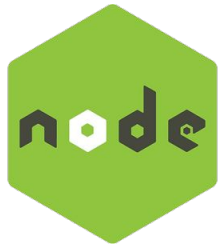
NodeJS



Prof. Celso Henrique Masotti

Você verá nesta aula:

Objetivo: Framework, servidor HTTP Express, Roteamento e Parâmetros.

Framework
Express
Express - Roteamento
Express - Parâmetros
Express - Página HTML

Framework

O módulo HTTP nativo do nodeJS é muito restrito e com aplicabilidade muito específica, assim foi criado alguns frameworks para ocupar tal espaço e que oferece todos os recursos necessários para o bom desenvolvimento web.

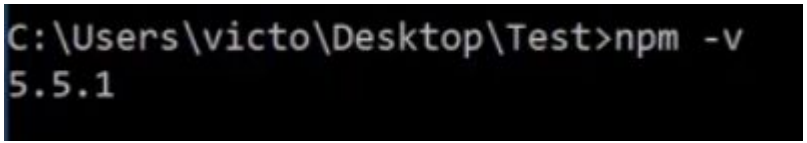
Um framework - em desenvolvimento de software - é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. Em outras palavras, um framework é um pacote utilitário pronto para uso que agiliza muito o desenvolvimento de softwares.

Express

O Express é um framework para aplicativo da web do Node.js mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel.

Seu site oficial é https://expressjs.com/pt-br/

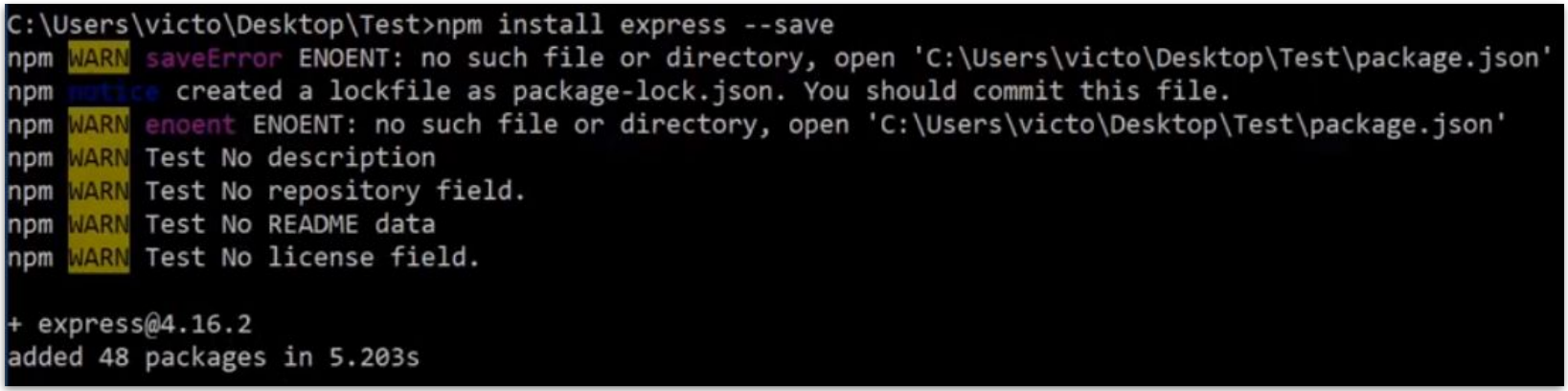
E sua instalação é feita com o terminal NPM. Para saber se tem o NPM instalado digite os comando “npm --v” em seu console. Caso tenha instalado a versão do seu NPM será exibida.



Express

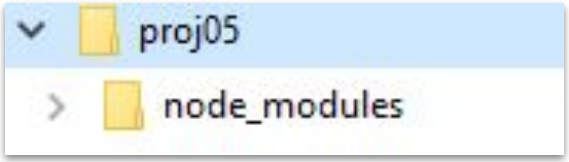
Abrindo seu terminal, vá até sua pasta de projeto e use os comando de instalação:

```
$ npm install express --save
```



Express

O NPM cria uma pasta de nome “node_modules” e realiza o download de todos os módulos que compõe o framework solicitado.



Caso queira saber tudo o que o NPM instalou para o seu projeto, abra o arquivo “package-lock.json” que aparece junto à pasta “node_modules”, neste está registrado tais informações.

Express

Vamos iniciar nosso trabalho.

Crie um arquivo intitulado “index.js” em sua pasta de trabalho e inclua o módulo “express” recém criado. Mas, observe uma coisa. Estamos acostumados com endereçamentos e nossa primeira ideia é fazer algo como:

```
proj05 > JS index.js > ...
  1    var express = require("express");
```

```
proj05 > JS index.js > ...
  1    var express = require("./node_modules/express");
```

Mas, como o NPM instalou adequadamente o “express” e este é reconhecido pelo nodeJS, não é preciso o endereçamento, ficando assim:

Express

Agora vamos criar uma variável que receberá o módulo “express” e todos os seus recursos:

proj05 > JS index.js > ...

```
1   var express = require("express");
2   var app = express();
```

Estas variáveis não se modificarão durante o processamento do sistema, logo, é melhor transformá-las em constantes.

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
```

Express

Para que o servidor “express” entre em funcionamento, basta inserir o listen e o número da porta.

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.listen(8080);
```

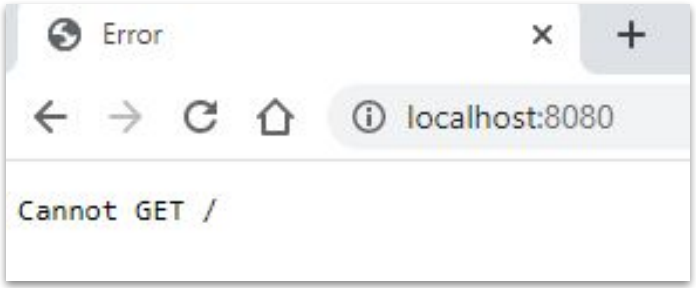
Uma informação muito importante: nunca escreva linhas de código que esteja relacionada com o “express” abaixo do *listen* pois não irá funcionar e pode dar erro. *listen* deve ser a última linha do “express” sempre.

Express

Vamos “subir” o servidor http da forma como já fizemos antes:

```
C:\NodeEstudos\proj05>node index.js
```

Agora abra uma aba no seu browser e verifique a atividade chamando o servidor local e a porta.



Express

O servidor está funcionando perfeitamente, embora o erro “get”. Ainda não mostramos qual “roteamento” será usada na aplicação. Iremos falar sobre isso um pouco mais a frente.

Desligue o servidor com as teclas “Ctrl + c” no console.

```
C:\NodeEstudos\proj05>node index.js
^C
C:\NodeEstudos\proj05>
```

Express

Vamos colocar um texto para que o console informe que o servidor entrou em atividade sempre dermos “start”. Para isso usaremos uma função de callback dentro de listen.

Você sabe que todos os eventos no javascript tem uma resposta, um retorno, que é o callback. Só precisamos capturar este retorno e pedir para que um texto seja colocado no console. E isto é bem simples. Basta inserir uma função como sendo um parâmetro de listen.

```
function(){
        console.log(“Servidor em funcionamento”);
}
```

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.listen(8080,function(){
5       // coloque o texto aqui
6       console.log("Servidor em funcionamento!");
7   });
```

Express

O resultado será:

```
C:\NodeEstudos\proj05>node index.js
Servidor em funcionamento!
```

Express - Roteamento

O Roteamento refere-se à determinação de como um aplicativo responde a uma solicitação do cliente por um endpoint específico, que é uma URI (ou caminho) e um método de solicitação HTTP específico (GET, POST, e assim por diante).

Cada rota pode ter uma ou mais funções de manipulação, que são executadas quando a rota é correspondida.

A definição de rotas aceita a seguinte estrutura:

```
app.METHOD(PATH, HANDLER)
```

- app é uma instância do express.
- METHOD é um método de solicitação HTTP.
- PATH é um caminho no servidor.
- HANDLER é a função executada quando a rota é correspondida.

Express - Roteamento

Os seguintes exemplos ilustram a definição de rotas simples.
Responder com Hello World! na página inicial:

```
app.get('/', function (req, res) {
  res.send('Hello World!');
});
```

Express - Roteamento

Responder a uma solicitação POST na rota raiz (/) com a página inicial do aplicativo:

```
app.post('/', function (req, res) {
  res.send('Got a POST request');
});
```

Express - Roteamento

Assim, quando o roteamento foi criado adequadamente a resposta no browser é certeira e imediata.

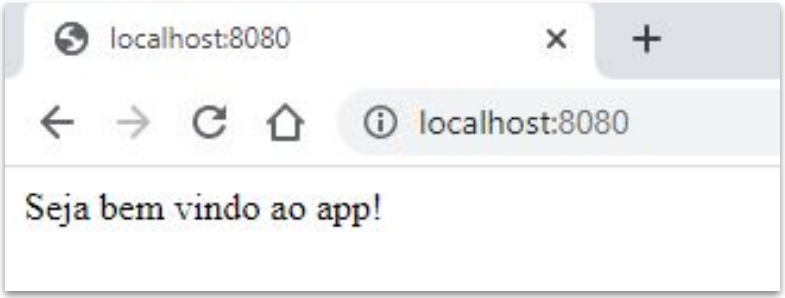
proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.listen(8080,function(){
9       // coloque o texto aqui
10      console.log("Servidor em funcionamento!");
11  });
```

Agora iniciemos novamente o servidor http no console.

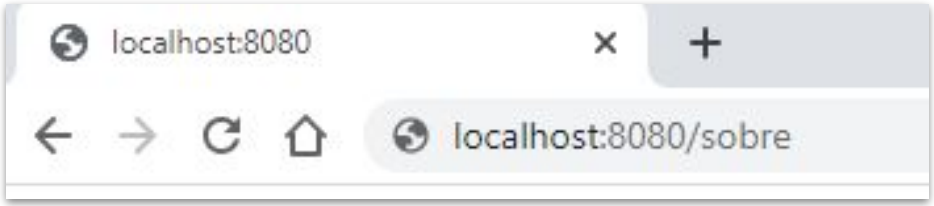
```
C:\NodeEstudos\proj05>node index.js
Servidor em funcionamento!
```

Vejamos como fica o browser:



Express - Roteamento

Agora iremos criar uma nova “rota”, um novo roteamento de tal forma que no browser, após a informação da porta, inseriremos “/” barra e roteamento, por exemplo, “sobre”. Ficará assim:



Express - Roteamento

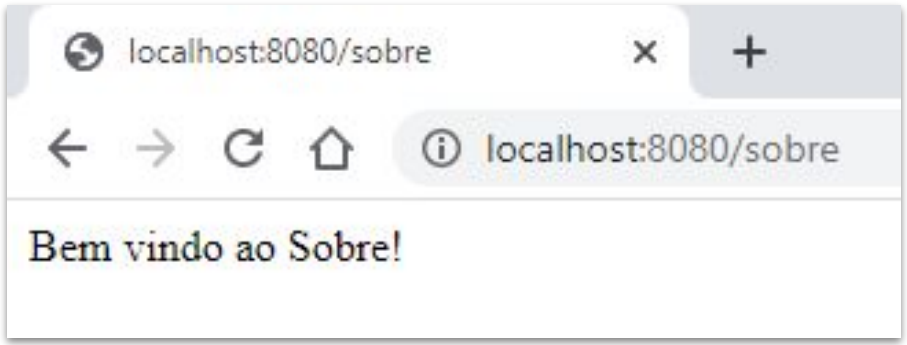
Precisamos informar ao sistema sobre essa nova rota, portanto:

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre",function(req,res){
9       res.send("Bem vindo ao Sobre!");
10  })
11
12  app.listen(8080,function(){
13      // coloque o texto aqui
14      console.log("Servidor em funcionamento!");
15  });
```

Express - Roteamento

Vejamos seu funcionamento. Reinicie o servidor http e solicite a nova rota em seu navegador.



Express - Parâmetros

Parâmetros são espaços dinâmicos aplicados ao roteamento. É possível enviar informações ao sistema por estes parâmetros.

Criamos parâmetros usando dois pontos “:” após barra e uma denominação para seu reconhecimento pelo sistema, algo como:

```
app.get("/sobre/:idade",function(req,res){
    res.send("Bem vindo ao Sobre!");
})
```

Express - Parâmetros

No código todo:

proj05 > JS index.js > ...

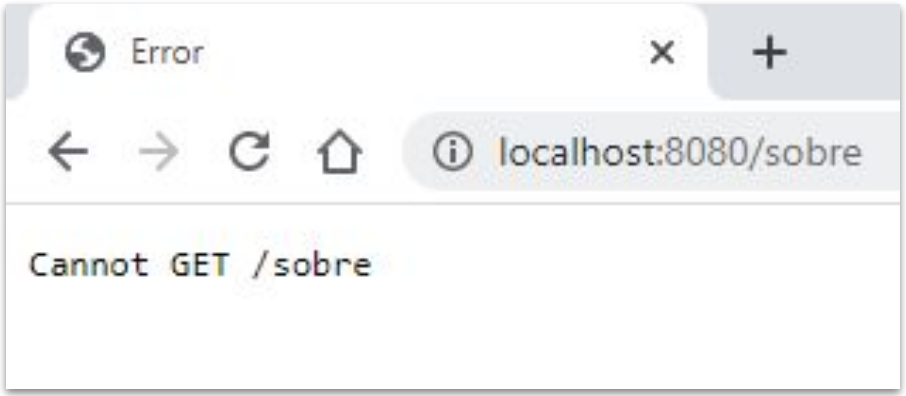
```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre/:idade",function(req,res){
9       res.send("Bem vindo ao Sobre!");
10  })
11
12  app.listen(8080,function(){
13      // coloque o texto aqui
14      console.log("Servidor em funcionamento!");
15  });
```

Express - Parâmetros

No exemplo acima “idade” é a identificação de um novo parâmetro criado para a rota “sobre”.

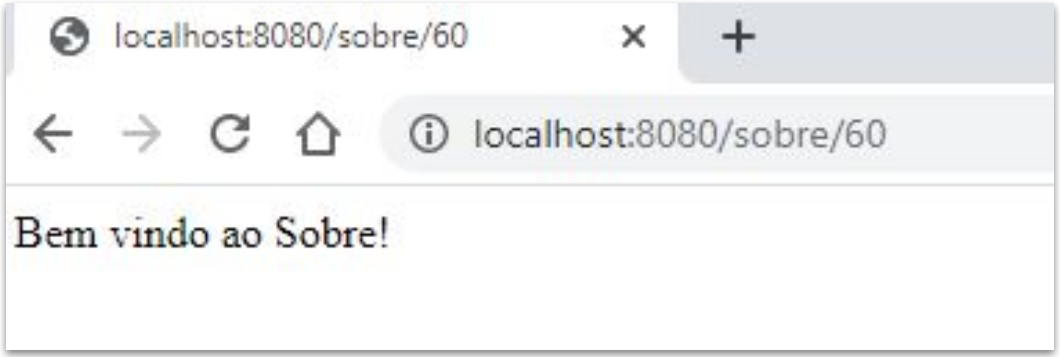
Vamos inicializar nosso servidor http e verificar no browser.

Caso utilizemos a informação de rota antiga, ocorrerá um erro:



Express - Parâmetros

Mas, se inserirmos barra após “sobre” e colocarmos algo, por exemplo, sua idade, você terá outro resultado.



Express - Parâmetros

Vamos fazer com que o sistema capture a informação deixada pelo usuário no parâmetro. Para isso, no “send”, aplicamos a requisição de parâmetro “req.params”. Nosso código ficará assim:

```
app.get("/sobre/:idade",function(req,res){
    res.send(req.params);
})
```

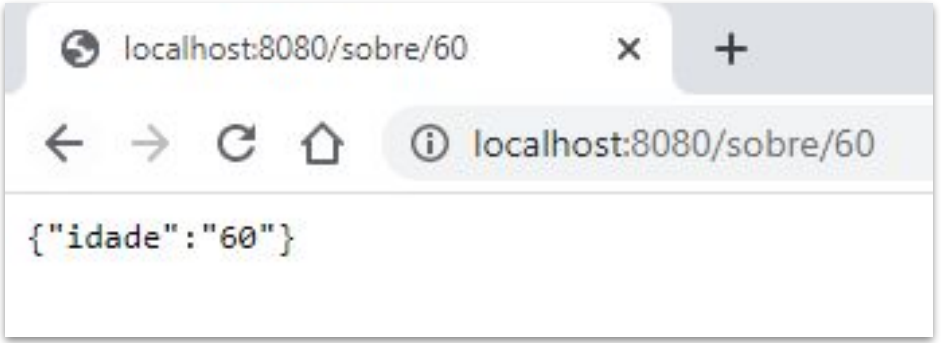
proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre/:idade",function(req,res){
9       res.send(req.params);
10  })
11
12  app.listen(8080,function(){
13      // coloque o texto aqui
14      console.log("Servidor em funcionamento!");
15  });
```

Express - Parâmetros

Reinicie o servidor http e verifique seu navegador inserindo algum valor no parâmetro.

```
C:\NodeEstudos\proj05>node index.js
Servidor em funcionamento!
```



Express - Parâmetros

Vamos criar mais um parâmetro na rota “sobre”. Agora será o “nome”. Então nosso código ficará assim:

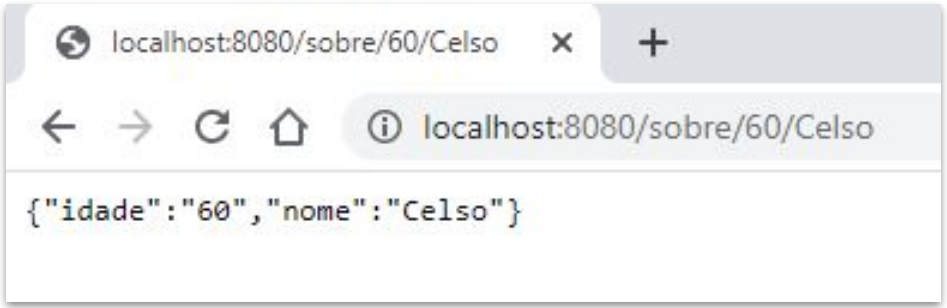
```
app.get("/sobre/:idade/:nome",function(req,res){
    res.send(req.params);
})
```

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre/:idade/:nome",function(req,res){
9       res.send(req.params);
10  })
11
12  app.listen(8080,function(){
13      // coloque o texto aqui
14      console.log("Servidor em funcionamento!");
15  });
```

Express - Parâmetros

Vamos executar o arquivo novamente no servidor http, passando o novo parâmetro:



Express - Parâmetros

Vamos trabalhar a impressão das informações de forma individualizadas. Para isso basta colocar um ponto e a identificação do parâmetro desejado.

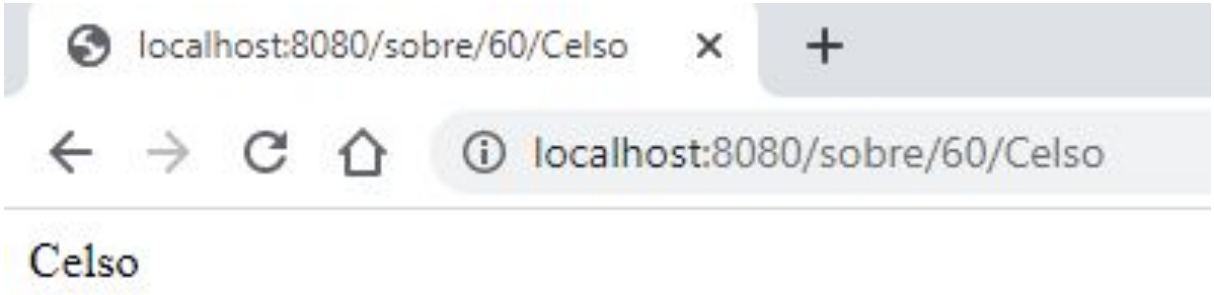
```
app.get("/sobre/:idade/:nome",function(req,res){
    res.send(req.params.nome);
})
```

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre/:idade/:nome",function(req,res){
9       res.send(req.params.nome);
10  })
11
12  app.listen(8080,function(){
13      // coloque o texto aqui
14      console.log("Servidor em funcionamento!");
15  });
```

Express - Parâmetros

Quando o servidor for ativado novamente, o seu navegador exibirá:



Express - Parâmetros

Podemos concatenar textos e parâmetros. Algo como:

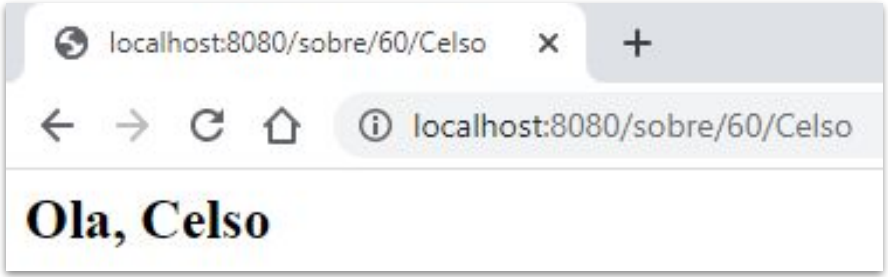
```
app.get("/sobre/:idade/:nome",function(req,res){
    res.send("<h2>Ola, " + req.params.nome + "</h2>");
})
```

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre/:idade/:nome",function(req,res){
9       res.send("<h2>Ola, " + req.params.nome + "</h2>");
10  })
11
12  app.listen(8080,function(){
13      // coloque o texto aqui
14      console.log("Servidor em funcionamento!");
15  });
```

Express - Parâmetros

Inicie novamente o servidor HTTP e dê uma olhada no resultado:



Express - Parâmetros

Vamos colocar as duas informações na tela. É preciso criar mais uma linha de “send” para que trabalhe o outro parâmetro.

```
app.get("/sobre/:idade/:nome",function(req,res){
    res.send("<h2>Ola, " + req.params.nome + "</h2>");
    res.send("<h3>Voce tem, " + req.params.idade + "</h3>");
})
```

Express - Parâmetros

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre/:idade/:nome",function(req,res){
9       res.send("<h2>Ola, " + req.params.nome + "</h2>");
10      res.send("<h3>Voce tem, " + req.params.idade + "</h3>");
11  })
12
13  app.listen(8080,function(){
14      // coloque o texto aqui
15      console.log("Servidor em funcionamento!");
16  });
```

Express - Parâmetros

Inicie novamente o servidor HTTP e veja o resultado no browser:
Não aparece o que desejamos? Ocorreu erro no servidor?

```
C:\NodeEstudos\proj05>node index.js
Servidor em funcionamento!
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:535:11)
    at ServerResponse.header (C:\NodeEstudos\proj05\node_modules\express\lib\response.js:771:10)
    at ServerResponse.contentType (C:\NodeEstudos\proj05\node_modules\express\lib\response.js:599:15)
    at ServerResponse.send (C:\NodeEstudos\proj05\node_modules\express\lib\response.js:145:14)
    at C:\NodeEstudos\proj05\index.js:10:9
    at Layer.handle [as handle_request] (C:\NodeEstudos\proj05\node_modules\express\lib\router\layer.js:95:5)
    at next (C:\NodeEstudos\proj05\node_modules\express\lib\router\route.js:137:13)
    at Route.dispatch (C:\NodeEstudos\proj05\node_modules\express\lib\router\route.js:112:3)
    at Layer.handle [as handle_request] (C:\NodeEstudos\proj05\node_modules\express\lib\router\layer.js:95:5)
    at C:\NodeEstudos\proj05\node_modules\express\lib\router\index.js:281:22
Error [ERR_HTTP_HEADERS_SENT]: Cannot set headers after they are sent to the client
    at ServerResponse.setHeader (_http_outgoing.js:535:11)
    at ServerResponse.header (C:\NodeEstudos\proj05\node_modules\express\lib\response.js:771:10)
    at ServerResponse.contentType (C:\NodeEstudos\proj05\node_modules\express\lib\response.js:599:15)
    at ServerResponse.send (C:\NodeEstudos\proj05\node_modules\express\lib\response.js:145:14)
    at C:\NodeEstudos\proj05\index.js:10:9
```

Express - Parâmetros

Isto ocorre porque só podemos fazer uso do “send” uma única vez por função. Logo, precisamos pegar o conteúdo da segunda linha do “send” e inserir concatenado na primeira linha.

```
app.get("/sobre/:idade/:nome",function(req,res){
    res.send("<h2>Ola, " + req.params.nome + "</h2>" +
    "<h3>Voce tem, " + req.params.idade + "</h3>");
    res.send();
})
```

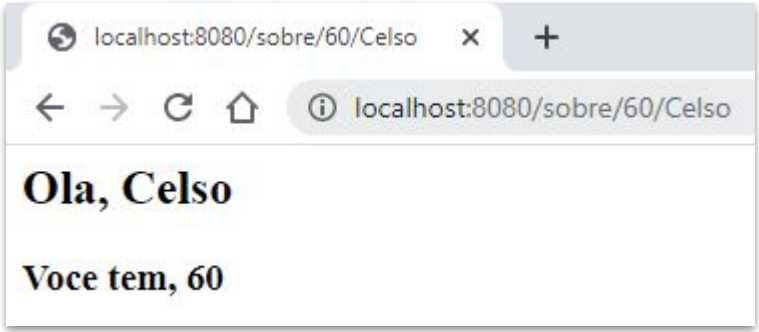
Express - Parâmetros

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/sobre/:idade/:nome",function(req,res){
9       res.send("<h2>Ola, " + req.params.nome + "</h2>" +
10      "<h3>Voce tem, " + req.params.idade + "</h3>");
11      res.send();
12  })
13
14  app.listen(8080,function(){
15      // coloque o texto aqui
16      console.log("Servidor em funcionamento!");
17  });
```

Express - Parâmetros

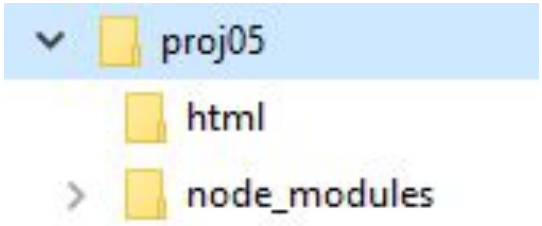
Reinicie novamente seu servidor HTTP e veja o resultado em seu navegador:



Express - Página HTML

Se seu objetivo é fazer com que uma página HTML apareça no navegador saiba que com o Express isso é muito simples. Tão simples que não precisará do módulo fs” instalado.

No interior do meu projeto criarei uma pasta chamada “html” onde colocarei os arquivos html que serão exibidos no navegador.



Express - Página HTML

No interior de html criarei dois arquivos html. O primeiro será chamado de “index.html” e o segundo de “info.html”.

O conteúdo do “index.html”

proj05 > html > index.html > html

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset='utf-8'>
5       <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6       <title>App</title>
7       <meta name='viewport' content='width=device-width, initial-scale=1'>
8   </head>
9   <body>
10      <h1>Voce esta no INDEX.HTML</h1>
11
12  </body>
13  </html>
```

Express - Página HTML

info.html

proj05 > html > info.html > html

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset='utf-8'>
5       <meta http-equiv='X-UA-Compatible' content='IE=edge'>
6       <title>Page Title</title>
7       <meta name='viewport' content='width=device-width, initial-scale=1'>
8   </head>
9   <body>
10      <h1>Você está no INFO.HTML</h1>
11  </body>
12  </html>
```

Express - Página HTML

No nosso programa no lugar de “send” usaremos “sendFile” e o endereçamento da página HTML na função que trabalha o roteamento. Para informar o endereçamento a partir do root, usamos a variável global “__dirname” concatenada com a pasta e o nome do arquivo que desejamos que seja impressa no browser. Criamos uma função para cada rota. Chamo a primeira rota de “inicio” e esta abrirá a página “index.html” e a segunda rota de “info” e abrirá o arquivo “info.html”.

```
app.get("/inicio",function(req,res){
    res.sendFile(__dirname + "/html/index.html")
})

app.get("/info",function(req,res){
    res.sendFile(__dirname + "/html/info.html")
})
```

Express - Página HTML

Visão geral do código:

proj05 > JS index.js > ...

```
1   const express = require("express");
2   const app = express();
3
4   app.get("/",function(req,res){
5       res.send("Seja bem vindo ao app!");
6   });
7
8   app.get("/inicio",function(req,res){
9       res.sendFile(__dirname + "/html/index.html")
10  })
11
12  app.get("/info",function(req,res){
13      res.sendFile(__dirname + "/html/info.html")
14  })
15
16  app.listen(8080,function(){
17      // coloque o texto aqui
18      console.log("Servidor em funcionamento!");
19  });
```

Express - Página HTML

Vamos inicializar o servidor http e verificar nosso nasso browser:

```
C:\NodeEstudos\proj05>node index.js
Servidor em funcionamento!
```

Obrigado

Profº. Celso Henrique Masotti