Neste artigo vamos aprender a utilizar a Template Engine EJS com Node.js e também alguns recursos interessantes desta ferramenta.

Wesley Gado · há 2 anos 4 meses

Node.js.

Assim como vimos no artigo sobre PUG, com a template engine nós podemos criar as páginas das nossas aplicações em Node.js de forma dinâmica sem depender das limitações do HTML.

A grande diferença entre o EJS e o PUG é que o EJS segue uma sintaxe muito semelhante ao HTML, desta forma qualquer desenvolvedor que já conhece HTML não terá nenhuma dificuldade de trabalhar com o EJS, ao contrário do PUG que possuí algumas particularidades e que pode, no início, afetar a produtividade do desenvolvedor.

Vamos criar uma aplicação simples utilizando o EJS para entender como podemos utiliza-lo na prática.

Ambiente Node.js e Express

Primeiramente precisamos configurar nosso ambiente Node.js, para instalar o Node.js você pode seguir o artigo Instalação do node.js no windows, mac e linux, logo em seguida é necessário instalar e criar um servidor utilizando o Express.

Feito estes passos podemos finalmente utilizar o EJS para renderizar as nossas páginas utilizando várias funções interessantes.

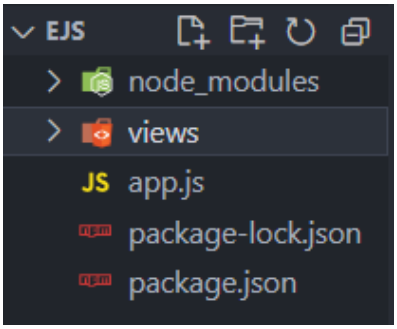
Instalando o template engine EJS

Com o ambiente Node.js instalado, vamos instalar o EJS executando o seguinte comando no terminal:

Copiar

```
npm install ejs
```

Agora vamos criar a estrutura da nossa aplicação da seguinte maneira:



Onde:

- node_modules: é a pasta criada automaticamente pelo npm com os pacotes que vamos utilizar.
- views: é a pasta que devemos criar para guardar os arquivos das páginas, no caso utilizando EJS.
- app.js: nosso arquivo base onde criamos nosso servidor e onde vamos criar a rota para apontar as views que deverão ser renderizadas quando houver uma requisição. Aqui vale a observação de que o ideal é cada arquivo ter sua responsabilidade bem definida, vamos utilizar somente o arquivo app.js neste exemplo para fins didáticos e facilitar o entendimento.
- package-lock.json e package.json: criados quando iniciamos a aplicação, vale ressaltar que estamos utilizando o padrão do ES6, portanto, é necessário adicionar a linha `'type': 'module'` ao arquivo package.json.

Ótimo, já estamos com tudo o que precisamos devidamente configurado, agora precisamos apontar para o Express que vamos utilizar o template engine EJS. Podemos fazer isso da seguinte maneira:

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

Revisando o código do arquivo `app.js`: importamos o `express`, criamos a constante `app`, com isso o Express permite a utilização do método `app.set()`.

Nele primeiramente passamos qual template engine vamos utilizar, como primeiro parâmetro passamos o termo `view engine`, e o segundo parâmetro `ejs` (poderia ser o PUG, caso fosse utilizar outro template engine). Desta forma o Express entende que queremos usar o EJS.

Logo após apontar a template engine que será utilizada, é necessário apontar também onde estarão os arquivos de interface, para isso vamos adicionar outro `app.set()`, passando o termo `views` como primeiro parâmetro e o diretório como segundo parâmetro, no do exemplo, `./views`.

Portanto, agora podemos criar uma rota get e nela vamos passar qual view será retornada quando houver essa requisição, o arquivo `app.js` ficará dessa forma:

Copiar

```
import express from 'express';

const app = express();
const port = 3000;

app.set('view engine', 'ejs');
app.set('views', './views');

app.get('/', (req, res) => {
    res.render('home');
});

app.listen(port,() => {
```

Copiar

```
<h1>Página sendo exibida utilizando EJS</h1>
```

Ao acessar a rota localhost:3000, vamos obter o seguinte resultado:

Desta forma podemos deixar a aplicação mais dinâmica, como exibir variáveis nas páginas e também utilizar recursos de reutilização de código, por exemplo.

Recursos básicos do template engine EJS

Vamos ver alguns recursos básicos que podemos utilizar com o EJS.

JavaScript e utilização de base de dados nas views

Um recurso interessante é caso seja necessário exibir alguma informação dinâmica no HTML.

Neste exemplo vamos criar um array de pessoas para simular uma base de dados, mas ela pode vir de um Banco de Dados, sem nenhum problema.

Dito isto, ao alterar o arquivo app.js, ele ficará da seguinte forma:

Copiar

```
var pessoas = [
    {
        'nome': 'Paulo',
        'idade': 12
    },
    {
        'nome': 'Jõao',
        'idade': 15,
    },
    {
        'nome': 'Marina',
        'idade': 25,
    },
]

app.get('/', (req, res) => {
    res.render('home', {pessoas: pessoas});
});

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

Repare que criamos o array `pessoas` com 3 itens, e quando apontamos para carregar o arquivo home de nossa view passamos outro parâmetro: `{pessoas: pessoas}`.

Aqui nós estamos recebendo o array pessoas e apontando que vamos utilizar este array em nossa view como... pessoas (veja só). Exemplo, se a declaração fosse `{nomes: pessoas}`, o array `pessoas` seria usado como `nomes` em nosso arquivo home.

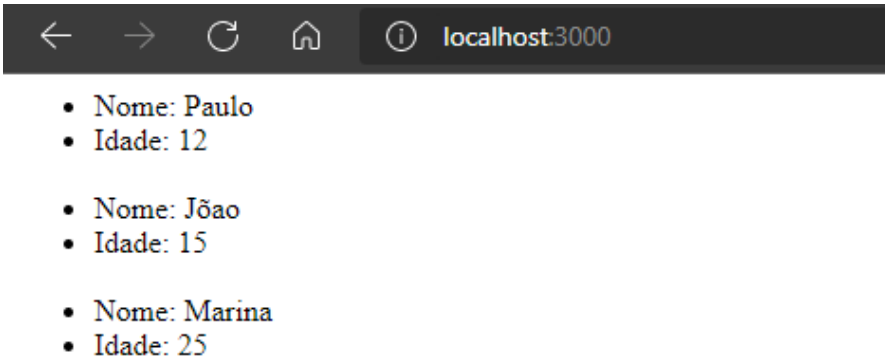
Configuramos o arquivo `app.js`, agora vamos criar uma lista dentro da view `home.ejs`:

Copiar

```
<ul>
    <% for(var i=0; i<pessoas.length; i++) {%>
        <li>
            Nome: <%= pessoas[i].nome %>
        </li>
        <li>
```

`for`) e (ao exibir o item do array `pessoas`).

Ao acessar a nossa rota principal, vamos então obter uma lista do array declarado no arquivo app.js:

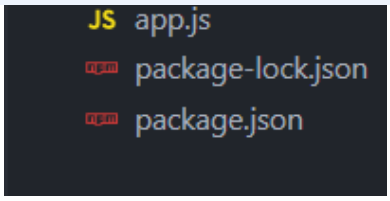


Dessa forma é possível exibir, por exemplo, os dados com origem de algum banco em nossa interface.

Reutilização de código com EJS

Imagina que você está criando uma aplicação web que possuí um menu e um footer em todas as páginas. Criar um menu e um footer para cada página, além de mais trabalhoso, pode também dificultar a manutenção da aplicação, pois qualquer alteração será necessária repeti-la em todas as páginas e ainda pode aumentar o risco de erros.

Para isso, podemos utilizar o recurso `includes` do EJS. Vamos então criar mais três views, uma com o nome de `index.ejs`, outra com o nome `header.ejs` e a última com o nome de `footer.ejs`:



Neste caso vamos criar um menu no `header.js`, um footer no `footer.ejs`, vamos manter o conteúdo já criado na `home.js` e a `index.ejs` será para exibir o conteúdo de cada arquivo. Cada arquivo ficará da seguinte maneira:

- **header.ejs**

Copiar

```
<html lang="pt">

<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>Includes EJS</title>
</head>
    <style>
        .menu-item{
            display: inline;
            margin: 0 10px 0 10px;
        }
    </style>

<body>
    <nav>
        <ul>
            <li class="menu-item">MENU 1</li>
            <li class="menu-item">MENU 2</li>
            <li class="menu-item">MENU 3</li>
        </ul>
    </nav>
```

- **home.ejs**

```
            <br>
            <br>
        </li>
        <% } %>
</ul>
```

- **footer.ejs**

Copiar

```
<footer>
    <nav>
        <ul>
            <li class="menu-item">TreinaWeb</li>
            <li class="menu-item">Artigo EJS </li>
            <li class="menu-item">Autor: Wesley Gado</li>
        </ul>
    </nav>
</footer>
</body>

</html>
```

Agora, o `index.ejs` só irá incorporar as views criadas anteriormente:

Copiar

```
<%- include('header'); %>
<%- include('home'); %>
<%- include('footer'); %>
```

Vamos alterar a rota criada no arquivo `app.js` para: `res.render('index', {pessoas: pessoas});`. Ao acessar a rota principal vamos obter o seguinte resultado:

- Nome: Marina
- Idade: 25

TreinaWeb Artigo EJS Autor: Wesley Gado

Perceba que se alterarmos o arquivo `home.ejs`, somente o conteúdo será modificado, mantendo o header e footer:

Conclusão

Neste artigo aprendemos a utilizar o template engine EJS, que diferente do PUG, possui uma sintaxe similar ao HTML, facilitando a utilização de desenvolvedores que possuem conhecimento prévio nessa tecnologia. Também aprendemos a utilizar recursos básicos como utilizar JavaScript em nossas views, utilizar base de dados dinâmicas e o recurso include, usado com o intuito de reaproveitar códigos repetitivos. Com este artigo base você será capaz de criar suas primeiras views com o Template Engine EJS :D

- Conhecendo a estrutura;
- Entendendo os objetos Request e Response;
- Servindo arquivos estáticos;
- Roteamento;
- Conexão com banco de dados;
- Template Engine;
- Express Generator.

#JavaScript #Template Engine #Node.js

Autor(a) do artigo

Wesley Gado

Formado em Análise e Desenvolvimento de Sistemas pelo Instituto Federal de São Paulo, atuou em projetos como desenvolvedor Front-End. Nas horas vagas grava Podcast e arrisca uns três acordes no violão.

Todos os artigos

Artigos relacionados

Ver todos →

que é Template Engine e as principais opções do mercado.

templates com os fragments parametrizados da temp...

utilizando Express com o Template Engine PUG e entender as...

Desenvolvimento Back-end

NestJS com template engine handlebars

Neste artigo vamos aprender a configurar o NestJS com o handlebars, uma template engine que permite...

C#

Trabalhando com a template engine Liquid no ASP.NET Core

Em alguns projetos pode ser necessário substituir o Razor por outra engine template. Neste caso, apr...

Java

Thymeleaf - Reaproveitando código com fragments

Neste artigo veremos como utilizar a funcionalidade de fragments da template engine Thymeleaf para r...

Neste artigo veremos o que é e os principais recursos de uma das template engines mais utilizadas no...

veremos como criar o primeiro projeto no Symfony. Falaremos sobre qual tipo de estrutur...

Neste artigo vamos aprender o passo a passo de como realizar o upload de arquivos para a AWS S3 util...

Desenvolvimento Back-end

Como realizar upload de arquivos locais no NestJS

Neste artigo vamos aprender a utilizar o recurso de upload de arquivos locais com o NestJS utilizand...

Desenvolvimento Back-end

Autenticação: Protegendo Rotas com NestJS e Passport

Neste artigo vamos aprender a proteger rotas com o NestJS e a biblioteca passport, utilizando a estr...

Javascript

Criando Rotas com Express.js

Vamos aprender a configurar rotas com o Express.js, com exemplos utilizando as requisições GET, POST...