

Tecnologias e Programação de Sistemas de Informação

ROUTING & FILE HANDLING

Desenvolvimento Web - Back-End | David Jardim

Cofinanciado por:



API: A SET OF TOOLS FOR BUILDING A SOFTWARE APPLICATION.

Stands for 'Application Programming Interface'. On the web the tools are usually made available via a set of URLs which accept and send only data via HTTP and TCP/IP.

ROUTING

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

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

ENDPOINT: ONE URL IN A WEB API.

Sometimes that endpoint (URL) does multiple thing by making choices based on the HTTP request headers.

ROUTING

- **app** is an instance of express
- **METHOD** is an HTTP request method, in lowercase
- **PATH** is a path on the server
- **HANDLER** is the function executed when the route is matched

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

Route Paths

- Route paths, in combination with a request method, define the endpoints at which requests can be made
- Route paths can be strings, string patterns, or regular expressions

```
app.get('/', function (req, res) {  
  res.send('root')  
})
```

Route Parameters

- Route parameters are named URL segments that are used to capture the **values specified at their position in the URL**



```
{"name":"mahesh","password":"password1","profession":"teacher","id":1}
```


Route Parameters

- The captured values are populated in the **request.params** object, with the name of the route parameter specified in the path

```
app.get('/user/:id', function(request, response) {  
  var userId = request.params.id;  
  response.send(userId);  
});
```


Query Parameters

```
app.get('/search', (req, res, next) => {  
  res.send(req.query)  
})
```

On query strings, the URL we create does not include the params

GET



http://localhost:3000/search?name=David&age=27

Files and expressions

```
app.get('/random.text', function (req, res) {  
  res.send('random.text')  
})
```

Will match anything with an “a”

```
app.get(/a/, function (req, res) {  
  res.send('/a/')  
})
```

Response Header

- The status code is a 3-digit HTTP status code, like 404. The last argument, headers, are the response headers
- Optionally one can give a human-readable statusMessage as the second argument

```
const body = 'hello world';
response.writeHead(200, {
  'Content-Length': Buffer.byteLength(body),
  'Content-Type': 'text/plain' });
```

HTTP Response Codes

Code	Description	Meaning
200	OK	Standard success response.
201	Created	New resource created.
301	Moved permanently	Permanent redirect to new URI.
304	Not modified	Safe to use page stored in cache.
307	Temporary redirect	Use new URI now; try old later.
401	Unauthorized	Authentication failed.
403	Forbidden	Disallowed, auth will not help.
404	Not found	Resource was not found.
405	Method not allowed	Used GET when should use POST.
500	Internal server error	Internal server error.

Response Methods

Method	Description
<code>res.download()</code>	Prompt a file to be downloaded.
<code>res.end()</code>	End the response process.
<code>res.json()</code>	Send a JSON response.
<code>res.jsonp()</code>	Send a JSON response with JSONP support.
<code>res.redirect()</code>	Redirect a request.
<code>res.render()</code>	Render a view template.
<code>res.send()</code>	Send a response of various types.
<code>res.sendFile()</code>	Send a file as an octet stream.
<code>res.sendStatus()</code>	Set the response status code and send its string representation as the response body.

Express Application Generator

- Use the application generator tool, express-generator, to quickly create an application skeleton
- Install the application generator globally

```
PS C:\Users\David.JARDIM\Documents\AxiansLabs\ml-serving> npm install express-generator -global
C:\Users\David.JARDIM\AppData\Roaming\npm\express -> C:\Users\David.JARDIM\AppData\Roaming\npm\node_modules\express-generator\bin\express-cli.js
+ express-generator@4.16.1
updated 1 package in 0.989s
```

Express Application Generator

```
$ express --view=pug myapp
```

```
create : myapp
create : myapp/package.json
create : myapp/app.js
create : myapp/public
create : myapp/public/javascripts
create : myapp/public/images
create : myapp/routes
create : myapp/routes/index.js
create : myapp/routes/users.js
create : myapp/public/stylesheets
create : myapp/public/stylesheets/style.css
create : myapp/views
create : myapp/views/index.pug
create : myapp/views/layout.pug
create : myapp/views/error.pug
create : myapp/bin
create : myapp/bin/www
```


Express Application Generator

```
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── error.pug
    ├── index.pug
    └── layout.pug
```

Navigate to the app folder:

```
$ cd myapp
```

Then install dependencies from the package.json:

```
$ npm install
```

Run the application:

```
$ npm start
```



CTeSP

CURSOS TÉCNICOS
SUPERIORES PROFISSIONAIS