IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

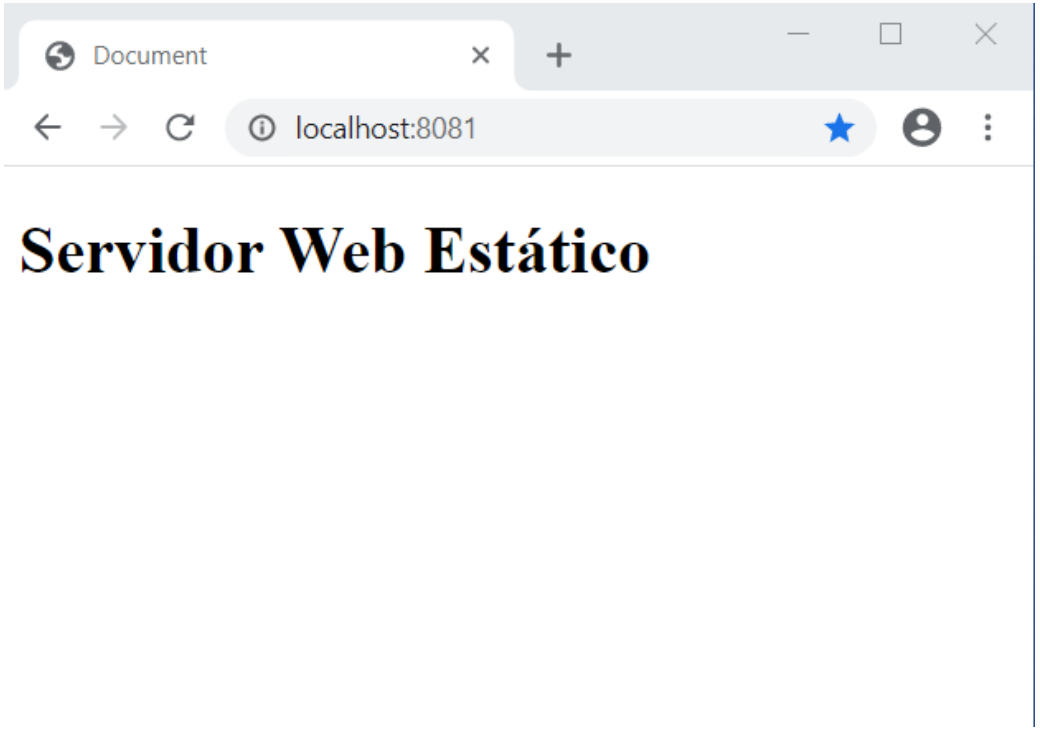
Programação Web
2020/2021

Lab05

Objetivos

Implementar um Servidor Web Estático em Node.JS

Complementar com a criação de mecanismos dinâmicos.



Requisitos

Se ainda não o fez, deve descarregar os ficheiro de apoio ao Laboratório, disponíveis no Moodle.

Após descarregar, deve extrair os ficheiros e abrir a pasta do projecto no VSCode.

Servidor Web Estático

- Inicie o projeto copiando o servidor Web Estático como foi apresentado na aula e executando o mesmo com o Debugger.

- Teste com o link http://localhost:8081

- Teste com a execução dos pedidos dinâmicos - /meupedido e /meupedido2

Limpeza e modularização do ficheiro app.js

Altere o ficheiro app.js para o tornar mais "limpo", ter menos código intrusivo.

1) Copie o objeto "options" para um ficheiro "options.json". Apenas a definição do objeto.
2) Substitua a atribuição das opções por:

```
var options = require("./options.json" )
```

3) Teste de novo o servidor

Limpeza e modularização do ficheiro app.js

1) Abaixo da variável options, crie a variável ACTIONS

```
var ACTIONS = {
   "/meupedido" : meupedido,
   "/meupedido2": meupedido2
};
```

2) Altere a função de routing "router", para que esta devolva um objecto com a função de suporte ao pedido ou a path do ficheiro estático de suporte

```
// Se devolver um nome de fiheiro:
return {
   handler : null,
   filename: "file"
};
```

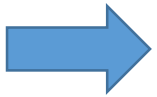
```
// Se devolver uma função:
return {
   handler : funcao,
   filename: null
};
```

Utilize a variável ACTIONS para identificar a função de suporte, caso exista (ver slide seguinte)

Limpeza e modularização do ficheiro app.js

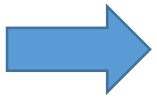
Na função router:

```
if ( pathname == "/meupedido" )
    return pathname;
if ( pathname == "/meupedido2" )
    return pathname;
```



```
if ( ACTIONS /* todo */ )
    return {
              handler: ACTIONS /* todo /*,
              filename: null
            }
```

```
return pathname
? path.join(__dirname, options.default.folder, pathname)
 : null
```



```
return {
      handler: null,
      filename: pathname
      ? path.join(__dirname, options.default.folder, pathname)
      : null
   }
```

Limpeza e modularização do ficheiro app.js

1) Altere o código de http.createServer(function (request, response)
Para que este responda às alterações feitas na função router

2) Teste o servidor após as alterações

Limpeza e modularização do ficheiro app.js

Vamos agora separar as funções de suporte aos pedidos do ficheiro app

1) Crie um novo modulo chamado “requests.js”
2) Copie as funções **meupedido** e **meupedido2** para o novo ficheiro e exporte-as!

   module.exports.meupedido = meupedido; …..
3) Importe as funções em app.js

   var requests = require(“./requests”);
4) Altere o objeto ACTIONS para usar as respetivas funções
5) Teste

Implementação do suporte a querystrings.

1) No módulo “requests.js”, crie mais uma função: **mostradados** e exporte-a.
2) Crie na variável ACTIONS o mapping respetivo.
3) Implemente a função mostradados, de forma a que esta mostre no browser o nome e a idade passados como parametros no link:
   http://localhost:8081/mostradados?nome=Albertino&idade=34
4) Mostre-os como quiser

Este exercício é um desafio. Obriga a perceber como ler os parâmetros passados na querystring.

Sugestão: var urlparsed = url.parse(request.url)
o objeto urlparsed tem um campo chamado **query** que contém os dados dos parâmetros.

Bom trabalho!