo caractere “?” seguido do “nome_do_parâmetro” sinal de “=” e o “valor_do_parâmetro”. No exemplo “.../?nome=João”.

Atividade 4 – Renderizar arquivos HTML utilizando o Express

Essa atividade tem como objetivo:

- Aprender como renderizar arquivos HTML como resposta da requisição;

- Utilizar o modulo “path” para configurar o caminho do diretório templates.

Comandos: path.join | res.sendFile | app.use

Renderizar templates HTMLs em aplicações Express

Vamos ver como configurar o caminho de um diretório de templates HTML e renderizá-los em aplicações do Express. Vamos aprender também como estabelecer uma rota de erro 404 para quando o usuário digitar uma rota que não existe.

1. Para iniciar a atividade, crie um subdiretório no diretório Cap04 chamado "at04-RenderizandoHTML". Dentro do subdiretório "at04-RenderizandoHTML" crie outro subdiretório chamado "templates".
2. Copie do material do curso ou crie-os na pasta "templates" os seguintes arquivos HTML com seus respectivos códigos:

    a) index.html

```
<!DOCTYPE html>
<html lang="pt-br">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Home</title>
</head>

<body>
    <h1>Página Inicial</h1>
</body>

</html>
```

    b) cadastrar.htm

```
<!DOCTYPE html>
<html lang="pt-br">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Cadastrar</title>
</head>

<body>
    <h1>A rota cadastrar foi acessada!</h1>
</body>

</html>
```

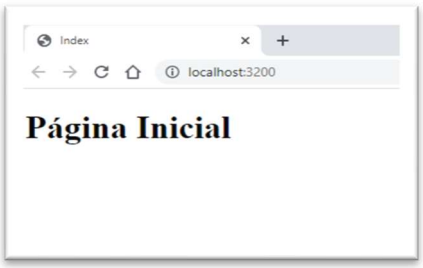
c) 404.html

```
<!DOCTYPE html>
<html lang="pt-br">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Erro 404</title>
</head>

<body>
    <h1>Página não encontrada - ERRO:404</h1>
</body>

</html>
```

3. No diretório “at04-RenderizandoHTML” crie um arquivo “index.js” e insira o seguinte código:

```
const app = require('express')()
const path = require('path')
const porta = '3200'
//Sctring com o caminho do diretório base + /templates
const baseDir = path.join(__dirname, 'templates')
app.get('/', (req, res)=>res.sendFile(`${baseDir}/index.html`))
app.get('/cadastrar', (req, res)=>res.sendFile(`${baseDir}/cadastrar.html`))
app.use((req, res)=>res.sendFile(`${baseDir}/404.html`))//Rota padrão -
Importante ficar por último
app.listen(porta, ()=>console.log(`Servidor rodando em:
http://localhost:${porta}`))
```

4. Agora execute o script index.js e teste as rotas:



Atividade 5 – Formato JSON e persistência de dados em arquivos

Essa atividade tem como objetivo:

- Familiarizar-se com respostas no formato dados JSON.

- Introdução a permanência de dados utilizando o NeDB que é um banco de dados baseado em arquivo e que trabalha com o formato JSON.

Comandos:

- npm i nedb

- app.use(express.urlencoded({extended:true}))

- app.use(express.json())

- app.post()

Respostas no formato JSON e persistência de dados em arquivos

Nesta atividade vamos ver como podemos responder as requisições com dados no formato JSON, também vamos gravar registros com rotas POST. Para persistir os dados, a título de exemplo utilizaremos o NeDB que é um banco de dados não relacional baseado em arquivos que utiliza o formato JSON.

Para iniciar duplicaremos o código exercício anterior e faremos a instalação do módulo NeDB:

1. A partir do diretório Cap04 emita o comando de instalação do NeDB – "**npm i nedb**"
2. Ainda dentro de Cap04 duplique a pasta "at04-RenderizandoHTML" e renomeie para "at05-NeDB" e depois acesse-o no console com o comando "**cd at05-NeDB**".
3. Vamos alterar o <body> dos arquivos "index.html" e "cadastrar.html" que estão na pasta "templates" com os respectivos códigos:
   a) index.html

```
<body>
    <nav>
        <ul>
            <li><a href="/cadastrar">Cadastrar</a></li>
            <li><a href="/registros">Ver Registros</a></li>
        </ul>
    </nav>
</body>
```

b) cadastrar.html

```
<body>
 <h1>Cadastrar Pets</h1>
 <form method="POST" action="/cadastrar">
  <div><label for="nome">Nome:</label>
  <input type="text" name="nome" autofocus placeholder="Nome do pet" />
  </div>
  <div><label for="especie">Especie:</label>
  <input type="text" name="especie" autofocus placeholder="Ex: cachorro" />
  </div>
  <div><label for="idade_aproximada">Idade Aproximada:</label>
   <input type="number" name="idade_aproximada" placeholder="Ex: 2.5" />
   </div>
   <div> <label for="porte">Porte:</label>
   <input type="text" name="porte" placeholder="Ex.: Pequeno" />
   </div>
   <div><label for="cor_predominante">Cor Predominante:</label>
   <input type="text" name="cor_predominante" placeholder="Ex.: cinza" />
   </div>
   <div><label for="cor_secundaria">Cor Secundaria:</label>
   <input type="text" name="cor_secundaria" placeholder="Ex.: preto" />
   </div>
   <div><input type="submit" value="Enviar" /></div>
 </form>
</body>
```

4. O próximo passo é criar um arquivo de conexão ao banco de dados. Para isso crie um modulo no diretório do exercício com o nome “db.js” e insira o seguinte código:

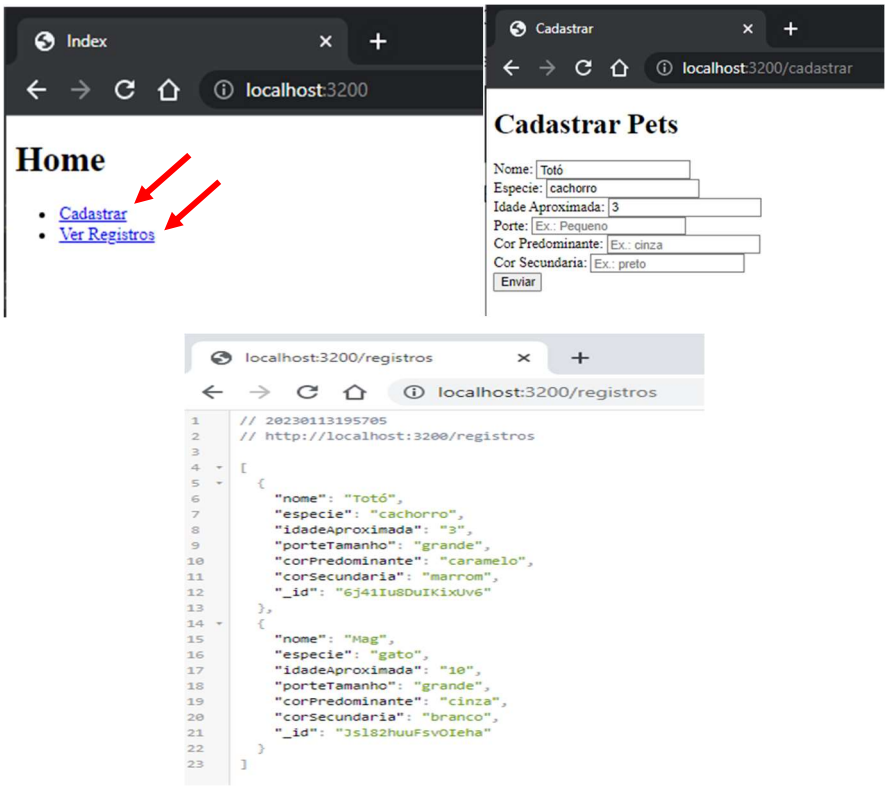
```
const NeDB = require('nedb')
module.exports = new NeDB({ //Cria uma instância da classe NeDB
    filename: 'pets.db', //Nome do arquivo do banco
    autoload:true //Biblioteca gerencia o arquivo de forma autônoma
})
```

5. Por último, vamos criar o script principal no arquivo “index.js”, que irá conter as rotas GET e POST da aplicação. Atualize o código do “index.js” da seguinte forma:

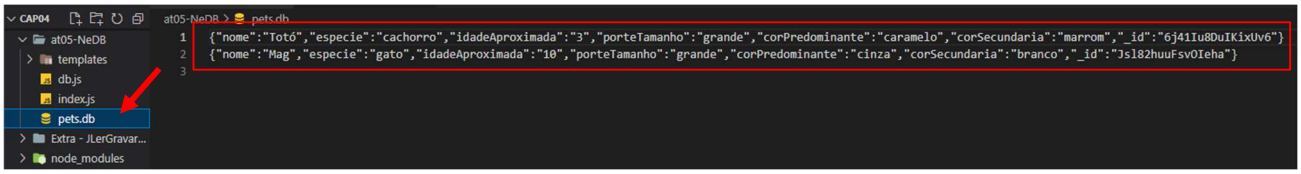
```
const express = require('express')
const app = express()
var db = require('./db')
const path = require('path')
```

```
const baseDir=path.join(__dirname, 'templates')
var porta = '3200'
app.use(express.urlencoded({extended:true}))
app.use(express.json())
app.get('/', (req, res)=>res.sendFile(`${baseDir}/index.html`))
app.get('/cadastrar', (req, res)=>res.sendFile(`${baseDir}/cadastrar.html`))

//rota do tipo GET para obter os dados
app.get('/registros', (req, res)=>{
    let dados;
    db.find({}).exec((err, dados)=>{ //ler dados
        if (err) {
            res
                .status(400) //Resposta com status de erro 400
                .json(err) //Resposta no formato JSON
        } else {
            res
                .status(200) //Resposta com status de sucesso 200
                .json(dados) //Resposta no formato JSON
        }
    })
})

//rota do tipo POST - recebe dados no corpo da requisição
app.post('/cadastrar', (req, res)=>{
    //Salvar dados
    db.insert(req.body, (err, dados)=>{
        if (err){
            res
                .status(400)
                .json(err)
        } else{
            res
                .status(200)
                .json(dados)
        }
    })
})
app.listen(porta, ()=>console.log(`Servidor rodando em:
http://localhost:${porta}`))
```

6. Execute o módulo index.js e realize os testes. Acesse a rota raiz “/” no navegador e clique no menu “Cadastrar” e realize alguns cadastros. Em seguida clique no menu “Ver Registros” e veja todos os registros listados no formato JSON.



7. Perceba que após a execução do script um arquivo chamado "pets.db" foi criado automaticamente. Após a realização dos cadastros ele ficou com o seguinte conteúdo:



Exercícios extras

Estas atividades extras têm como objetivo expandir o conhecimento e a prática do express:

1. Acesse a documentação do **Express:** https://expressjs.com/pt-br/ e pesquisar para que serve, como instalar e como implementar uma aplicação simples.
2. Reescreva o exercício da **Atividade 1 do Capítulo 3 - Ler inserir e gravar em arquivos JSON** utilizando o Express, renderizando páginas HTML e recebendo requisições GET e POST.