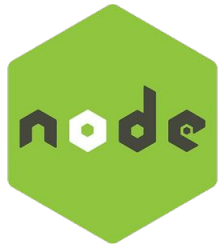
NodeJS



Prof. Celso Henrique Masotti

Você verá nesta aula:

Objetivo: protocolo HTTP, servidor nativo HTTP e Middleware.

HTTP
Usando Módulo do HTTP
Middleware

HTTP

O HTTP (Hypertext Transfer Protocol), em português “Protocolo de Transferência de Hipertexto”, é um protocolo para transferência de arquivos hipermídia, como o HTML, por exemplo. Ele foi criado para a comunicação de navegadores web e servidores, mas também pode ser empregado em outras finalidades.

Ele trabalha com o modelo cliente-servidor – o cliente abre a conexão, faz uma requisição ao servidor e aguarda até receber uma resposta. Ele é um protocolo stateless, ou seja, o servidor não armazena informações entre as requisições. Para ler mais sobre o assunto acesse o site do W3C.

O nodeJS possui o módulo HTTP como nativo. Vamos usá-lo.

Usando Módulo do HTTP

Em sua pasta de projeto crie um arquivo denominado “app_http.js” e vamos requisitar o módulo nativo.

Digite: `var http = require("http");`

proj04 > JS app_http.js > ...

```
1   var http = require("http");
2
3
```

Usando Módulo do HTTP

Mas, o servidor precisa ser “criado” para uso e para isso usamos: http.createServer()

Mas, o createServer exige a identificação da porta de comunicação com o usuário, que é o “listen(número da porta)”. Pode ser qualquer porta, por exemplo: 80, 90, etc. Contudo, não pode ser indicado uma porta que já está em uso. Por exemplo, o Skype faz uso da porta 80 quando em uso, logo, não podemos usá-la. Para evitar problemas, normalmente usamos portas de identificação alta, como “8080”, “8081”, etc. Vamos adotar a porta “8080” como padrão em nossos estudos.

Então fica:

proj04 > JS app_http.js > ...

```
1   var http = require("http");
2
3
4   http.createServer().listen(8080)
5
```

Usando Módulo do HTTP

Para sabermos que o servidor está funcionando corretamente, vamos deixar uma mensagem de console.

proj04 > JS app_http.js > ...

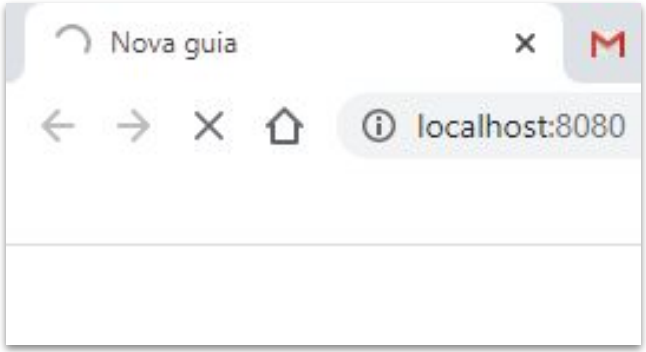
```
1   var http = require("http");
2
3
4   http.createServer().listen(8080)
5
6   console.log("Servidor ativo");
7
```

```
C:\NodeEstudos\proj04>node app_http.js
Servidor ativo
```

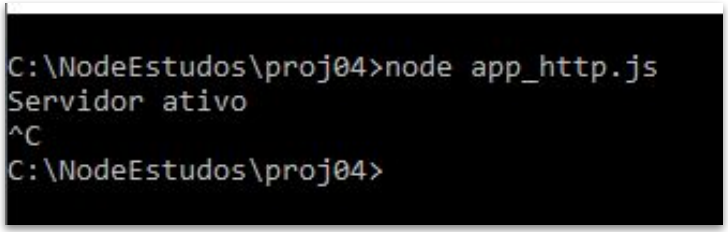
Agora, no console, vá até a pasta de projetos onde o arquivo app_http.js foi salvo e faça o node ler o arquivo: “node app_http.js”

Usando Módulo do HTTP

Se o servidor estiver funcionando, o texto “servidor ativo” irá aparecer. Caso contrário ocorrerá um erro. Agora, em uma página/aba nova do navegador (browser) chame o servidor local (localhost) http e a porta especificada: 8080.



O que acontece? Nada apareceu? Pois é! E a página fica carregando indefinidamente, não é? Isso acontece porque não colocamos nada para o servidor ler e dar ao visitante. Vamos fazer isso. Mas, antes, vamos parar o servidor http do nodeJS. Utilize o “Ctrl + C”



Middleware

Middleware é todo tipo de função que está entre um pedido HTTP e a resposta final que o servidor envia de volta para o cliente (callback).

Middleware tem acesso ao objeto de solicitação (*req*), o objeto de resposta (*res*), e a próxima função de middleware no ciclo solicitação-resposta do aplicativo. A próxima função middleware é comumente denotada por uma variável chamada *next*.

Funções de middleware podem executar as seguintes tarefas:

- Executar qualquer código.
- Fazer mudanças nos objetos de solicitação e resposta.
- Encerrar o ciclo de solicitação-resposta.
- Chamar o próximo middleware na pilha.

Middleware

A função middleware tem 3 parâmetros, pedido (“request” ou “req”), resposta (“response” ou “res”) e callback (“next”). Pode ter n middleware a processar um pedido HTTP, encadeados. Quando um middleware acaba de processar coloca-se no final do código next();, invocando assim a callback e o código continua a correr para o próximo middleware ou resposta final.

O middleware é portanto uma funcionalidade, funções que executam processos intermédios. Os exemplos mais comuns são interagir com a BD, ir buscar arquivos estáticos, tratar de erros ou redirecionamentos.

Middleware

Voltamos ao nosso servidor. Para que um arquivo retorno ao visitante, temos de trabalhar com o “res”, portanto. Assim, no interior de “createServer” inseriremos a função.

proj04 > JS app_http.js > ...

```
1   var http = require("http");
2
3
4   http.createServer(function(req,res){
5       // aqui inserimos o que será devolvido ao usuário.
6   }).listen(8080)
7
8   console.log("Servidor ativo");
9
```

Middleware

Enviemos apenas um “Olá!” ao usuário para sabermos se o middleware está funcionando adequadamente. Para isso especificaremos um “end” contendo a mensagem após o “res”, ficando assim:

proj04 > JS app_http.js > ...

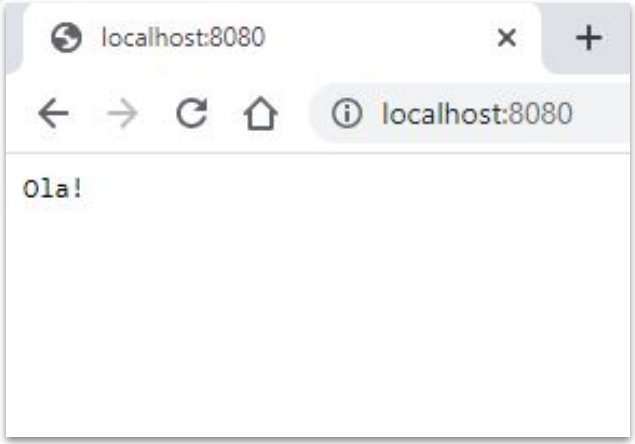
```
1   var http = require("http");
2
3
4   http.createServer(function(req,res){
5       // aqui inserimos o que será devolvido ao usuário.
6       res.end("Ola!");
7   }).listen(8080)
8
9   console.log("Servidor ativo");
```

Middleware

Agora “subir” nosso servidor mais uma vez e ver o que acontece. Vá ao seu terminal e “chame” outra vez o arquivo.

```
C:\NodeEstudos\proj04>node app_http.js
Servidor ativo
```

Vejamos em um navegador (browser)



Pronto! Nosso servidor HTTP está funcionando e enviando a mensagem ao usuário.

Obrigado

Prof. Celso Henrique Masotti