Desenvolvimento Backend

Aula 10 - View Engines

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula veremos como fazer para o *nodejs* servir páginas HTML e como criar essas páginas dinamicamente.

Objetivos

- Conhecer como integrar *view engines* para inserir dados nos *templates*;
- Conhecer como operar sobre os dados na renderização da página;
- Conhecer como criar *templates* fixos.

EJS como view engine

Link do video da aula: https://youtu.be/jd1yI5yu7cA

Primeiramente, será usada uma biblioteca chamada *EJS* que facilitará a criação dinâmica do HTML.

Instalando o *EJS*

Essa biblioteca permite o acesso a variáveis do *nodejs* para construir o HTML e criá-lo dinamicamente.

Para fazer o uso dessa biblioteca, deve antes fazer a instalação. Então, execute no terminal o comando:

```
    npm install ejs
```

Utilizando o *EJS*

Após instalar a biblioteca, é preciso informar ao *express* que será usado esse *template engine*. A configuração será feita no arquivo *index.js*. Então, nas linhas que sucedem à configuração geral do *Express*, adicione a linha abaixo:

```
    app.set('view engine', 'ejs')
```

Daí em diante o *EJS* já pode ser utilizado e, para isso, crie na raiz do projeto uma pasta de *views* e dentro dela ficarão os arquivos HTML com extensão *.ejs*.

Dentro dessa página, crie um arquivo *index.ejs* que deve conter o código abaixo:

```
<!DOCTYPE html>
```

```
<html lang="en">
<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>Simples exemplos de EJS</title>
</head>
<body>
    <p>Exemplo de EJS!!</p>
    <p><%=variavel%></p>
</body>
</html>
```

Agora, sempre que o usuário acessar algum endereço, essa página deverá ser retornada.

Renderizando o HTML

Para a página ser exibida ao usuário, uma nova rota deve ser criada para que, ao ser acessada, ao invés de retornar *json*, retorne a página *HTML*. Então, no arquivo *index.js* adicione a seguinte rota:

```
    app.get('/home', (req, res) => {
            const number = Math.random()
            res.render('pages/index', {variavel: number})
    })
```

Operando sobre os dados

Link do video da aula: https://youtu.be/owrUxK4riqs

Continuando com o uso do EJS, veremos mais algumas funcionalidades importantes dessa biblioteca.

Manipulando listas

É comum que a aplicação precise manipular dados em *arrays* para criar a página. Para ver como fazer isso, crie no arquivo *index.js* uma rota passando um *array* de cursos. A rota deve ser implementada de acordo com o modelo abaixo:

```
    app.get('/cursos', (req, res) => {
```

```
            const cursos = [
                    {nome: "Programação fronend", ch: 280},
                    {nome: "Programação backend", ch: 330},
                    {nome: "Programação concorrente", ch: 300},
                    {nome: "Programação distribuída", ch: 400}
            ]
            res.render('pages/cursos/index', {cursos: cursos})
    })
```

Após isso, acompanhe a aula para criar e implementar um arquivo *index.ejs* contendo o HTML que renderizará as informações passadas na rota anteriormente criada. O código, ao final, deve ficar conforme o código abaixo:

```
    <!DOCTYPE html>
    <html lang="en">
    <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>Cursos</title>
    </head>
    <body>
        <ul>
            <%  cursos.forEach( function (curso) { %>

                    <% if (curso.ch > 300) { %>
                            <li><%=curso.nome%> (<%=curso.ch%>)</li>
                    <% } else { %>
                            <li><%=curso.nome%>
(<b><%=curso.ch%></b>)</li>
            <% }})  %>
        </ul>
    </body>
```

Adicionando estilo à página

Link do video da aula: https://youtu.be/ylr8EArXHeA

O aspecto visual de uma página é extremamente importante, é ele que faz o usuário ser atraído e entretido.

A aplicação em desenvolvimento ainda se encontra muito limpa e para mudar isso será adicionada uma estilização.

Aplicando estilo na aplicação

Para aplicar o estilo será criada uma *tag*, chamada *style*, no arquivo *index.ejs*, que contém o HTML de cursos. Acompanhe a aula e ao final o arquivo deve ficar de acordo com o código abaixo:

```
<!DOCTYPE html>
<html lang="en">
<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>Cursos</title>
    <style>
        body {
            background-color: aquamarine;
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica
Neue', sans-serif;
        }
        h1 {
            background-color:cornflowerblue;
            color: white;
            font-weight: 400;
            padding: 15px;
        }
    </style>
</head>
 <body>
        <ul>
            <%  cursos.forEach( function (curso) { %>

                    <% if (curso.ch > 300) { %>
                            <li><%=curso.nome%> (<%=curso.ch%>)</li>
                    <% } else { %>
                            <li><%=curso.nome%>
(<b><%=curso.ch%></b>)</li>
            <% }})  %>
        </ul>
    </body>
```

```
</html>
```

Criando a página de alunos

Agora será implementada uma nova rota. Então, acompanhe a aula e no arquivo *index.js* adicione o código abaixo:

```
    app.get('/alunos', (req, res) => {
            const alunos = [
                    {nome: "João Pedro"},
                    {nome: "Fernanda"},
                    {nome: "Francisco"}
            ]
            res.render('pages/alunos/index', {alunos: alunos})
    })
```

Após isso, crie um *index.ejs* e implemente o HTML de listagem de alunos como o código abaixo:

```
!DOCTYPE html>
<html lang="en">
<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>Alunos</title>
    <style>
        body {
            background-color: aquamarine;
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica
Neue', sans-serif;
        }
        h1 {
            background-color:cornflowerblue;
            color: white;
            font-weight: 400;
            padding: 15px;
        }
    </style>
</head>
<body>
```

```
        <h1>Alunos</h1>
        <ul>
                <%  alunos.forEach( function (aluno) { %>
                        <li><%=aluno.nome%></li>
                <% }) %>
        </ul>
</body>
</html>
```

Criando Layouts com express-ejs-layouts

Link do video da aula: https://youtu.be/cQ6l_sD1kMo

Adicionar a *tag style* resolve os problemas temporariamente. Porém, pensando no futuro, existirão telas que possuem a mesma estilização e não seria vantajoso criar uma *tag style* para cada um dos arquivos e, ao ter de fazer uma alteração, sair alterando em todos eles. Então, nesta aula, veremos como criar um estilo fixo para as páginas e assim, se preocupar em alterar apenas um único arquivo.

Utilizando o express-ejs-layout

Para lidar com essas questões, usaremos a ferramenta *express-ejs-layout*.

Instale executando o seguinte comando no terminal:

```
    npm install express-ejs-layouts
```

Após instalado, acompanhe a aula e modifique o arquivo *index.js* para configurar a aplicação e fazer uso dessa ferramenta. As configurações feitas devem ficar como se vê abaixo:

```
    const express = require('express')
    const rotaUsuario = require('./rotas/usuario.rota')
    const rotaPost = require('./rotas/posts.rota')
    var expressLayouts = require('express-ejs-layouts')

    const app = express()

    app.use(express.json())
    app.set('view engine', 'ejs')
```

```
    app.set('layout', 'layouts/layout')

    app.use(expressLayouts)

    app.use('/static', express.static('public'))

    app.use('/usuarios', rotaUsuario)
    app.use('/posts', rotaPost)
```

Criando o layout

Na pasta *views*, crie uma pasta *layout* e dentro um arquivo chamado *layout.ejs*. Após isso, coloque nesse arquivo todo o código HTML presente na página, removendo apenas o conteúdo do *body*, que será substituído por uma variável. O código ficará conforme se vê abaixo:

```
<!DOCTYPE html>
<html lang="en">
<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>IMD</title>
    <style>
        body {
            background-color: aquamarine;
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans', 'Helvetica
Neue', sans-serif;
        }
        h1 {
            background-color:cornflowerblue;
            color: white;
            font-weight: 400;
            padding: 15px;
        }
    </style>
</head>
<body>
    <%- body %>
</body>
</html>
```

Atualizando o conteúdo da página

Após criado o layout, ele ficará fixo e a única coisa que mudará será o conteúdo da *tag body*, que já foi substituído por uma variável, a qual será, durante a execução, substituída pelo corpo da página.

Então, acompanhe a aula e altere os arquivos das páginas retirando todo o *HTML* e adicionando uma nova instrução no início para que todo código após ela seja exibido no corpo da página. Os arquivos, ao final, devem ficar dessa forma:

alunos/index.js:

```
    <%- contentFor('body') %>
    <h1>Alunos</h1>
    <ul>
            <%  alunos.forEach( function (aluno) { %>
                    <li><%=aluno.nome%></li>
            <% }) %>
    </ul>
```

cursos/index.js

```
    <%- contentFor('body') %>
    <h1>Cursos</h1>
            <ul>
            <%  cursos.forEach( function (curso) { %>

                    <% if (curso.ch > 300) { %>
                            <li><%=curso.nome%> (<%=curso.ch%>)</li>
                    <% } else { %>
                            <li><%=curso.nome%>
(<b><%=curso.ch%></b>)</li>
            <% }})  %>
    </ul>
```

Resumo

Nesta aula, foi visto como servir páginas HTML e como criar essas páginas dinamicamente. Além disso, foi abordado o uso de *layouts* fixos para manter um padrão na página com maior desempenho.