Node.js

Utilizando variáveis de ambiente em aplicações Node.js com a lib dotenv

Utilize variáveis de ambiente para parametrizar sua aplicação utilizando o pacote dotenv.

Marcelo Ribas Vismari
9 de Out de 2023 • 5 min read

Parametrizando sua aplicação com dotenv

Ao criar uma aplicação, é comum ter valores que podem variar ao longo do código dependendo do ambiente em que a aplicação está sendo executada. Por exemplo, a string de conexão com o banco de dados pode ser diferente dependendo do ambiente onde o código está sendo executado, porém o código fonte em si é exatamente o mesmo independente do ambiente.

Nesse contexto torna-se vantajoso parametrizar estes valores por meio do uso de variáveis de ambiente.

Para ler o valor de uma variável de ambiente, podemos utilizar a propriedade `env` do módulo `process`.

```
1 | console.log(process.env.MINHA_VARIAVEL);
```

hello.js

Já para configurar a chave e valor da variável de ambiente, podemos por exemplo, utilizar o comando `export` no bash antes de executar a aplicação:

```
1 | $ export MINHA_VARIAVEL=olá
2 | $ node hello.js
3 |
4 | olá
```

Embora seja uma abordagem simples, na prática, as aplicações do mundo real frequentemente lidam com dezenas de variáveis. Em vez de recorrer repetidamente ao comando `export`, podemos aprimorar a organização dessas variáveis através do uso de um arquivo chamado `.env`. Nesse arquivo, consolidamos uma série de pares "chave=valor", conforme ilustrado no exemplo abaixo:

```
minha_variavel=123
porta=3000
string_conexao_sas=456
```

Exemplo de um arquivo .envc

A seguir, é necessário um trecho de código para efetuar a leitura desse arquivo e carregar os valores no `process.env`. Podemos alcançar esse

objetivo através do seguinte código:

```
const { readFile } = require("fs");

function pegarConteudoArquivo(arquivo) {
  return new Promise((resolve, reject) => {
    readFile(arquivo, (error, buffer) => {
      if (error) {
        reject(error);
        return;
      }

      const conteudoArquivo = buffer.toString("utf-8");
      resolve(conteudoArquivo);
    });
  });
}

(async function () {
  let conteudoArquivoEnv;

  try {
    // Contéudo do arquivo .env:
    //   minha_variavel=123
    //   porta=3000
    //   string_conexao_sas=456
    conteudoArquivoEnv = await pegarConteudoArquivo(".env");
  } catch (error) {
    console.error("Erro ao ler arquivo .env", error);
    process.exit(-1);
  }

  const linhas = conteudoArquivoEnv.split("\n");

  // Para cada linha do arquivo .env:
  for (const linha of linhas) {
    // Separa a chave e valor:
    const [chave, valor] = linha.split("=");

    // Preenche o process.env.<chave> = <valor>
    process.env[chave] = valor;
  }

```

```
  // Conferindo se os valores do arquivo .env estão
  // disponíveis no process.env.<chave>:
  console.log(process.env.minha_variavel);
  console.log(process.env.porta);
  console.log(process.env.string_conexao_sas);
})();

// Resultado da execução:
// $ node arquivo.js
//   123
//   3000
//   456
```

Embora a lógica seja simples (código acima), não precisamos "reinventar a roda". Para efetuar a leitura do arquivo `.env` e popular o `process.env`, podemos recorrer a uma biblioteca amplamente conhecida: dotenv. Ela é uma biblioteca muito útil em aplicações Node.js, e sua principal finalidade é carregar variáveis de ambiente de um arquivo chamado `.env` para o ambiente Node.js. De forma alternativa, também podemos realizar esta tarefa nativamente utilizando Node.js a partir da versão `20.6`.

Starting from Node.js v20.6.0, Node.js supports `.env` files for configuring environment variables.

https://nodejs.org/en/blog/release/v20.6.0

Agora, exploraremos o uso do pacote dotenv e discutiremos como realizar o mesmo procedimento utilizando a versão mais recente do Node.js de forma nativa.

Utilizando o pacote dotenv

O uso da `dotenv` é bem simples. Para começar, vamos criar um novo projeto e instalar a dependência:

```
# Versão do Node.js
node --version
v18.18.0

# Iniciando o projeto NPM
npm init -y

# Instalando a lib dotenv
npm i dotenv
```

Abrindo o VS Code criamos o arquivo `index.js` e o arquivo `.env`:

```
// Para iniciar a leitura do arquivo `.env`
// basta adicionar a seguinte linha:
//
// Se estiver utilizando ES6, a síntaxe será:
// import 'dotenv/config'
require("dotenv").config();

// Efetuando a leitura das variáveis carregadas
// a partir do arquivo .env
console.log(process.env.VARIAVEL_1);
console.log(process.env.VARIAVEL_2);
console.log(process.env.VARIAVEL_3);
```

index.js

```
VARIAVEL_1=VALOR 1
VARIAVEL_2=VALOR 2
VARIAVEL_3=VALOR 3
```

.env

Observação: se preferir utilizar o ES6, basta incluir o `"type": "module"` no arquivo `package.json` do seu projeto. Desta forma passamos a utilizar o `import 'dotenv/config'` ao invés do `require("dotenv").config`.

Executando a aplicação temos o seguinte resultado:

```
$ node index.js

VALOR 1
VALOR 2
VALOR 3
```

Explicação

Quando executamos a aplicação, a linha `require("dotenv").config()` é executada. Esta linha lê o arquivo `.env` e carrega seus valores no `process.env`. Que então pode ser acessado ao longo da aplicação.

Vantagens

A ideia geral é que seja possível parametrizar valores que sua aplicação necessita que variam de acordo com o ambiente. Por exemplo, o nível de log no `localhost` pode ser muito mais detalhado que no ambiente de produção, onde normalmente registramos somente as mensagens de erro.

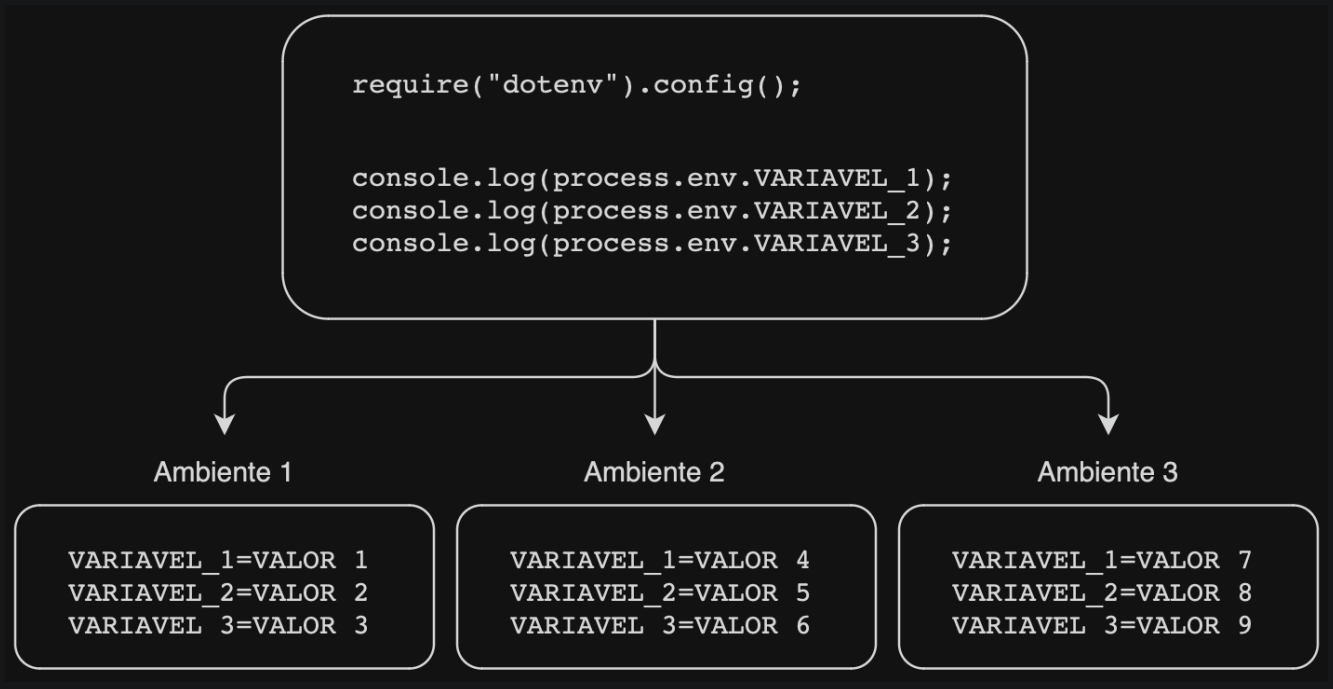

Ilustração da parametrização de variáveis por ambiente

Desta forma, no mundo real vamos ter um arquivo `.env` por ambiente.

⚠️ É desaconselhável incluir o arquivo `.env` diretamente no seu repositório. Em vez disso, disponibilize um modelo (por exemplo, `.env.example`) que contenha as variáveis de ambiente necessárias, mas sem os valores reais, para que outros desenvolvedores tenham conhecimento sobre quais variáveis precisam ser configuradas. A razão é bem simples: prevenir o vazamento de dados sensíveis. Se estiver utilizando o Git, certifique-se de adicionar uma linha ao seu arquivo `.gitignore` para excluir o arquivo `.env` e evitar que seja rastreado no repositório.

Utilizando o require module

Uma outra forma de iniciar a leitura do arquivo `.env` utilizando a `dotenv`, é através do parâmetro `-r` ou `--require` do Node.js (link da documentação):

```
node -r dotenv/config index.js
```

Implementação do Node.js 20.6.0

A partir da versão 20.6.0 do Node.js, é possível realizar a leitura do arquivo `.env` de maneira semelhante ao dotenv, mas agora de maneira nativa, eliminando a exigência de instalação da biblioteca dotenv.

Starting from Node.js v20.6.0, Node.js supports `.env` files for configuring environment variables.

Your configuration file should follow the INI file format, with each line containing a key-value pair for an environment variable. To initialize your Node.js application with predefined configurations, use the following CLI command: `node --env-file=config.env index.js`.

https://nodejs.org/ar/blog/release/v20.6.0

Então instalei a versão do 20.8.0 do Node.js para testar.

```
VARIAVEL_1=VALOR 1
VARIAVEL_2=VALOR 2
VARIAVEL_3=VALOR 3
```

.env

```
console.log(process.env.VARIAVEL_1);
console.log(process.env.VARIAVEL_2);
console.log(process.env.VARIAVEL_3);
```

index.js

Para especificar o arquivo (de variáveis de ambiente) ao Node.js, basta utilizar o argumento `--env-file`, conforme exemplificado abaixo:

```
$ node --env-file=.env index.js
VALOR 1
VALOR 2
VALOR 3
```

Considerações

Parametrizar uma aplicação Node.js com variáveis de ambiente é extremamente útil. Ajuda na segurança, flexibilidade e simplifica a configuração nos diversos ambientes onde a aplicação é executada.

Bibliotecas como dotenv ou a opção nativa (a partir do Node.js v20.6) simplificam o gerenciamento dessas variáveis. Essa abordagem não só protege dados sensíveis, mas também facilita colaboração, manutenção e implantação em ambientes diversos.

Para finalizar, vale destacar que nem sempre sua aplicação poderá utilizar o arquivo `.env`. Por exemplo, isto pode ocorrer por uma limitação de ambiente ou mesmo política de segurança da empresa. Para estes casos é necessário avaliar como podemos configurar estas variáveis de ambiente. Por exemplo, plataformas como Azure, Heroku e AWS oferecem métodos específicos para essa configuração.

Links interessantes:

- https://www.npmjs.com/package/dotenv
- https://github.com/motdotla/dotenv
- https://nodejs.org/ar/blog/release/v20.6.0

Member discussion

0 comments

Start the conversation

Become a member of **console.log** to start commenting.

Sign up now

Already a member?

Como aplicar uma máscara de telefone, CPF, IP e data com ngx-mask

Aprenda a aplicar uma máscara de telefone, CPF, entre outros, em seus input's utilizand...

24 de Nov de 2023 · 10 min read · 1 comment

Como criar um teste unitário envolvendo timers | Angular

Aprenda a criar testes unitários para componentes Angular que utilizam...

17 de Nov de 2023 · 7 min read