Guia Completo de Atributos HTTP

1. Estrutura de uma Requisição HTTP

Exemplo de requisição HTTP com todos os atributos:

<Method> <URL> <Version>

<Header1>: <Value1>

<Header2>: <Value2>

...

<HeaderN>: <ValueN>

<Body>

- Método: Indica o tipo de ação a ser executada.
- URL: O endpoint que o cliente quer acessar.
- Versão: A versão do protocolo HTTP (por exemplo, HTTP/1.1, HTTP/2).
- Cabeçalhos (Headers): Contêm metadados sobre a requisição.
- Corpo (Body): Contém os dados da requisição (se aplicável).

2. Métodos HTTP

1. GET

Objetivo: Recuperar dados do servidor.

Exemplo:

GET /api/users HTTP/1.1

Host: example.com

Accept: application/json

Quando usar: Quando você precisa buscar informações de um recurso.

2. POST

Objetivo: Enviar dados para o servidor (geralmente para criação de um novo recurso).

Exemplo:

POST /api/users HTTP/1.1

Host: example.com

Content-Type: application/json

Authorization: Bearer <token>

{

"name": "John Doe",

"email": "john@example.com"

}

Quando usar: Quando você precisa criar ou enviar dados para o servidor.

3. PUT

Objetivo: Atualizar um recurso completo no servidor.

Exemplo:

PUT /api/users/123 HTTP/1.1

Host: example.com

Content-Type: application/json

{

"name": "John Updated",

"email": "johnupdated@example.com"

}

Quando usar: Quando você deseja substituir completamente um recurso existente.

4. PATCH

Objetivo: Atualizar parcialmente um recurso no servidor.

Exemplo:

PATCH /api/users/123 HTTP/1.1

Host: example.com

Content-Type: application/json

{

"email": "newemail@example.com"

}

Quando usar: Quando você precisa modificar apenas algumas partes de um recurso.

5. DELETE

Objetivo: Excluir um recurso do servidor.

Exemplo:

DELETE /api/users/123 HTTP/1.1

Host: example.com

Authorization: Bearer <token>

Quando usar: Quando você deseja excluir um recurso.

6. HEAD

Objetivo: Recuperar apenas os cabeçalhos da resposta (sem o corpo).

Exemplo:

HEAD /api/users HTTP/1.1

Host: example.com

Quando usar: Para verificar metadados sobre um recurso sem transferir o corpo da resposta.

7. OPTIONS

Objetivo: Recuperar os métodos HTTP suportados por um servidor para um recurso específico.

Exemplo:

OPTIONS /api/users HTTP/1.1

Host: example.com

Quando usar: Para obter as opções de comunicação com o servidor (geralmente usado para verificação de CORS).

8. TRACE

Objetivo: Usado para diagnosticar a comunicação, retornando o conteúdo da requisição HTTP como está sendo recebida pelo servidor.

Exemplo:

TRACE / HTTP/1.1

Host: example.com

Quando usar: Para fins de depuração e diagnóstico.

3. Cabeçalhos HTTP (Headers)

1. Host

Descrição: Indica o nome do servidor e o número da porta do recurso.

Exemplo:

Host: example.com

2. Content-Type

Descrição: Especifica o tipo de mídia do corpo da requisição ou resposta.

Exemplo:

Content-Type: application/json

3. Authorization

Descrição: Usado para enviar credenciais de autenticação (por exemplo, tokens de API).

Exemplo:

Authorization: Bearer <token>

4. Accept

Descrição: Especifica o tipo de mídia que o cliente está disposto a aceitar na resposta.

Exemplo:

Accept: application/json

5. User-Agent

Descrição: Contém informações sobre o cliente (navegador ou cliente HTTP) que está fazendo a requisição.

Exemplo:

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36

6. Content-Length

Descrição: Indica o tamanho do corpo da requisição ou resposta.

Exemplo:

Content-Length: 1234

7. Cache-Control

Descrição: Controla como os caches (do cliente ou intermediários) devem armazenar a resposta.

Exemplo:

Cache-Control: no-cache

8. Cookie

Descrição: Envia cookies do cliente para o servidor.

Exemplo:

Cookie: sessionId=123456; loggedIn=true

4. Corpo da Requisição (Request Body)

O corpo da requisição é a parte onde você envia dados para o servidor. O formato dos dados no corpo depende do valor do cabeçalho Content-Type.

Exemplos de corpo em diferentes tipos de Content-Type:

1. JSON (application/json)

{

```
  "name": "John Doe",
  "email": "john.doe@example.com"
}
```

2. Formulário (application/x-www-form-urlencoded)

```
name=John+Doe&email=john.doe%40example.com
```

3. Formulário multipart (para upload de arquivos)

```
--boundary
Content-Disposition: form-data; name="file"; filename="example.jpg"
Content-Type: image/jpeg

<binary data>
--boundary--
```

5. Exemplo Completo de Requisição HTTP (POST)

```
POST /api/users HTTP/1.1
Host: example.com
Authorization: Bearer <token>
Content-Type: application/json
Accept: application/json
User-Agent: Mozilla/5.0

{
  "name": "John Doe",
  "email": "john.doe@example.com"
```

}

6. Códigos de Status HTTP

Além dos métodos e cabeçalhos, os servidores respondem com códigos de status que indicam o resultado da requisição:

- 2xx: Sucesso (ex. 200 OK, 201 Created)
- 3xx: Redirecionamento (ex. 301 Moved Permanently, 302 Found)
- 4xx: Erro do cliente (ex. 400 Bad Request, 401 Unauthorized)
- 5xx: Erro do servidor (ex. 500 Internal Server Error, 503 Service Unavailable)