

Node Package Manager

Programação Web 1



Objetivos de Aprendizagem

- Utilizar o gerenciador de pacotes NPM
- Identificar funções básicas do Express

Agenda

- Instalação Node.js
- NPM
- Módulos típicos
- *Framework* Express

Node Package Manager

- Sistema de gerenciamento de pacotes do Node.js
- Similar ao pip do Python, Apt do Ubuntu, dentre outros
- Realiza todo o gerenciamento (publicação, instalação, exclusão, etc) no sistema público

Node Package Manager

- Normalmente, o NPM é instalado junto com Node.js
- Para verificar a instalação e versão basta digitar no console

```
1 node -v
```

```
2 npm -v
```

Node Version Manager

- O Node.js disponibiliza ainda um utilitário para o gerenciamento de versões
 - `nvm` (OSX e Linux)
 - `nodist` ou `nvm-windows` (Windows)
- Controladores de versões normalmente são instalados a parte
- Permite instalar e alternar entre diversas versões para testes
- Uso do NVM é feito via linha de comando

npm init

- Configuração inicial dos requisitos de um projeto Node.js novo ou existente
- Dentre outras ações cria um arquivo `package.json`
- O comando deve ser executado já na pasta do projeto

```
1 npm init
```

package.json

- Arquivo que contém as informações sobre o gerenciamento de pacotes da aplicação
- Lista todas as dependências de pacotes, inclusive versões, em um único arquivo
- Torna a estrutura da aplicação replicável em servidores de produção ou testes

Exemplo

```
1  {
2    "name": "@slidev/theme-seriph",
3    "version": "0.25.0",
4    "description": "Seriph theme for Slidev",
5    "author": "antfu <anthonyfu117@hotmail.com>",
6    "license": "MIT",
7    "funding": "https://github.com/sponsors/antfu",
8    "homepage": "https://sli.dev",
9    "repository": {
10      "type": "git",
11      "url": "https://github.com/slidesjs/themes"
12    },
13    "keywords": [
14      "slidev-theme",
15      "slidev"
16    ],
17    "engines": {
18      "node": "≥14.0.0",
19      "slidev": "≥v0.47.0"
20    },
21    "slidev": {
```

Exemplo

```
16  {
17    "engines": {
18      "node": " $\geq 14.0.0$ ",
19      "slidev": " $\geq v0.47.0$ "
20    },
21    "dependencies": {
22      "@slidev/types": " $\wedge 0.47.0$ ",
23      "codemirror-theme-vars": " $\wedge 0.1.2$ ",
24      "prism-theme-vars": " $\wedge 0.2.4$ "
25    }
26  }
```

Como criar o package.json?

1. Fornecer as informações iniciais

```
1  npm init
```

- Nome
- Versão
- Descrição
- Palavras Chaves
- Autor
- Etc

Como instalar as dependências de um projeto?

2. Efetiva a instalação

```
1  npm install
```

- `npm install` instala pacotes e as dependências
- Quando executado sem parâmetros instala **todas** as dependências listadas no arquivo `package.json`

Como instalar um pacote específico?

- `npm install <module>` instala o pacote especificado e as dependências
- O arquivo `package.json` é atualizado automaticamente

```
1 npm install lodash  
2 npm install nodemon
```

- Usando uma única linha de comando

```
1 npm install lodash nodemon
```

Instalação Global

- `npm install <module> -g` instala o pacote especificado e as dependências globalmente
- O arquivo `package.json` é atualizado automaticamente
- Em ambientes de nuvem é pouco utilizado

Resumo NPM

Comando	Ação
<code>npm init</code>	Cria o arquivo <code>package.json</code> personalizado
<code>npm init -y</code>	Cria o arquivo <code>package.json</code> padrão
<code>npm install <package></code>	Insere o pacote especificado em <code>package.json</code>
<code>npm install <package> -g</code>	Instala globalmente o pacote especificado
<code>npm install</code>	Instala todas as dependências contidas em <code>package.json</code>

Nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.

Nodemon

- Reinicia a aplicação quando o código é modificado
- Mesmo que a aplicação seja finalizada sem erros o monitoramento continua e reinicia a aplicação
- Substitui o comando `node` para inicializar as aplicações
- Não demanda nenhuma modificação no código para sua utilização

PM2 is a daemon process manager that will help you manage and keep your application online 24/7

PM2

- Realiza monitoramento similar ao nodemon, porém mais completo
- Permite o monitoramento de diversos microserviços em conjunto
- Instalação `npm install pm2 -g`
- Inicia uma aplicação `pm2 start app.js`
- Monitora todos as aplicações inicializadas com `pm2 pm2 monit`

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

Express

- *Framework* para desenvolvimento web com Nodejs bastante popular
- Provê um servidor web através do módulo `http`
- Utiliza-se de funções de *middlewares* para a criação das funcionalidades de um site ou aplicação
- Provê ainda a funcionalidade de *templates*

Exemplo de Aplicação (node-http)

```
1 const http = require('http');
2 const server = http.createServer((request, response) => {
3     const method = request.method;
4     const url = request.url;
5     response.setHeader('Content-type', 'text/html');
6     response.write('<!DOCTYPE html><html lang="pt-br">');
7     response.write('<head><meta charset="UTF-8"></head>');
8     response.write('<body>');
9     response.write('<h1>Dados da requisição</h1><br>');
10    response.write('<hr>');
11    response.write('Método da requisição: ' + method + '<br>'); response.wri
12    response.write('</body></html>');
13    response.end()
14 });
15 server.listen(3000)
```

{ Use the application generator tool, express-generator, to quickly create an application skeleton.

Express application generator

Gerador de Aplicações Rápidas

- A estrutura de uma aplicação pode ser criada rapidamente com `express-generator`
- Instalação via NPM

```
1 npm install express express-generator  
2 npm install
```

Usando o express-generator

- Com `express` e o `express-generator` instalados
- Para executar diretamente um pacote, utiliza-se o comando

`npx`

- Executa-se o `express-generator` com NPX, conforme mostrado ao lado
- Execute o comando na pasta do projeto a ser criado

```
1 npx express-generator --view=ejs app1
```

- Cria `app1` com *template engine* EJS

Usando o express-generator

- Instala as dependências e inicia a aplicação

```
1 npm install  
2 npm start
```

Dúvidas

Exercícios

1

1. Utilizar o NPM para criar o arquivo `package.json` da aplicação `node-http`, versão 1.0
 - `npm init`

2. Instalar os pacotes:
 - Nodemon
 - HTTP (`npm install nodemon http`)
3. Criar o arquivo `index.js`, inserir o código mostrado anteriormente
4. Rode a aplicação (`node index.js`)

1

5. Pare a aplicação (CTRL + D)
6. Utilize o `nodemon` para rodar a aplicação novamente (`nodemon index.js`)
7. Modifique o código em `index.js` (linha 9, por exemplo)
8. O que acontece?

2

- Instalar os pacotes:
 - nodemon
 - express
 - express-generator
- Utilize o EG para criar a estrutura básica da aplicação **app1**
- Inicie a aplicação utilizando a linha de comando (node, NPM, nodemo ou pm2)
- Verifique a seção *scripts* do arquivo `package.json`
- Modifique o *script start* para que a aplicação seja iniciada utilizando um monitor como Nodemon ou pm2

3

- Utilize o nodemon para rodar a aplicação novamente (`nodemon index.js`)
- Modifique o código em `/app1/routes/index.js` pelo código a seguir:

```
1 res.render('index', {title: 'Express', mysite: 'My first Express Site'})
```

- Adicione a linha abaixo dentro da *tag* `<body>` do arquivo

`/app1/views/index.ejs`

```
1 <p><%= mysite%>
```

- Execute a aplicação
- O que acontece?

Referências

- NPM
- Nodemon
- PM2
- Express

José Roberto Bezerra

✉️ jbroberto@ifce.edu.br

⌚ [jbroberto76](#)

⌚ [pw1-repo](#)

