

HTTP2 - Dados binários, GZIP ativo e TLS

Transcrição

Até agora sempre usamos o browser para realizar uma requisição. Mas podemos realizar fora dele usando a linha de comando por exemplo. Um programa famoso para isso é o **CURL**. No Linux e MacOS ele já vem instalado por padrão.

Caso esteja usando o Windows é necessário a instalação dele. O download deve ser feito por aqui:

<https://curl.haxx.se/download.html> (<https://curl.haxx.se/download.html>)

Para realizar e depurar uma requisição via CURL podemos simplesmente executar no terminal o seguinte comando:

```
curl -v www.caelum.com.br
```

Uma saída típica dele seria:

```
Fabios-MacBook-Pro:~ fabiopimentel$ curl -v www.caelum.com.br
* Rebuilt URL to: www.caelum.com.br/
*   Trying 172.217.29.51...
* Connected to www.caelum.com.br (172.217.29.51) port 80 (#0)
> GET / HTTP/1.1
> Host: www.caelum.com.br
> User-Agent: curl/7.49.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Vary: Accept-Encoding,User-Agent
< Content-Language: pt-br
< Content-Type: text/html; charset=UTF-8
< X-DNS-Prefetch-Control: on
< X-Cloud-Trace-Context: 3e5e270ee3ab1e79f81b10d2cdef53cd
< Date: Fri, 24 Mar 2017 19:20:12 GMT
< Server: Google Frontend
< Content-Length: 95776
<
    <!DOCTYPE html>
    <html class="no-js"lang="pt-br"> <head> <title>Caelum | Cursos de Java, .NET, Android, PHP,
    ...
```

Pode-se notar pela saída que temos logo no começo as informações do request efetuado:

```
> GET / HTTP/1.1
> Host: www.caelum.com.br
> User-Agent: curl/7.49.1
> Accept: */*
,
```

E após essas infos temos o cabeçalho da resposta obtida pelo servidor:

```
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Vary: Accept-Encoding,User-Agent
< Content-Language: pt-br
< Content-Type: text/html; charset=UTF-8
< X-DNS-Prefetch-Control: on
< X-Cloud-Trace-Context: 3e5e270ee3ab1e79f81b10d2cdef53cd
< Date: Fri, 24 Mar 2017 19:20:12 GMT
< Server: Google Frontend
< Content-Length: 95776
```

Logo depois vem o corpo da resposta (HTML da página requisitada):

```
<!DOCTYPE html> <html class="no-js"lang="pt-br"> <head> <title>Caelum | Cursos de Java, .NET, /
```

Em resumo o output apresentando pelo CURL possui essa divisão:

```
MBP-de-Fabio:~ fabiopimentel$ curl -v www.caelum.com.br
* Rebuilt URL to: www.caelum.com.br/
* Trying 216.58.222.115...
* TCP_NODELAY set
* Connected to www.caelum.com.br (216.58.222.115) port 80 (#0)

GET / HTTP/1.1
Host: www.caelum.com.br
User-Agent: curl/7.51.0
Accept: */*
REQUEST

...ESPERA...

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding,User-Agent
Content-Language: pt-br
Content-Type: text/html; charset=UTF-8
X-DNS-Prefetch-Control: on
X-Cloud-Trace-Context: d58c6460837073c01e35848bac3d3848
Date: Mon, 20 Feb 2017 18:32:33 GMT
Server: Google Frontend
Content-Length: 94230
RESPONSE

<!DOCTYPE html> <html class="no-js"lang="pt-br"> <head> <title>Caelum </title>
```

HTTP/2

O protocolo que estamos trabalhando até agora foi especificado na década de 90 e de lá até hoje muitas alterações foram feitas até na forma como usamos a internet.

Com a chegada do mundo mobile novas preocupações apareceram e otimizações são cada vez mais necessárias para uma boa performance. Por isso uma mudança foi necessária e em 2015 depois de alguns anos de especificações e reuniões surgiu a versão 2 desse protocolo.

A nova versão é batizada de **HTTP/2** e tem como página principal de documentação e referência essa:

<https://http2.github.io/>.

A nova versão do protocolo HTTP traz mudanças fundamentais para a Web. Recursos fantásticos que vão melhorar muito a performance da Web além de simplificar a vida dos desenvolvedores.

No HTTP 1.1, para melhorar a performance, habilitamos o **GZIP** no servidor para comprimir os dados das respostas. É uma excelente prática, mas que precisa ser habilitada explicitamente. No HTTP/2, o **GZIP é padrão e obrigatório**.

```
GET / HTTP/1.1
Host: www.caelum.com.br
User-Agent: curl/7.51.0
Accept-Encoding: gzip

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Vary: Accept-Encoding,User-Agent
Content-Language: pt-br
Content-Type: text/html; charset=UTF-8
Date: Mon, 20 Feb 2017 18:32:33 GMT
Server: Google Frontend
Content-Encoding: gzip
```

HTTP/2

GZIP

H4sIAAAAAAAAA/12Q3U4CMRBGX2XsdRfj/S4JQ0wXo1S0GC+HdmDH9GdtZ+HGL/FZfDG7gEG8mnbyzcmXA/XN/ct0/bac
h16h0qk1S04DQ1iEsjsZTJNd7+IRpn3LMYAkecY8aRs+ztYZJsCmy1bCclZWsT0q9hvl68aRhu1rBKVzW3AnUtyckQ01.
e20jaW9myo0n40BxZGV2Wdjpq7oeUfxDYmj1JZEjLCMVxQQo66NgZqQvx3lGhLKV6GhGPiHYcSuy5oKR/7X4EncBYj5KI
405YmYRN+wIPIHKGRIXckmFbICGCP2TQRg0merBsv7+EDZZX6UzBMFosBA5ZU18xUYaPHh1vebjKIzX+Acdf8vD3AQAA

Já na nova versão, os headers passam a ser binários:

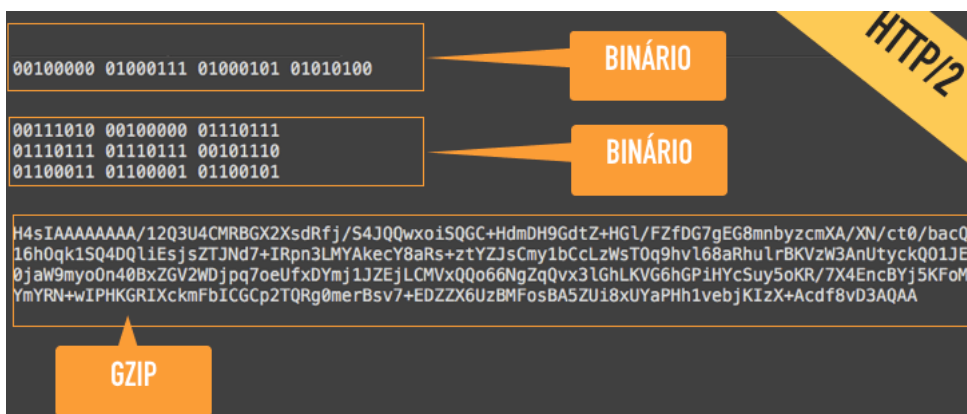


Diagram illustrating the compression of an HTTP GET request using various methods.

The original request (shown in a large box) is:

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cache-Control: max-age=0

```

The diagram shows three different compression methods being applied to the request:

- BINÁRIO + HPACK** (Binary + HPACK): This method compresses the request into a binary format using the HPACK algorithm.
- BINÁRIO + HPACK** (Binary + HPACK): This method compresses the request into a binary format using the HPACK algorithm.
- GZIP**: This method compresses the request using the GZIP algorithm.

The diagram also shows the original request being sent over the network, with the compressed versions being sent over the network.

3/4

