

HTTP2 - Cabeçalhos Stateful

Transcrição

Agora, queremos representar uma requisição. No código abaixo, estamos fazendo uma requisição através do método GET , que já conhecemos. Essa requisição está sendo feita para a raiz, bem parecido com o que fizemos no CURL no vídeo anterior.

```
GET /

Host: www.caelum.com.br
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:34.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
```

Quando realizamos uma requisição para essa URL via Firefox, por exemplo, o navegador envia alguns cabeçalhos que são padrões. Por exemplo, no cabeçalho Host , temos o domínio para onde estamos realizando essa requisição, que é www.caelum.com.br . Como estamos realizando um GET para / , o path para onde estamos realizando a requisição é www.caelum.com.br/ .

Além disso, vemos uma informação de User-Agent , que é basicamente a fonte dessa requisição, neste caso é o navegador, Mozilla. Quando realizarmos uma requisição pelo CURL , aparecerá CURL , e se for num Safari, Chrome, qualquer outro navegador, irá aparecer a informação de onde estamos realizando de fato a requisição. Ou seja, nesse cabeçalho a gente especifica quem é o usuário.

Nele também é dito que é aceito, por padrão, o HTML, na linguagem tanto português quanto inglês, e que estamos aceitando uma codificação GZIP , já que no HTTP1 podemos especificar que tipo de compressão nossa requisição está aceitando.

Precisamos repetir os cabeçalhos enviados em uma requisição anterior?

Agora vamos realizar uma outra requisição, mas dessa vez para o arquivo principal.js. Então, quando a requisição para página principal foi feita, provavelmente recebemos um HTML, e desse HTML foi necessário realizar uma outra requisição, porque era um recurso importante para a página ser exibida, como por exemplo um arquivo JavaScript.

```
GET /principal.js

Host: www.caelum.com.br
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:34.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
```

Então, na próxima requisição teremos que enviar novos parâmetros, novos dados. Então, mais uma vez, é uma nova requisição, agora para /principal.js , um recurso do nosso sistema. Qual o Host ? www.caelum.com.br . Qual o Agent ? Mozilla, versão 5.0. O que é aceito? qual linguagem é aceita? que tipo de compressão estamos aceitando?

Podemos perceber que o que colocamos nessa segunda requisição é exatamente o mesmo que fizemos na primeira? Os mesmos dados estão sendo trafegados nas duas requisições. Seria ótimo se só trafegássemos isso uma única vez, pois se estamos enviando mais dados, oneramos ainda mais nosso usuário, usando mais banda, deixando essa requisição mais lenta.

Seria muito bom se só pudéssemos ou só tivéssemos que enviar uma única vez, e é exatamente isso que o HTTP2 faz. A partir do HTTP2, não precisamos mais repetir os **Headers**, os cabeçalhos que já enviamos em uma requisição anterior. Logo, quando fazemos uma requisição para o **principal.js**, onde teríamos os cabeçalhos exatamente iguais aos da requisição passada, nós não precisamos enviar novamente esses dados.

Cabeçalhos diferentes

Agora, se temos uma imagem, os cabeçalhos podem mudar, por exemplo, o **Host**, que pode estar especificado na página principal. Logo, na primeira requisição, o conteúdo HTML especificou que tem que buscar uma imagem do Host, que é **image.caelum.com.br**, um subdomínio dentro da nossa aplicação. Então, esse cabeçalho terá que ser alterado, logo enviaremos apenas os cabeçalhos que são diferentes.

Isso está especificado no HTTP2, para que uma requisição fique mais leve e não onere tanto o usuário. Isso é conhecido como **Headers Stateful**.

No início do curso, comentamos que o HTTP era *stateless*, ou seja, ele não guarda informações das requisições passadas. E isso continua valendo, mas no caso dos cabeçalhos, existe um ambiente que guarda estado.