Utilizando a engine EJS para aplicações em NodeJS

por  Igor Souza Martins no blog Blog da Consultoria Técnica - TecnoSpeed
Sex - 25/11/2016, 16:16

O EJS é uma engine de visualização, com ele conseguimos de uma maneira fácil e simples transportar dados do back-end para o front-end, basicamente conseguimos utilizar códigos em javascript no html de nossas páginas.

O que vamos construir?

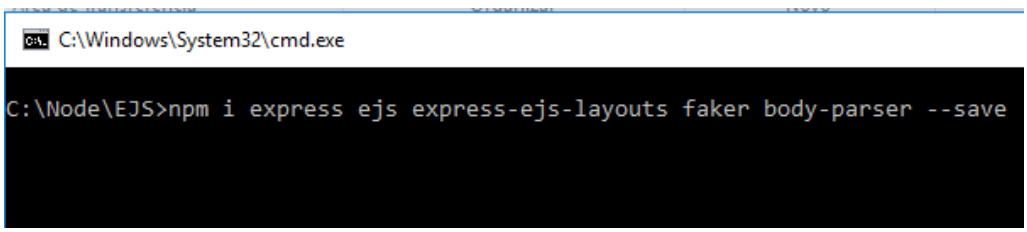
Vamos criar uma pequena aplicação que utilizará o EJS para transportarmos dados para o nosso front-end e também vamos ver uma maneira de reaproveitar algumas partes do nosso html .

Para isso vamos precisar da seguinte estrutura de arquivos:

```
  - public/
---- css/
------------ styles.css
- views/
---- layout.ejs
---- pages/
------------ about.ejs
------------ contact.ejs
------------ home.ejs
- package.json
- server.js
```

Mão na massa!

Instalando os pacotes do npm.
No seu console digite " npm i ou install express ejs express-ejs-layouts faker body-parser ndemon --save "
Dessa forma o npm vai baixar os pacotes e já inserir eles nas dependências do nosso projeto, dentro do `package.json`



express — É o pacote mais simples para criarmos as rotas do nosso app.

ejs — É o pacote responsável pela engine EJS.

express-ejs-layouts Usamos ele para conseguirmos enviar dados para nossas páginas ejs pelo express.

faker Usamos ele para gerar algumas informações aleatórias como Nome, email, imagens. (Útil para testes)

nodemon Pacote usado para subir sua aplicação com a vantagem de que a cada vez que alterar ou criar um arquivo js ele reinicia automaticamente.

Após a instalação dos pacotes seu arquivo package.json ficará da seguinte forma:

```
  {
  "dependencies": {
    "body-parser": "^1.15.2",
    "ejs": "^2.5.2",
    "express": "^4.14.0",
    "express-ejs-layouts": "^2.2.0",
    "faker": "^3.1.0"
  }
}
```

Variáveis que vamos utilizar.

No inicio do seu arquivo `server.js` adicione:

```
var express = require('express')
var faker = require('faker')
var bodyParser = require('body-parser')
var expressLayouts = require('express-ejs-layouts')
var app = express()
var port = 3000
```

Configurando o nosso `server` .

Logo após adicionarmos as variáveis, vamos configura-lás.

```
app.set('view engine', 'ejs')    // Setamos que nossa engine será o ejs
app.use(expressLayouts)          // Definimos que vamos utilizar o express-ejs-layouts na nossa aplicação
app.use(bodyParser.urlencoded()) // Com essa configuração, vamos conseguir parsear o corpo das requisições
```

Criando as rotas.

Vamos utilizar as seguintes rotas

- GET /

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

- Aqui o `server` recebeu uma requisição do `client` e devolveu o arquivo `home.ejs`, com isso o EJS renderiza ele para o `client` em forma de uma página html.

- GET /about

```
app.get('/about', (req, res) => {
  var users = [{
    name: faker.name.findName(),
    email: faker.internet.email(),
    avatar: 'http://placekitten.com/300/300'
  }, {
    name: faker.name.findName(),
    email: faker.internet.email(),
    avatar: 'http://placekitten.com/400/300'
  }, {
    name: faker.name.findName(),
    email: faker.internet.email(),
    avatar: 'http://placekitten.com/500/300'
  }]

  res.render('pages/about', { usuarios: users })
})
```

O que o `server` faz aqui é praticamente a mesma coisa que foi feita na rota `home`, porém aqui nós devolvemos para o `client` além da página `about`, uma variável chamada usuarios.

- GET /contact

```
app.get('/contact', (req, res) => {
  res.render('pages/contact')
})
```

- POST /contact

```
app.post('/contact', (req, res) => {
  res.send('Obrigado por entrar em contato conosco, ' + req.body.name + '! Responderemos em breve!')
})
```

O que o `server` faz aqui é recepcionar o conteúdo da requisição e devolver uma mensagem para o `client` aproveitando o nome da pessoa que enviou a mensagem para nós. Essa mensagem que vem no corpo da requisição, só é possível ser manipulada utilizando o pacote body-parser.

No final o nosso arquivo `server.js` ficará assim:

```
var express = require('express')
var faker = require('faker')
var bodyParser = require('body-parser')
var expressLayouts = require('express-ejs-layouts')
var app = express()
var port = 3000

// Definimos que vamos utilizar o ejs
app.set('view engine', 'ejs')
app.use(expressLayouts)
app.use(bodyParser.urlencoded())

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

app.get('/about', (req, res) => {
  var users = [{
```

```
    name: faker.name.findName(),
    email: faker.internet.email(),
    avatar: 'http://placekitten.com/300/300'
  }, {
    name: faker.name.findName(),
    email: faker.internet.email(),
    avatar: 'http://placekitten.com/400/300'
  }, {
    name: faker.name.findName(),
    email: faker.internet.email(),
    avatar: 'http://placekitten.com/500/300'
  }]

  res.render('pages/about', { usuarios: users })
})

app.get('/contact', (req, res) => {
  res.render('pages/contact')
})

app.post('/contact', (req, res) => {
  res.send('Obrigado por entrar em contato conosco, ' + req.body.name + '! Responderemos em breve!')
})

app.use(express.static(__dirname + '/public'))
app.listen(port)
console.log('Servidor iniciado em http://localhost:' + port)
```

Arquivos ejs

Agora vamos ver como vão ficar nossas páginas e também entender como trabalhar com elas.

- Layout

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Utilizando o EJS</title>
  <link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="/css/styles.css">
</head>
<body>
  <header>
     <nav class="navbar navbar-inverse">
      <ul class="nav navbar-nav">
        <li><a href="/">Home</a></li>
        <li><a href="/about">About</a></li>
        <li><a href="/contact">Contact</a></li>
      </ul>
     </nav>
  </header>

   <main>
     <div class="container">
      <%- body %>
     </div>
   </main>

  <footer>
    EJS
  </footer>
</body>
</html>
```

Aparentemente é uma página em html apenas, porém quando iniciamos nosso aplicativo e requisitamos alguma rota que devolve o arquivo `ejs` , o express-ejs-layouts procura o arquivo`layout.ejs` dentro da pasta **views** , então onde está o código <%- body %> será carregado o conteúdo de outro arquivo ejs. No próximo passo você vai entender como funciona essa ligação entre os arquivos.

- Home

```
<%- contentFor('body') %>
<div class="jumbotron text-center">
    Página Inicial
</div>
```

No arquivo `layout.ejs` vimos que o código **<%- body %>** seria substituído por outro assim que a página fosse renderizada, agora no arquivo `home.ejs` temos **<%- contentFor('body') %>** , é aqui que ocorre a ligação entre os dois arquivos, no lugar de **<%- body %>** ficará essa div.

- About

```
<%- contentFor('body') %>
  <div class="jumbotron text-center">
     <h1>O Time!</h1>
     <div class="row">
        <% for (usuario of usuarios){%>
          <div class="col-sm-4">
                <h2><%= usuario.name %>
                <img class="img-responsive" style="height: 250px" src="<%= usuario.avatar %>">
          </div>
        <% } %>
     </div>
  </div>
```

O que fazemos aqui é renderizar para o `client` todos os nossos usuários, isso é possível pois fizemos um loop com o `ejs` , para capturarmos o valor de cada usuário que existe na variável **usuarios** que a rota `/about` devolveu.

- Contact

```
<div class="jumbotron text-center">
      <div class="row">
            <div class="col-sm-6 col-sm-offset-3">
                  <h2>Entre em contato!</h2>
                  <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
                        tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
                        quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
                        consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
                        cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
                  proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
                  <form action="/contact" method="POST">
                        <div class="form-group">
                              <label>Nome</label>
                              <input type="text" name="name" class="form-control">
                        </div>
                        <div class="form-group">
                              <label>Email</label>
                              <input type="text" name="email" class="form-control">
                        </div>
                        <div class="form-group">
                              <label>Sua mensagem</label>
                              <input type="text" name="message" class="form-control">
                        </div>
                        <div class="form-group text-right">
                              <button type="submit" class="btn btn-primary btn-lg">Enviar</button>
                        </div>
                  </form>
            </div>
      </div>
</div>
```

Nessa página usamos apenas o html para enviar um `POST` para a rota `/contact` .

Conclusão

Este mini tutorial tem a intenção de mostrar para vocês o básico do que o EJS pode fazer, agora que você já sabe trabalhar com a comunicação entre o back-end e o front-end, por que você não tenta mesclar este tutorial e nosso tutorial anterior Autenticando uma API em NodeJS em um projeto seu?

O projeto completo está em anexo no post, e você também pode clonar ele para sua maquina pelo `git clone https://github.com/nulldreams/EJS.git`

Caso queira algum tutorial especifico nos deixe um comentário, valeu pessoal!