

Programação Web 1

Express

Objetivo de Aprendizagem

- Aplicar os conceitos fundamentais do *framework* Express

Agenda

- Funcionamento básico
- *Middlewares*
- Rotas

Funcionamento Básico

app.js

```
1  const express = require('express')
2  const app = express()
3  const port = 3000
4
5  app.get('/', (req, res) => {
6    res.send('Hello World!')
7  })
8
9  app.listen(port, () => {
10    console.log(`Example app listening on port ${port}`)
11  })
```

Tratamento de Requisições

Caso o servidor receba (`req`) uma requisição GET em `/` a função de *callback* será retornada

```
1  const express = require('express')
2  const app = express()
3  const port = 3000
4
5  app.get('/', (req, res) => {
6    res.send('Hello World!')
7  })
8
9  app.listen(port, () => {
10    console.log(`Example app listening on port ${port}`)
11  })
```

Roteamento Básico

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET , POST , and so on).

Roteamento Básico

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

- *Routing*
- `app` = instância do `express`
- **METHOD** = método HTTP a ser tratado
- **PATH** = caminho no servidor
- **HANDLER** = função que será executada quando a rota for recebida

Exemplos

```
1 app.put('/', (req, res) => {  
2     res.send('Got a PUT request')  
3 })  
4 app.post('/', (req, res) => {  
5     res.send('Got a POST request')  
6 })
```

Exemplos

```
1 app.put('/', (req, res) => {  
2     res.send('Got a PUT request')  
3 })  
4 app.post('/', (req, res) => {  
5     res.send('Got a POST request')  
6 })
```

**Como realizar
testes com
diferentes
métodos?**

Postman is an API platform for building and using APIs. It simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs –faster.

app.all

Método de roteamento especial utilizado para todos os métodos HTTP (GET, POST, PUT, DELETE, etc)

```
1 app.all('/secret', (req, res, next) => {  
2   console.log('Accessing the secret section ...')  
3   next() // pass control to the next handler  
4 })  
5 app.get('/secret', (req, res) => {  
6   res.send("Secret method accessed")  
7 })
```

Parâmetros de Rotas

São utilizados para passar parâmetros pelas URLs via `req.params`

```
1 app.get('/user/:id', (req, res) => {  
2     res.send(`user ${req.params.id}`)  
3 })
```

Exemplos

Route path: `/flights/:from-:to`

Request URL: `http://localhost:3000/flights/LAX-SFO`

`req.params: { "from": "LAX", "to": "SFO" }`

Exemplos

Exemplos

Route path: `/plantae/:genus.:species`

Request URL: `http://localhost:3000/plantae/Prunus.persica`

`req.params: { "genus": "Prunus", "species": "persica" }`

Propriedades e Métodos Úteis

request

- `req.url`
- `req.cookies`
- `req.ip`
- `req.params`
- `req.path`

response

- `res.attachment()`
- `res.cookie()`
- `res.end()`
- `res.status()`
- `res.json()`

Middleware

Middleware functions have access to the request object (`req`), the response object (`res`), and the `next` function in the application's request-response cycle.

Middlewares

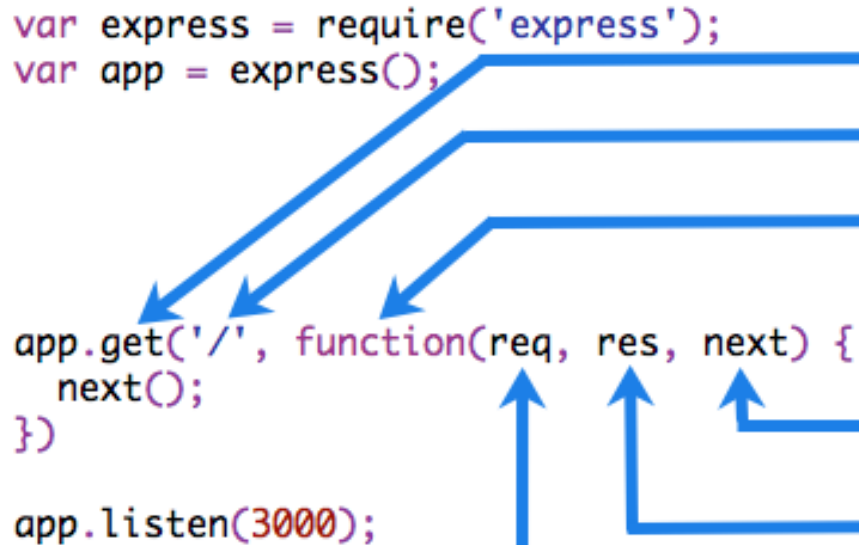
Executam diversas tarefas

- Execução de código
- Modificam os objetos de requisição e/ou resposta
- Finalizam o ciclo de requisição e resposta
- Chamam a próxima função de *middleware* da pilha

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

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

app.listen(3000);
```



- Método HTTP
 - Caminho (rota)
 - Função de MW
-
- `next` *callback*
 - HTTP *response* (para o cliente)
 - HTTP *request* (vinda do cliente)

Exemplo

```
1  const express = require('express')
2  const app = express()
3
4  const myLogger = function (req, res, next) {
5    console.log('LOGGED')
6    next()
7  }
8
9  app.use(myLogger)
10
11 app.get('/', (req, res) => {
12   res.send('Hello World!')
13 })
```

Exemplo

```
1  const express = require('express')
2  const app = express()
3
4  const requestTime = function (req, res, next) {
5    req.requestTime = Date.now()
6    next()
7  }
8
9  app.use(requestTime)
10
11 app.get('/', (req, res) => {
12   let responseText = 'Hello World!<br>'
13   responseText += `<small>Requested at: ${req.requestTime}</small>`
14   res.send(responseText)
```


Tipos de *Middlewares*

- MW de aplicação (*apllication level*)
- MW de roteamento (*router level*)
- MW de tratamento de erro
- *Embedded*

Application Level MW

- Associadas a uma instância de app
 - `app.use()`
 - `app.METHOD()`

Router Level MW

- Funcionamento similar aos MW de aplicação
- Rodam em uma instância `express.router()` ao invés de `express.app()`
- Podem ser usadas através de `router.use()` ou `router.METHOD()`

Exemplo

```
1  const express = require('express')
2  const app = express()
3  const router = express.Router()
4
5  // simple logger for this router's requests
6  // all requests to this router will first hit this middleware
7  router.use((req, res, next) => {
8    console.log('%s %s %s', req.method, req.url, req.path)
9    next()
10 })
11
12 // this will only be invoked if the path starts with /bar from the m
13 router.use('/bar', (req, res, next) => {
14   // ... maybe some additional /bar logging ...
```

MW de Tratamento de Erros

Similar aos demais, porém com 4 argumentos

```
1 app.use((err, req, res, next) => {  
2   console.error(err.stack)  
3   res.status(500).send('Something broke!')  
4 })
```

Embedded

Middlewares incluídos no Express

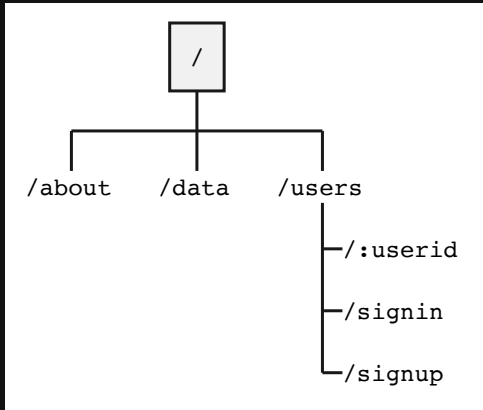
- `express.static`
 - Serve arquivos estáticos (imagens, CSS, *scripts*)
- `express.json`
 - Serve arquivos JSON
- `express.urlencoded`
 - Trata requisições com URLs codificadas

Perguntas

Exercício

Atividade Express

Criar uma aplicação utilizando o *framework* Express que contenha a estrutura mínima mostrada no diagrama abaixo. Cada página atende com requisições GET. A rota `/data` atende com requisição POST.



Atividade Express (Requisitos)

1. Criar uma função de Middleware para cada uma das rotas mostradas exibindo apenas uma página com nome da rota
2. Criar uma função Middleware de aplicação que realiza o registro no console do acesso a cada página (ver exemplos anteriores nesta aula)
3. Em `signin`, criar uma rota (`/users/:userid`) que recebe o `userid` do usuário e exibe na página uma msg de boas vindas usando esse valor
4. Caso o usuário acesse sem `userid` é direcionado a página signup (pesquise como usar `res.redirect()`)

Atividade Express (Requisitos)

5. Qualquer outra página que não tenha a rota indicada é direcionada para página de erro 404 com link para o index (pesquise como usar `res.status()`)

Referências

- Express
- Express JS Tutorial

Prof. José Roberto Bezerra

jbroberto@ifce.edu.br