Laboratório Nº 3: Express e Funções de Middleware [Continuação]

Programação e Integração de Serviços
2020/2021

Prof. Filipe Mariano

Objetivos

- Aprofundar alguns aspetos de Node JS, Express e utilização de funções de Middleware

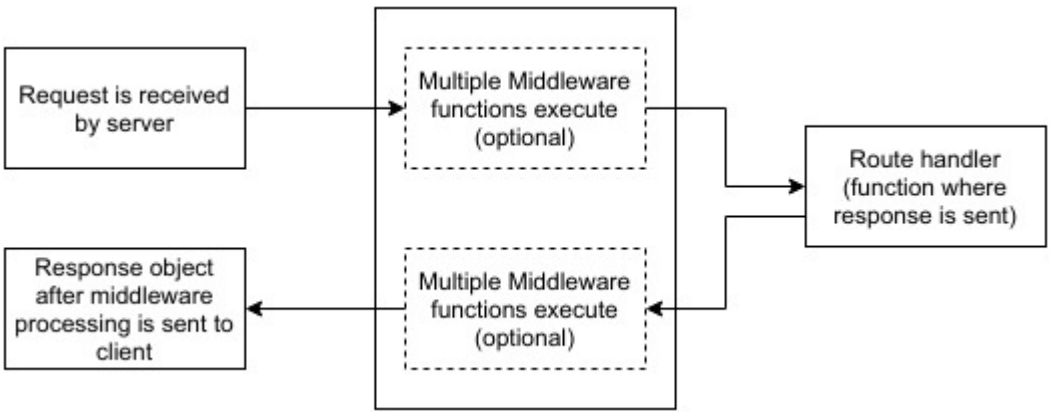
Introdução

No laboratório anterior iniciou-se a criação de um Servidor Web em Node JS recorrendo ao módulo do Express, em que os principais aspetos abordados foram a inicialização do servidor e o roteamento (com ou sem módulos).

O principal objetivo deste laboratório é esclarecer e aprofundar alguns aspetos acerca da framework Express e respetiva utilização de funções de Middleware que suscitaram algumas dúvidas na aula anterior.

Funções de *Middleware*

As funções de *middleware* são funções que têm acesso ao objeto do *request*, ao *response* e à próxima ação de *middleware* a ser executada (normalmente através da chamada à função `next()`) no ciclo de *request-response*. Geralmente, são funções utilizadas para modificar os objetos de *request* ou *response* para realizarem *parsing* ao body dos pedidos, adicionar cabeçalhos à resposta do servidor, etc.



Exemplo 1

Exemplo 1.1

O exemplo seguinte demonstra uma função de *middleware* que é chamada a cada pedido do cliente.

```
var express = require('express');
var app = express();
```

```
//Função de middleware para registar um log a cada pedido
app.use(function(req, res, next){
   console.log("A new request received at " + Date.now());

   //A seguinte função é muito importante e indica que existe ainda processamento
a ser feito
   next();
});

app.listen(8081);
```

Exemplo 1.2

Também pode ser utilizado de forma a restringir a sua chamada apenas a certas rotas. Para tal, deverá indicar a rota ou parte dela:

```
var express = require('express');
var app = express();

app.use('/rotas', function(req, res, next){
   console.log("A request for 'rotas' received at " + Date.now());
   next();
});

app.get('/rotas', function(req, res){
   res.send('teste');
});

app.listen(8081);
```

Exemplo 2

Um dos aspetos mais importantes acerca de *middleware* em Express é a ordem pela qual são escritas, são executadas e qual a rota correspondente. De seguida, será possível ver um exemplo em que é possível utilizar o *middleware* antes e depois da função de *callback* da rota atuar.

```
var express = require('express');
var app = express();

//Middleware: sempre executado
app.use(function(req, res, next){
   console.log("Start");
   next(); //importante para continuar o processamento
});

//Rota: apenas executada porque o app.use anterior tem o next()
app.get('/', function(req, res, next){
```

```
   res.send("Middle");
   next(); //importante para continuar o processamento
});

//Middleware: apenas executada porque o app.get anterior tem o next()
app.use('/', function(req, res){
   console.log('End'); //finaliza por isso não precisa do next()
});

app.listen(8081);
```

Exemplo 3

Exemplo 3.1

Crie uma função de *middleware* através do módulo Express que sirva de `contador` para todas as rotas que tenham sido pedidas pelo cliente e que vá guardando o url e o número de vezes que foi pedida essa mesma rota. A cada nova rota pedida, deverá escrever na consola todas as rotas já visitadas e respetivas contagens, por ordem decrescente do número de pedidos.

```
var express = require('express');
var app = express();

var urls = [];

//Middleware: sempre executado
app.use(function (req, res, next) {
    if (urls[req.url]) {
        urls[req.url] += 1;
    } else {
        urls[req.url] = 1;
    }

    console.log(urls.sort());
    next();
});

app.get('/', function(req, res, next){
   res.send("Root");
});

app.get('/rota/subrota1', function(req, res, next){
   res.send("Rota 1");
});

app.get('/rota/subrota2', function(req, res, next){
   res.send("Rota 2");
});
```

```
app.listen(8081);
```

Exemplo 3.2

Middleware idêntico ao Exemplo 4.1 mas a funcionar apenas para os caminhos `/rota`.

```
var express = require('express');
var app = express();

var urls = [];

//Middleware: apenas executado para as rotas /rota
app.use('/rota',function (req, res, next) {
    if (urls[req.url]) {
        urls[req.url] += 1;
    } else {
        urls[req.url] = 1;
    }

    console.log(urls.sort());
    next();
});

app.get('/', function(req, res, next){
   res.send("Root");
});

app.get('/rota/subrota1', function(req, res, next){
   res.send("Rota 1");
});

app.get('/rota/subrota2', function(req, res, next){
   res.send("Rota 2");
});


app.listen(8081);
```

Exemplo 4

Exemplo 4.1

Crie uma função de *middleware* através do módulo Express que sirva para guardar num ficheiro `api-requests.txt` a `data`, o `ip` e o `url` dos pedidos que chegam à API RESTful. Integre essa função de *middleware* no módulo de rotas `api.js`.

```

```

```
var express = require('express');
var app = express();

//Middleware: sempre executado
router.use(function (req, res, next) {
    fs.appendFileSync("api-requests.txt", Date.now() + ' | ' + req.ip + ' : ' +
req.url + " \n");
    next();
});

app.get('/', function(req, res, next){
   res.send("Root");
});

app.get('/rota/subrota1', function(req, res, next){
   res.send("Rota 1");
});

app.get('/rota/subrota2', function(req, res, next){
   res.send("Rota 2");
});



app.listen(8081);
```

Exemplo 4.2

Middleware idêntico ao Exemplo 4.1 mas a funcionar apenas para os caminhos `/rota`.

```
var express = require('express');
var app = express();

//Middleware: apenas executado para as rotas /rota
router.use('/rota',function (req, res, next) {
    fs.appendFileSync("api-requests.txt", Date.now() + ' | ' + req.ip + ' : ' +
req.url + " \n");
    next();
});

app.get('/', function(req, res, next){
   res.send("Root");
});

app.get('/rota/subrota1', function(req, res, next){
   res.send("Rota 1");
});

app.get('/rota/subrota2', function(req, res, next){
   res.send("Rota 2");
```

```
});

app.listen(8081);
```