

Introdução à Coleta de Dados na Web



- Aula 03 -
Coleta, Preparação e
Análise de Dados

Prof. Me. Lucas R. C. Pessutto



PUCRS

Pontifícia Universidade Católica
do Rio Grande do Sul



Slides adaptados do material do Prof. Lucas Silveira
Kupssinskü e do Prof. Luan Fonseca Garcia

Definição de Web Scraping

“A web scraper **accesses** web pages, **finds** specified data elements on the page, **extracts** them, **transforms** them if necessary, and finally **saves** these data as a structured data set.”

Geoff Boeing e Paul Waddell

- Coletar dados por qualquer meio que não seja um programa interagindo com uma API (ou um ser humano interagindo...)
- Em geral, um programa automatizado que consulta um servidor web, requisita dados e faz o parsing desses dados para extrair informações necessárias.

Spider vs Crawling vs Scraping

- *Crawler* (ou Spider)
 - Programa que navega por uma página ou um conjunto de páginas seguindo os links e baixando páginas inteiras;
 - Também chamado de spider bot porque vai procurando links dentro de links e acaba formando uma “teia”.
- Scraper
 - Programa que faz *parsing* do conteúdo da página web e extrai informações que são armazenadas para um determinado uso.

Ferramentas Disponíveis

- Selenium
- Scrapy
- BeautifulSoup
- Requests

	Scrapy	Requests	Beautiful Soup	Selenium
What is it?	Web scraping framework	Library	Library	Library
Purpose	Complete web scraping solution	Simplifies making HTTP requests	Data parser	Scriptable web browser to render javascript
Ideal use case	Development of recurring or large scale web scraping projects	Simple non-recurring web scraping tasks	Simple non-recurring web scraping tasks	Small-scale web scraping of javascript heavy websites
Built-in Data Storage Supports	JSON, JSON lines, XML, CSV	Need to develop your own	Need to develop your own	Customizable
Available selectors	JCSS & Xpath	N/A	CSS	CSS & Xpath
Asynchronous	Yes	No	No	No
Javascript support	Yes, via Splash library	N/A	No	Yes
Documentation	Excellent	Excellent	Excellent	Good
Learning curve	Easy	Very easy	Very easy	Easy
Ecosystem	Large ecosystem of developers contributing projects and support on Github and StackOverflow	Few related projects or plugins	Few related projects or plugins	Few related projects or plugins
Github stars	32,690	34,727	-	14,262

Fonte: <https://www.zyte.com/learn/what-python-web-scraping-tools-are-available/>

Quando *web scraping* é útil?

- Quero fazer algum experimento/produto com ML e não tenho um conjunto de dados disponível.
 - Web é uma fonte rica de:
 - Imagens
 - Textos (multi-idioma)
 - Vídeos
 - Dados estruturados
- Monitorar preços de mercado.
- Monitorar atualizações em serviços online.

Quando *web scraping* é útil?

Lembre-se:

- Em situações ideais *web scraping* não deveria ser necessário.
- *Front End* da web é feito para renderização em navegadores, não para distribuição de dados
- APIs REST fornecem acesso a dados de uma forma mais elegante.
 - HTTP
 - JSON

Por que não usar APIs?

- Uma API é projetada para fornecer uma stream de dados bem **formatados** e de forma **conveniente**.
- APIs são ótimas... Se existir uma que atenda ao que precisamos.
- Talvez não seja tão conveniente usar uma.
 - São poucos dados e não existe uma API que disponibilize.
 - Os dados estão espalhados em diversos sites e não há uma única API coesa.

Questões legais

Sem open banking, consolidadores de carteiras buscam alternativas para capturar dados.

Quarta, 20 Janeiro 2021

Cade investiga Bradesco por dificultar acesso a dados pelo GuiaBolso

Por Wagner Wakka | 03 de Maio de 2019 às 10h53

Bradesco faz acordo com Cade para encerrar investigação sobre práticas anticompetitivas contra GuiaBolso

Pelo acordo, o banco vai pagar cerca de R\$ 23,8 milhões e parar de dificultar acesso a dados pedidos por clientes para uso pelo aplicativo da fintech.



Por Reuters

07/10/2020 17h52 · Atualizado há um ano



Questões legais

- Permissões e restrições de *web scraping* ainda não são bem estabelecidas.
- Usualmente, capturar informações públicas para “fair use”, uso pessoal e privado não acarretam problemas.
- Atenção redobrada ao trabalhar com dados pessoais.
- LGPD – Dados pessoais http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm

Questões Legais

- A primeira multa da Autoridade Nacional de Proteção de Dados, de R\$ 14,4 mil, foi para uma empresa de telemarketing pela oferta de disparos em massa via celular e Whatsapp para até 130 milhões de pessoas.



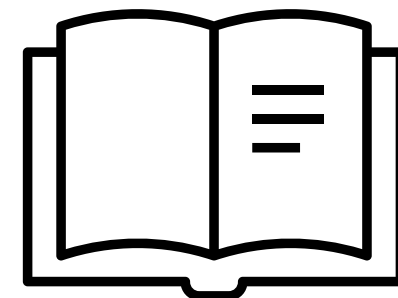
Única multa da ANPD alerta que raspagem de dados é ilegal



Luís Osvaldo Grossmann ... 25/07/2023 ... Convergência Digital

Como fazemos quando precisamos pesquisar algo em um livro físico?

1. Sabendo o endereço da biblioteca, nos deslocamos até ela
2. Procuramos o livro nas estantes pelo seu código
3. Pegamos o livro e levamos a algum local para leitura
4. Realizamos a busca de fato
5. Anotamos o que encontramos de relevante na busca

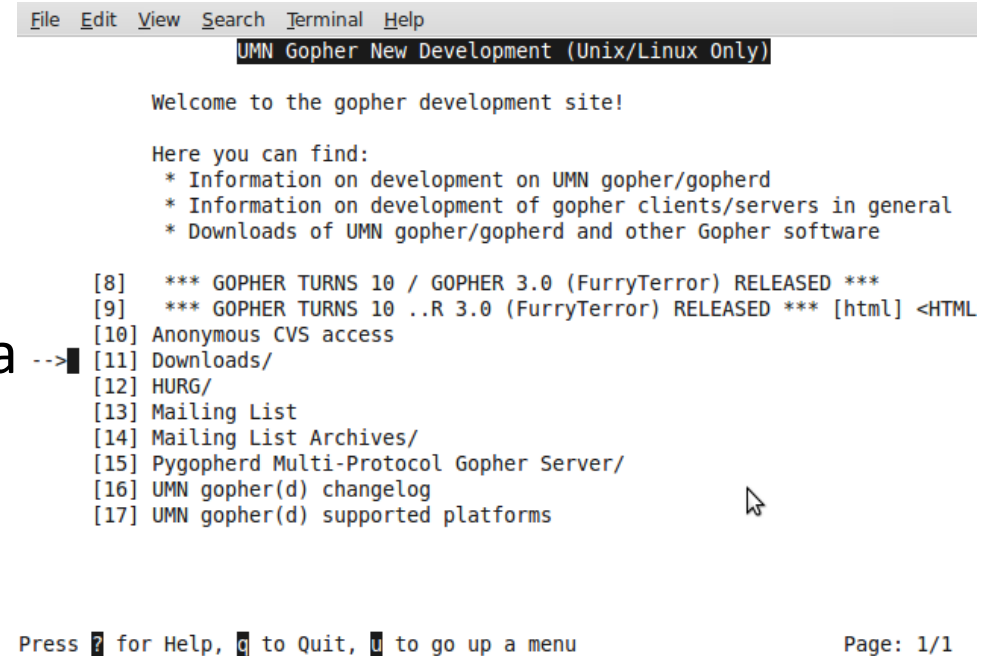


O que precisamos saber para fazer scraping?

- Precisamos ter uma boa noção de como a *web* funciona e quais protocolos e linguagens são utilizadas.
 - Como nos “deslocar” até a biblioteca e encontrar o “livro” que queremos
 - Protocolo HTTP, URIs
 - Como levar o “livro” a algum lugar para ler
 - Fazer crawling para baixar a página (bibliotecas urllib, requests, etc)
 - Como saber onde “procurar” dentro do livro
 - HTML, CSS, DOM e REGEX para saber onde e como procurar a informação
 - Ferramentas para fazer o scraping de fato (BeautifulSoup, Selenium...)
 - Como “anotar” as informações encontradas
 - Armazenar o conteúdo extraído em algum arquivo ou banco de dados

Para pensar...

- A Internet surgiu em 1969.
- O protocolo HTTP, que permite que a gente transita informações através de páginas HTML só foi desenvolvido nos anos 1990.



```
File Edit View Search Terminal Help
UMN Gopher New Development (Unix/Linux Only)

Welcome to the gopher development site!

Here you can find:
* Information on development on UMN gopher/gopherd
* Information on development of gopher clients/servers in general
* Downloads of UMN gopher/gopherd and other Gopher software

[8] *** GOPHER TURNS 10 / GOPHER 3.0 (FurryTerror) RELEASED ***
[9] *** GOPHER TURNS 10 ..R 3.0 (FurryTerror) RELEASED *** [html] <HTML
[10] Anonymous CVS access
[11] Downloads/
[12] HURG/
[13] Mailing List
[14] Mailing List Archives/
[15] Pygopherd Multi-Protocol Gopher Server/
[16] UMN gopher(d) changelog
[17] UMN gopher(d) supported platforms

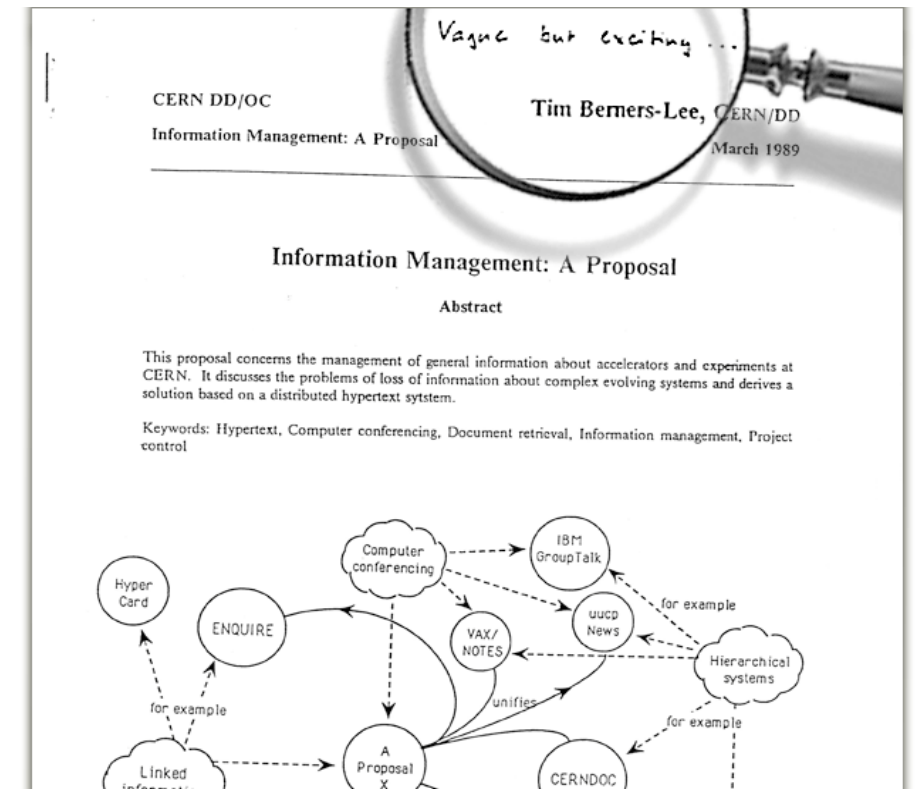
Press ? for Help, q to Quit, u to go up a menu
```

- Como era a Internet antes do HTML????
 - “The pre-Web Internet was an almost entirely text-based world. There were ASCII-based end-user programs such as [gopher](https://www.zdnet.com/article/before-the-web-the-internet-in-1991/), which let you use a menu to search through organized collections of files.”

<https://www.zdnet.com/article/before-the-web-the-internet-in-1991/>

História da Web

- Problema:
 - Informação em vários computadores.
 - Acesso local nestes computadores.
- Tim Berners-Lee propôs a web em 1989:
 - “Information Management: A Proposal”.
- Três tecnologias fundamentais:
 - HTML
 - URI
 - HTTP

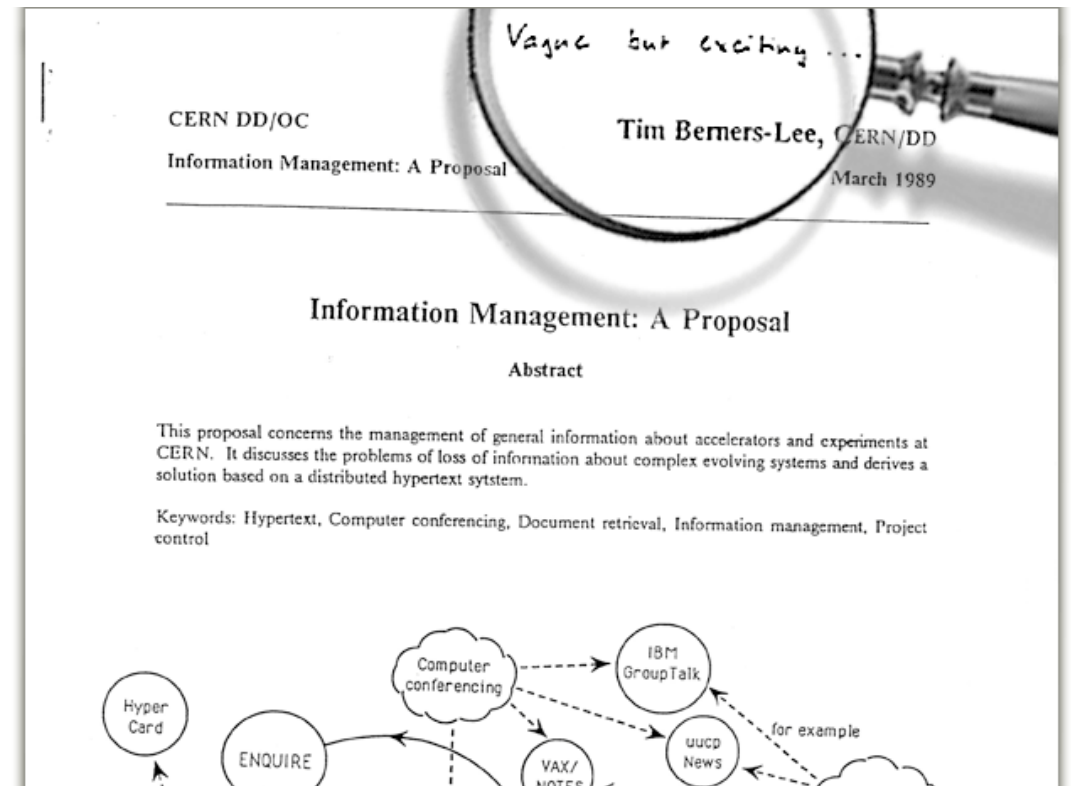


Um pouco de história...

“Creating the web was really an act of desperation, because the situation without it was very difficult when I was working at CERN later. Most of the technology involved in the web, like the hypertext, like the Internet, multifold text objects, had all been designed already. **I just had to put them together**. It was a step of generalising, going to a higher level of abstraction, thinking about all the documentation systems out there as being possibly part of a larger imaginary documentation system.” Tim Berners-Lee

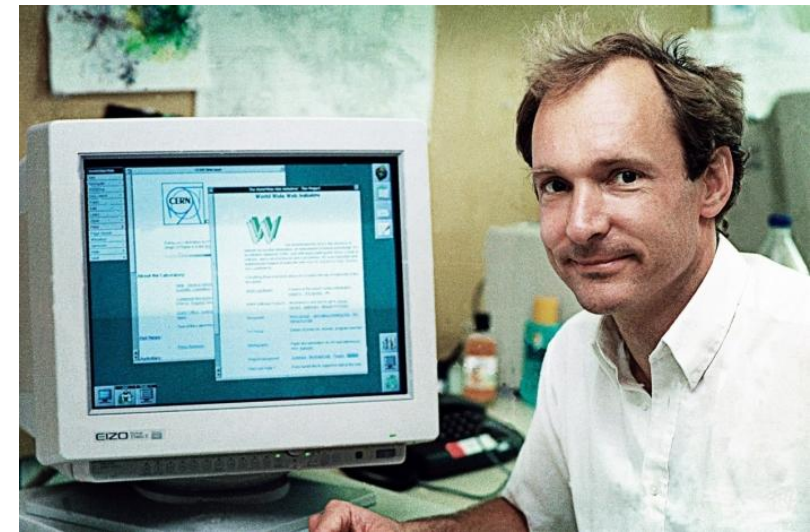


Foto de exposição no CERN em 2005



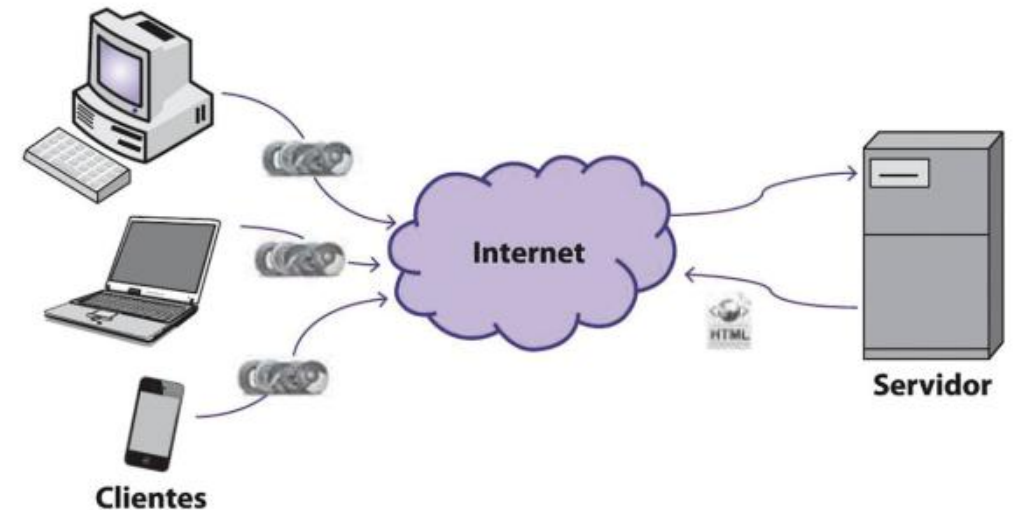
Um pouco de história...

- Tim Berners-Lee é o criador da web, porém a internet já existia antes!
- Enquanto a internet diz respeito a rede de computadores, a www (world wide web) necessita dessa rede para ser útil.
- Não confunda os dois conceitos!
- Em 1993 o CERN anuncia que o código e as tecnologias criadas para a web poderiam ser usadas sem nenhum tipo de restrição (sem pagamento de royalties).
 - Segundo Lee, essa decisão foi vital para o sucesso da web.



World Wide Web

- É um sistema de informação.
- Recursos compartilhados na rede são recursos web.
- Recursos web possuem identificadores únicos = URI.
- Recursos podem ser interligados = hiperlinks.
- Recursos acessíveis via internet = HTTP.



HyperText Markup Language

A linguagem básica para páginas web

HTML

- HyperText Markup Language
 - Linguagem padrão para escrita de documentos criados para serem visualizados em um **web browser**.
- Ela é composta de **Elementos** demarcados por **tags** que ajudam a **estruturar** o conteúdo de uma página web em uma estrutura de **árvore**.
- É muito comum que seja utilizada em conjunto com outras tecnologias, como **CSS** ou **JavaScript**.

Hello World em HTML

Document Type Declaration

Tag de abertura

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <div>
      <p>Hello world!</p>
    </div>
  </body>
</html>
```

Tag de fechamento

Conteúdo

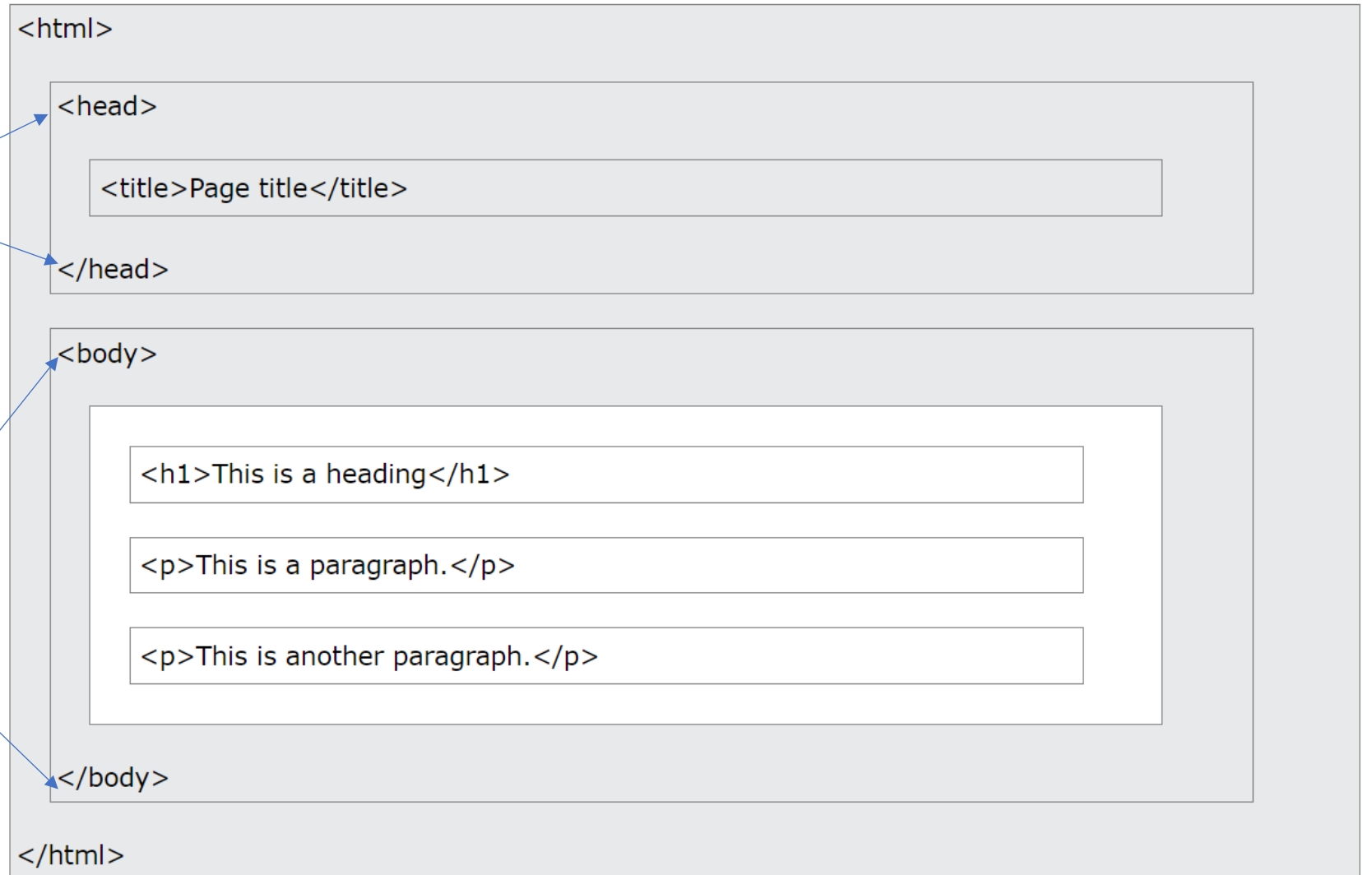
HTML Element

- Construto mais básico do HTML.
- São os elementos que permitem adicionar **semântica** e **formatação** ao conteúdo de uma página web.
 - Criar parágrafos
 - Formatar texto em **negrito**
 - Organizar texto em listas ou tabelas
 - Inserir *hyperlinks* ou imagens
- Elementos são demarcados através de pares de tags, que delimitam o início e o fim do elemento
 - Tags que iniciam elementos são representadas com **<nome>**, enquanto tags que terminam elementos são representadas com **</nome>**
 - **Alguns elemento não possuem tag de encerramento!**

HTML Element

Conteúdo que aparece na
barra de título do navegador
ou da tab

Conteúdo que aparece no
browser



HTML Element

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Título nível 1</h1>
    <h2>Título nível 2</h2>
    <h3>Título nível 3</h3>
    <h4>Título nível 4</h4>
    <h5>Título nível 5</h5>
    <h6>Título nível 6</h6>
  </body>
</html>
```



Título nível 1

Título nível 2

Título nível 3

Título nível 4

Título nível 5

Título nível 6

```
<!DOCTYPE html>
<html>
  <body>
    <h2>HTML Links</h2>
    <p>Links HTML são criados com a tag a</p>
    <a href="https://www.pucrs.com">Isso é um link</a>
  </body>
</html>
```



HTML Links

Links HTML são criados com a tag a

[Isso é um link](https://www.pucrs.com)

HTML Element

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Lista sem ordem definida</h2>
    <ul>
      <li>Café</li>
      <li>Chá</li>
      <li>Leite</li>
    </ul>
    <h2>Lista ordenada</h2>
    <ol>
      <li>Café</li>
      <li>Chá</li>
      <li>Leite</li>
    </ol>
  </body>
</html>
```



Lista sem ordem definida

- Café
- Chá
- Leite

Lista ordenada

1. Café
2. Chá
3. Leite

```
<html>
<body>
  <h2>Tabelas em HTML</h2>
  <p>Tabelas começam com a tag table.</p>
  <p>Linhas da tabela começam com a tag tr.</p>
  <p>Os dados da tabela usam a tag td.</p>
  <h2>Exemplo com 3 linhas e 3 colunas:</h2>
  <table>
    <tr>
      <td>100</td>
      <td>200</td>
      <td>300</td>
    </tr>
    <tr>
      <td>400</td>
      <td>500</td>
      <td>600</td>
    </tr>
    <tr>
      <td>700</td>
      <td>800</td>
      <td>900</td>
    </tr>
  </table>
</body>
</html>
```



Tabelas em HTML

Tabelas começam com a tag table.

Linhas da tabela começam com a tag tr.

Os dados da tabela usam a tag td.

Exemplo com 3 linhas e 3 colunas:

100	200	300
400	500	600
700	800	900

HTML Attribute

- Palavras-chave utilizadas dentro de tags de elementos para **controlar** o seu **comportamento**.
- Sempre ficam dentro das **tags de abertura**.

```
<element attribute="value">element content</element>
```

- Uso mais comum é para **estilizar** os elementos.

HTML Attribute

```
<!DOCTYPE html>
<html>
  <body>
    <p>Parágrafo normal</p>
    <p style="color:red;">Parágrafo vermelho</p>
    <p style="color:blue;">Parágrafo azul</p>
    <p style="font-size:50px;">Parágrafo maior</p>
  </body>
</html>
```

Parágrafo normal

Parágrafo vermelho

Parágrafo azul

Parágrafo maior

HTML Attribute

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p.error {
      color: red;
    }
  </style>
</head>
<body>
  <p>Parágrafo sem classe</p>
  <p class="error">Parágrafo com classe.</p>
  <p>Outro parágrafo</p>
  <p class="error">Parágrafo com classe.</p>
</body>
</html>
```



Parágrafo sem classe

Parágrafo com classe.

Outro parágrafo

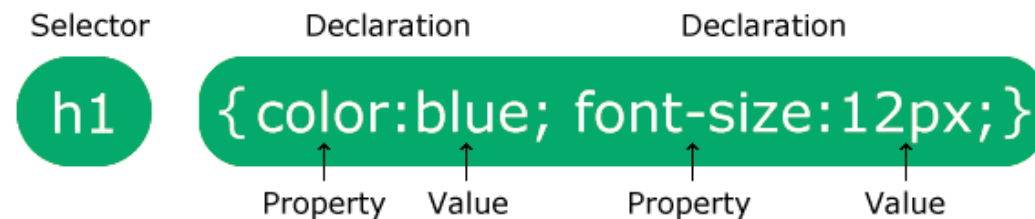
Parágrafo com classe.

Cascading Style Sheets

Um facilitador para estilizar páginas web

Cascading Style Sheets (CSS)

- Usado para formatar o **layout** de uma página web.
- Facilita o controle de diversos elementos ao mesmo tempo através de **regras de CSS**.



- CSS pode ser utilizado de três maneiras:
 - **Inline**: Diretamente na tag de abertura de um **element**
 - **Internamente**: Diretamente no cabeçalho da página html
 - **Externamente**: Estilo declarado em um arquivo **.css** que é carregado pela página html

Cascading Style Sheets (CSS)

- No uso *inline* de CSS, o estilo é definido diretamente no elemento através do atributo **style**

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;">Um título azul</h1>

<p style="color:red;">Um parágrafo vermelho.</p>

</body>
</html>
```



Um título azul

Um parágrafo vermelho.

Cascading Style Sheets (CSS)

- No uso **interno** de CSS, o estilo é definido dentro da seção **<head>** do documento html e pode ser utilizado na **página inteira**.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {background-color: powderblue;}
      h1   {color: blue;}
      p    {color: red;}
    </style>
  </head>
  <body>
    <h1>Este é um título h1.</h1>
    <p>Este é um parágrafo.</p>
  </body>
</html>
```



Este é um título h1.

Este é um parágrafo.

Cascading Style Sheets (CSS)

- No uso **externo** de CSS, o estilo é definido em um arquivo **.css** independente que é carregado na seção <head> e pode ser utilizado na **página inteira**.

```
body {  
  background-color: powderblue;  
}  
h1 {  
  color: blue;  
}  
p {  
  color: red;  
}
```

Arquivo styles.css



Este é um título h1.

Este é um parágrafo.

```
<!DOCTYPE html>  
<html>  
  <head>  
    <link rel="stylesheet" href="styles.css">  
  </head>  
  <body>  
    <h1>This is a heading</h1>  
    <p>This is a paragraph.</p>  
  </body>  
</html>
```

Cascading Style Sheets (CSS)

- **Seletores** na prática apontam para qual elemento HTML queremos definir o estilo.
- Podem ser definidos para todos elementos de um tipo, para uma classe, para um id e de muitas outras formas.

```
p {  
  text-align: center;  
  color: red;  
}
```

Estilo vale para todos
elementos do tipo
parágrafo

`<p>`

```
p.center {  
  text-align: center;  
  color: red;  
}
```

Estilo vale para todos
parágrafos com a
classe "center"

`<p class="center">`

```
.center {  
  text-align: center;  
  color: red;  
}
```

Estilo vale para
qualquer elemento
com com a classe
"center"

`<p class="center">`

`<h1 class="center">`

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

Estilo só vale para o
elemento com o id
único "para1"

`<p id="para1">`

Seletores

- Veremos nas próximas aulas que seletores são uma maneira fácil de encontrar um determinado conteúdo quando estamos fazendo scraping.
- Com as bibliotecas que vamos usar, podemos fazer buscas por elementos diretamente pelos seletores.

Uniform Resource Identifier

Identificadores na internet

Uniform Resource Identifier (URI)

- Sequência **única** de caracteres que identifica um recurso físico ou lógico nas tecnologias da web.
- URN: Universal Resource Name
 - Serve para identificar um determinado recurso em um contexto específico.
 - Exemplos:
 - ORCID: 0000-0003-2580-3996
 - ISBN: 8573937610
- URL: Universal Resource Locator
 - Identifica o “local” usado para interagir com o recurso.
 - Exemplos:
 - <https://www.pucrs.br/>
 - <mailto:john.doe@pucrs.com>

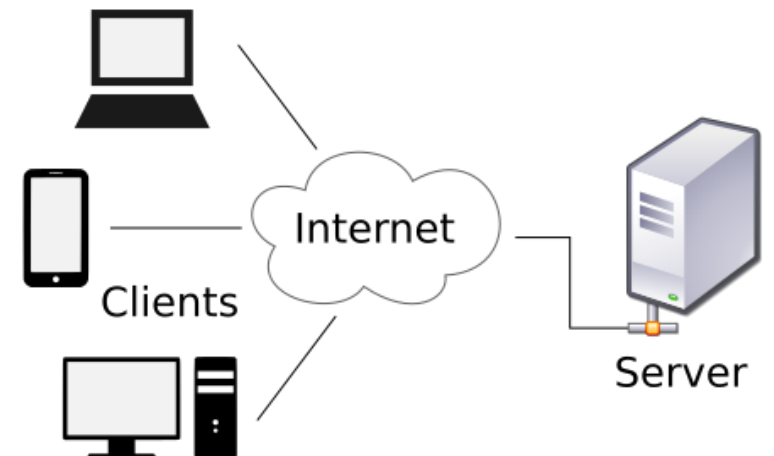
Curioso sobre outros URIs? https://en.wikipedia.org/wiki/List_of_URI_schemes

HyperText Transfer Protocol

O protocolo de aplicação que é a base da web

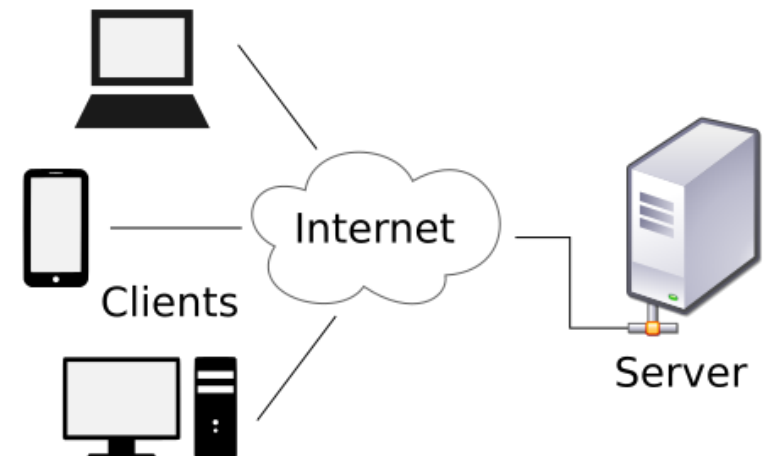
HyperText Transfer Protocol (HTTP)

- Protocolo para transferência de hipertexto e seus recursos em um paradigma **cliente-servidor**;
 - Hipertextos são documentos de texto acessíveis em computadores e que possuem referências (links) para outros documentos.
- O cliente envia requisições (*requests*);
- O servidor manda respostas (*responses*);



HyperText Transfer Protocol (HTTP)

- A comunicação sempre começa de um cliente (usualmente seu web browser) requisitando recursos de um servidor.
- Consiste da troca de mensagens individuais e não em uma comunicação contínua, embora atualmente utilize conexões persistentes.
- É stateless!
 - Servidor não preserva o estado da sessão de requisições anteriores.
 - Toda requisição é autocontida.



HTTP

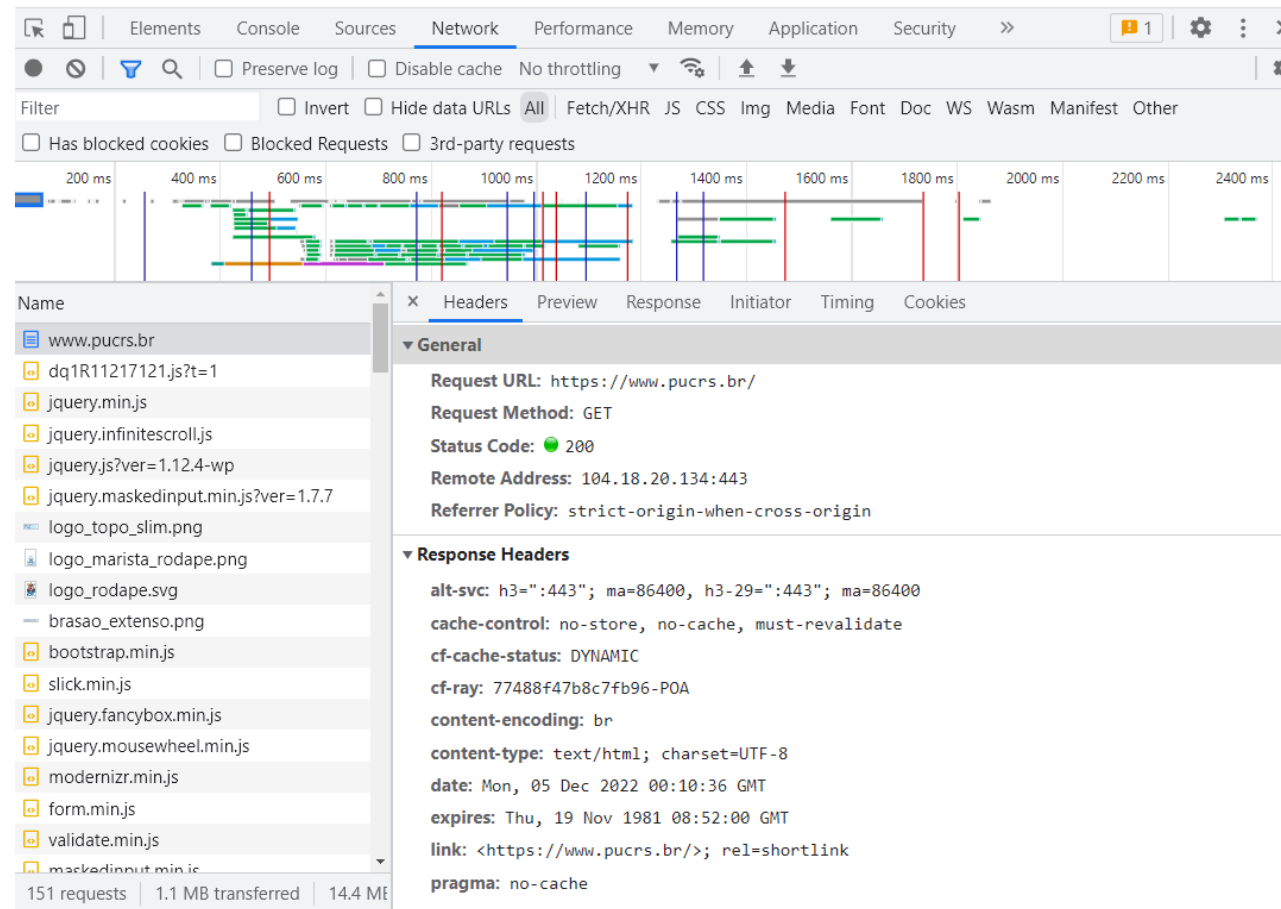
- Como funciona um processo de requisição/resposta em http:
 1. Cliente: Abre uma conexão TCP com o servidor, é nessa conexão que irão trafegar as requisições e as respostas;
 2. Cliente: Envia uma requisição HTTP (aqui no exemplo usaremos a versão HTTP 1.0 por ser mais fácil de interpretar).
 3. Servidor: Recebe a solicitação, processa e retorna uma resposta;
 4. Cliente fecha a conexão ou reutiliza para novas requisições:

```
GET / HTTP/1.1
User-Agent: PostmanRuntime/7.29.2
Accept: */*
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://www.pucrs.br/
Host: www.pucrs.br
```

```
HTTP/1.1 200 OK
Date: Sun, 04 Dec 2022 23:54:02 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/7.3.21
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Link: <https://www.pucrs.br/>; rel=shortlink
Vary: Accept-Encoding
X-Turbo-Charged-By: LiteSpeed
X-LiteSpeed-Cache: hit
CF-Cache-Status: DYNAMIC
Server-Timing: cf-q-config;dur=5.9999947552569e-06
Server: cloudflare
CF-RAY: 774877054817d098-POA
Content-Encoding: br
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400
```

HTTP – Como ver isso acontecendo?

- Podemos usar as ferramentas do desenvolvedor disponíveis no nosso navegador;
 - Exemplo usando o Google Chrome



HTTP – Como ver isso acontecendo?

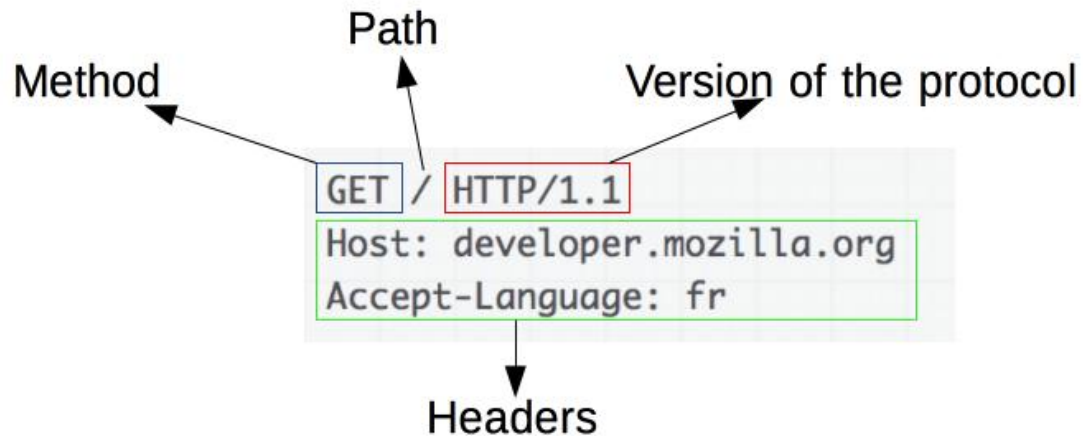
- Outra opção, podemos usar curl
 - `curl -ksvDL --request GET 'www.pucrs.br' -o /dev/null`

HTTP – Como ver isso acontecendo?

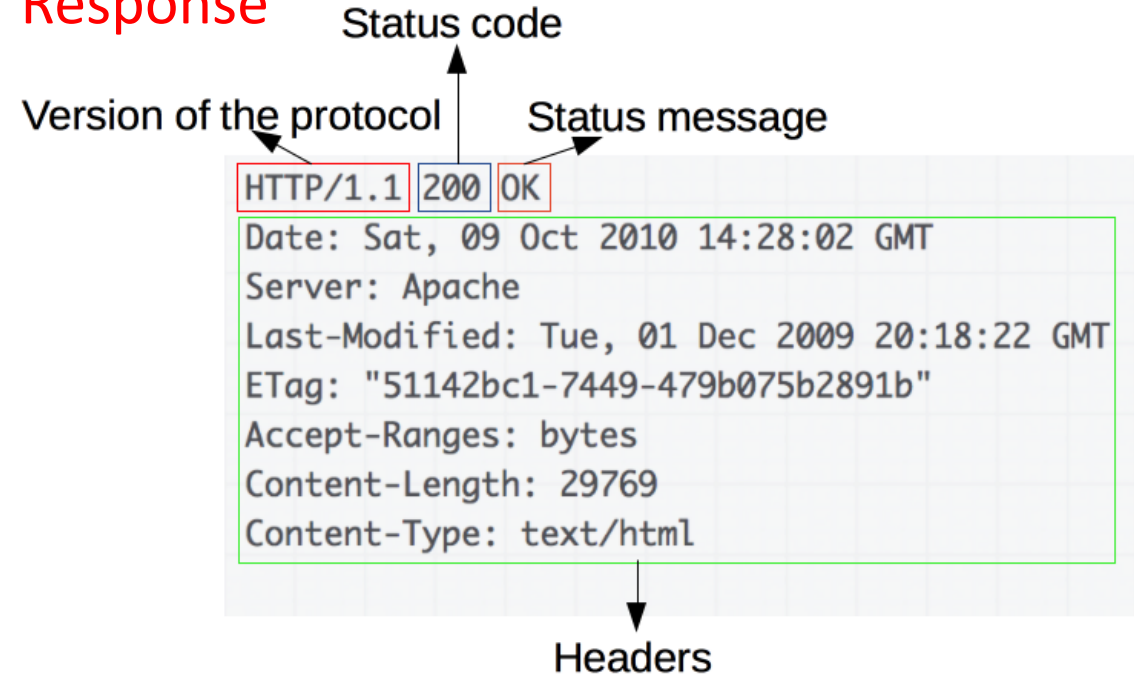
- Outra opção, podemos usar o postman
 - <https://www.postman.com/>

HTTP – Resumindo

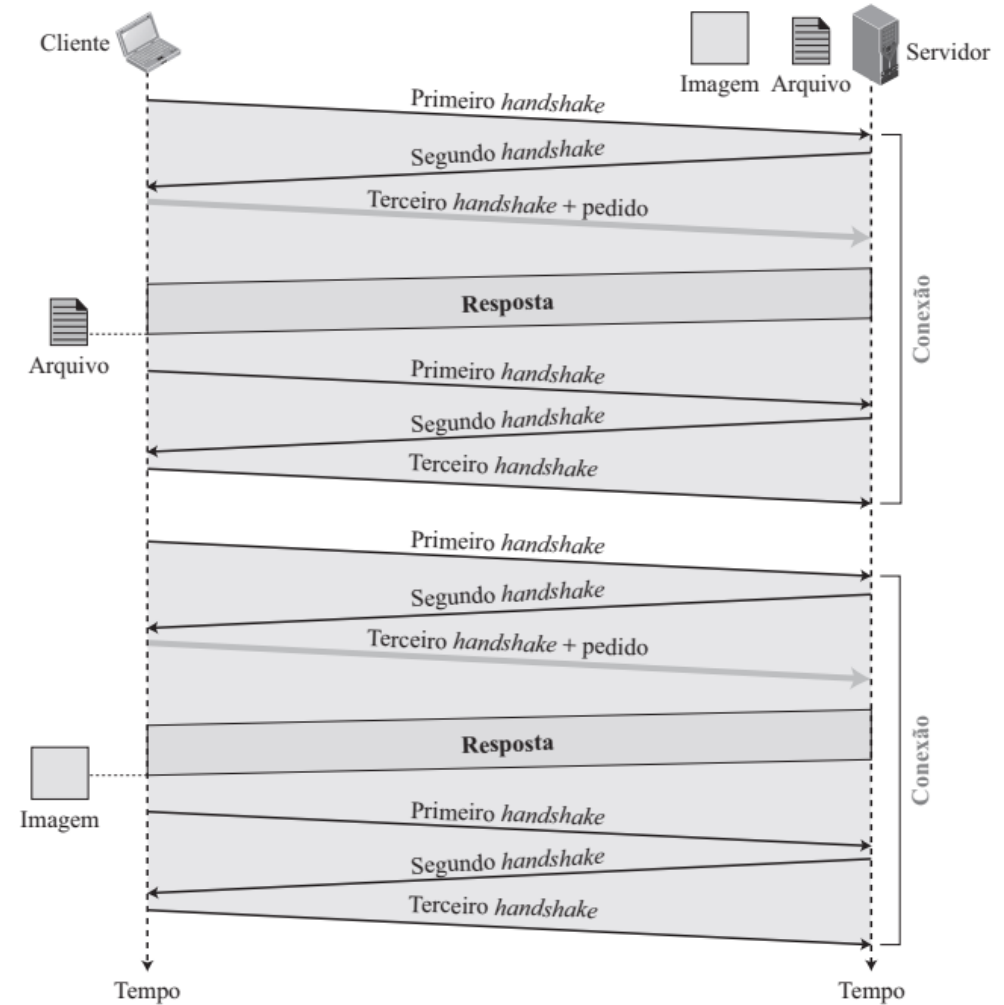
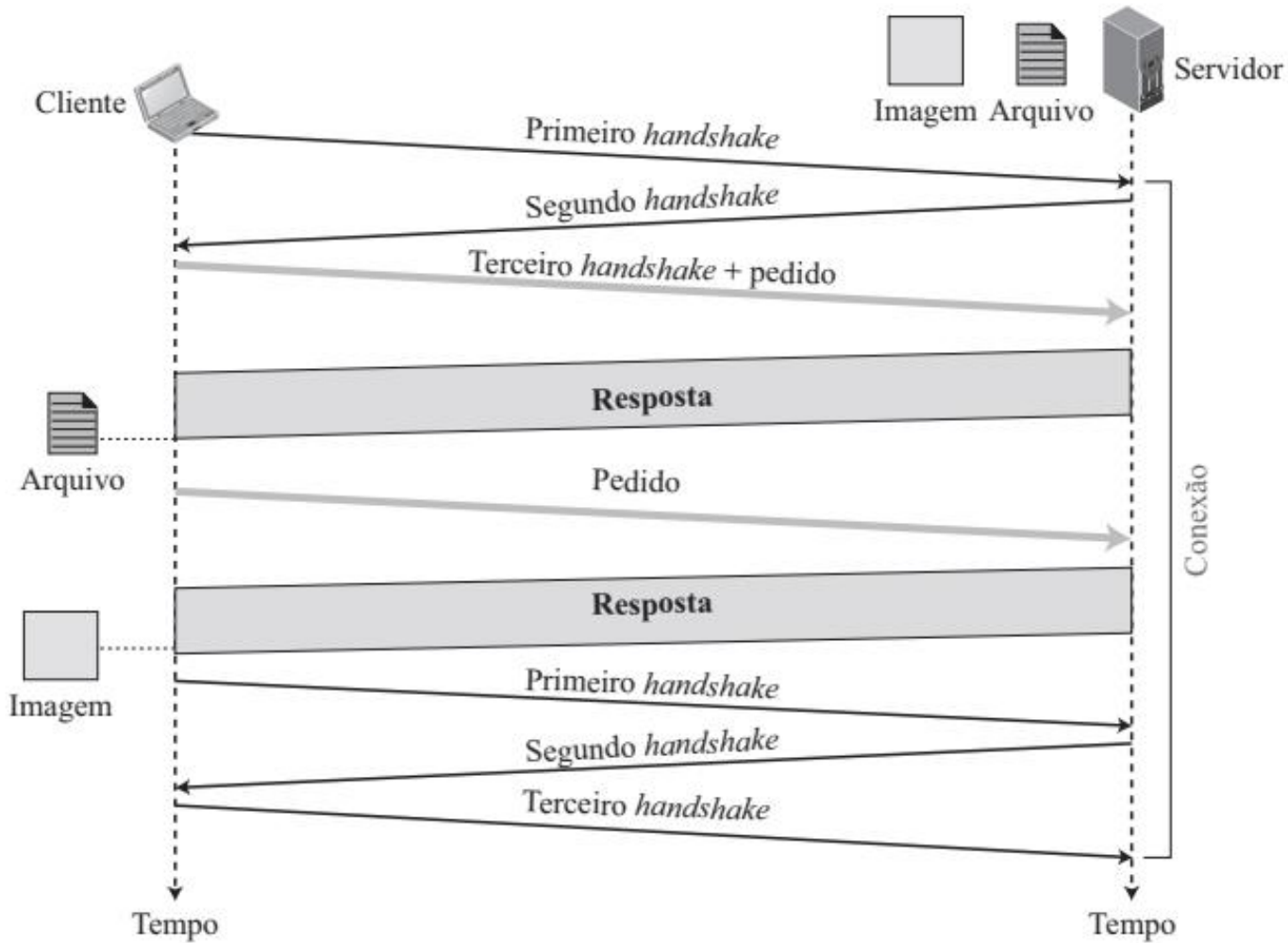
Request



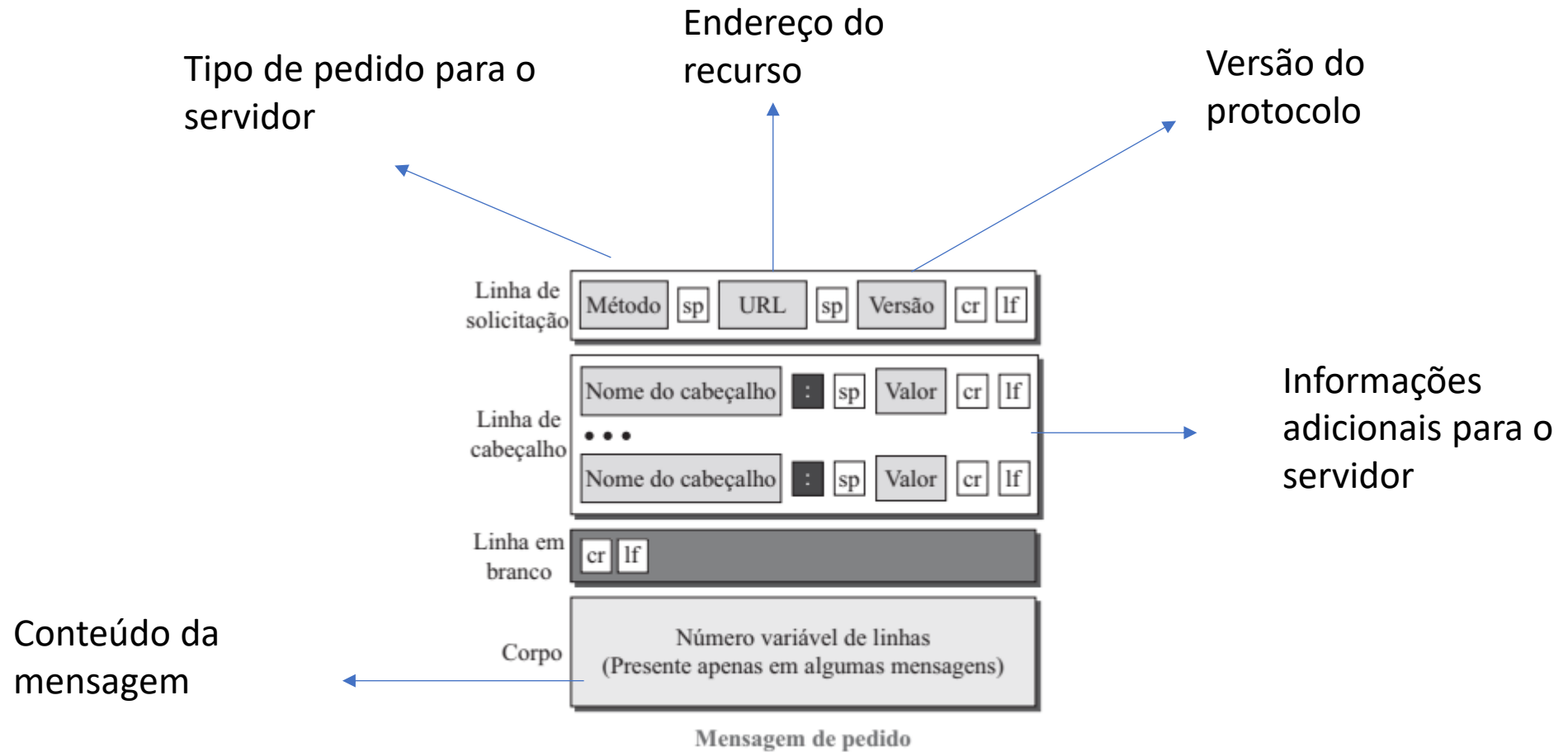
Response



Conexão persistente vs não persistente



Mensagem de Pedido



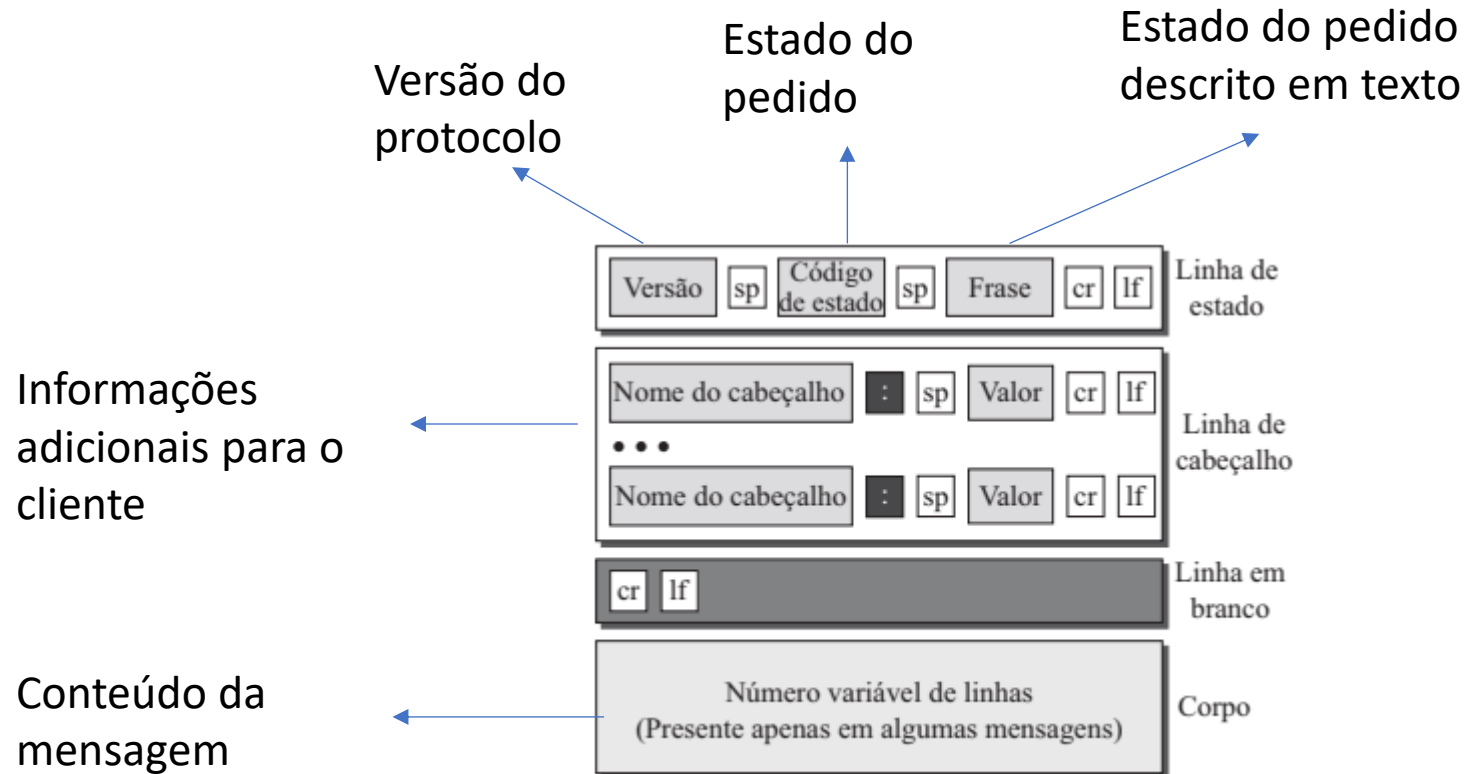
Métodos para Requisição

- **GET**: solicita recurso web ao servidor.
- **PUT**: inverso do Get, envia um novo recurso do cliente para o servidor.
- **POST**: envia alguma informação para ser adicionada a recurso existente.
- **DELETE**: cliente pede a remoção de um recurso web (se tiver permissão).

Nomes de cabeçalho de pedido

Cabeçalho	Descrição
User-agent	Identifica o programa-cliente
Accept	Mostra o formato de mídia que o cliente pode aceitar
Accept-charset	Mostra o conjunto de caracteres que o cliente pode manipular
Accept-encoding	Mostra o esquema de codificação que o cliente pode manipular
Accept-language	Mostra o idioma que o cliente pode aceitar
Authorization	Mostra quais permissões o cliente tem
<i>Host</i>	Mostra o <i>host</i> e o número de porta do cliente
Date	Mostra a data atual
Upgrade	Especifica o protocolo de comunicação preferencial
<i>Cookie</i>	Devolve o <i>cookie</i> para o servidor (explicado mais adiante)
If-Modified-Since	Se o arquivo foi modificado desde uma data específica

Mensagem de Resposta



Códigos de estado

- 100 até 199 – códigos informativos.
- 200 até 299 – sucesso de solicitação.
- 300 até 399 – redirecionamento para outra URL.
- 400 até 499 – erro no lado cliente.
- 500 até 599 – erro no lado servidor.

504 Gateway Time-out

server

404

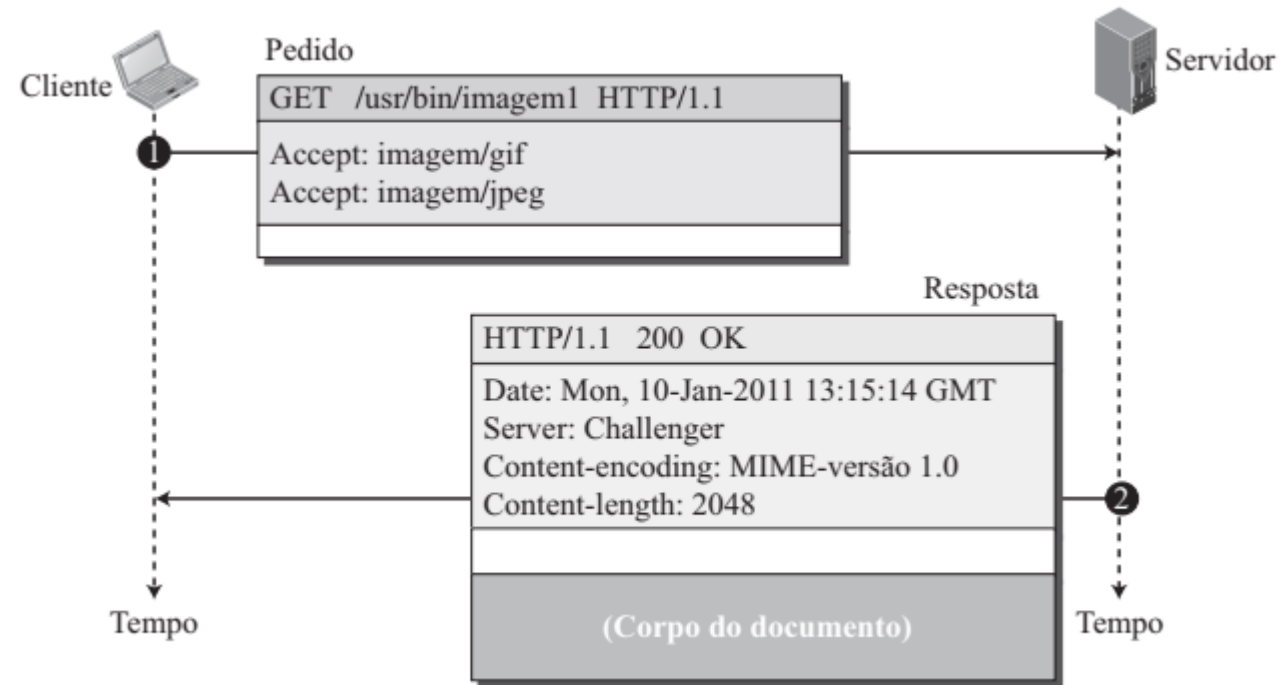
Not Found

The resource requested could not be found on this server!

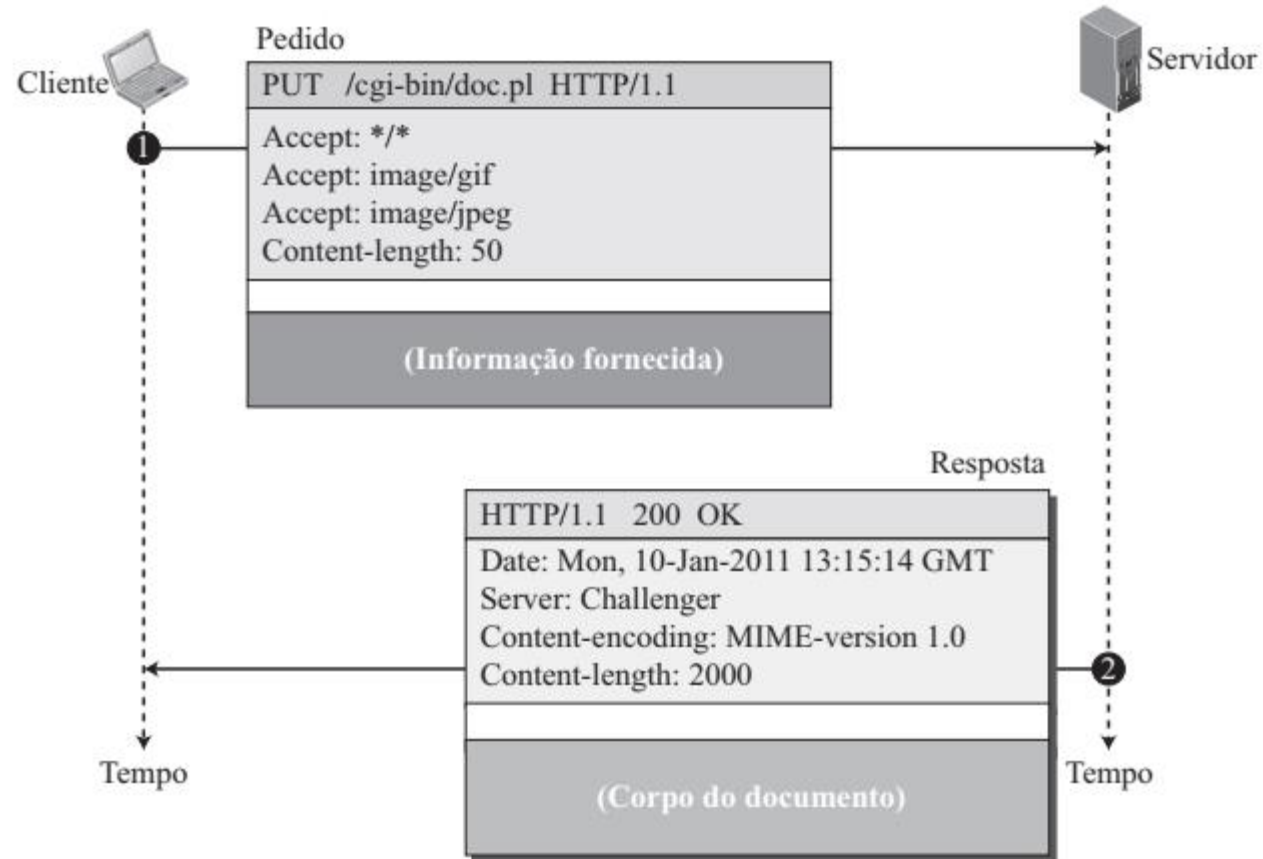
Nomes de cabeçalho de resposta

Cabeçalho	Descrição
Date	Mostra a data atual
Upgrade	Especifica o protocolo de comunicação preferencial
Server	Fornecer informações sobre o servidor
Set-Cookie	O servidor pede ao cliente que salve um <i>cookie</i>
Content-Encoding	Especifica o esquema de codificação
Content-Language	Especifica o idioma
Content-Length	Mostra o comprimento do documento
Content-Type	Especifica o tipo de mídia
Location	Para pedir ao cliente que envie o pedido a outro <i>site</i>
Accept-Ranges	O servidor aceitará as faixas de <i>byte</i> requisitadas
Last-modified	Fornecer a data e a hora da última alteração

Exemplo de mensagem GET



Exemplo de mensagem PUT



Pedido condicional

Mensagem de pedido

```
GET http://www.commonServer.com/information/arquivo1 HTTP/1.1  
If-Modified-Since: Thu, Sept 04 00:00:00 GMT
```

Linha de solicitação

Linha de cabeçalho

Linha em branco

Mensagem de resposta

```
HTTP/1.1 304 Not Modified  
Date: Sat, Sept 06 08 16:22:46 GMT  
Server: commonServer.com  
  
(Corpo vazio)
```

Linha de estado

Primeira linha de cabeçalho

Segunda linha de cabeçalho

Linha em branco

Corpo vazio

Usando uma requisição GET

- O que acontece se fizermos uma requisição GET contra uma dessas páginas?
- <https://www.pucrs.br/politecnica/a-escola/professores/>
- <http://books.toscrape.com/>
- <https://www.scrapethissite.com/pages/simple/>

Document Object Model

Interface para acesso dinâmico

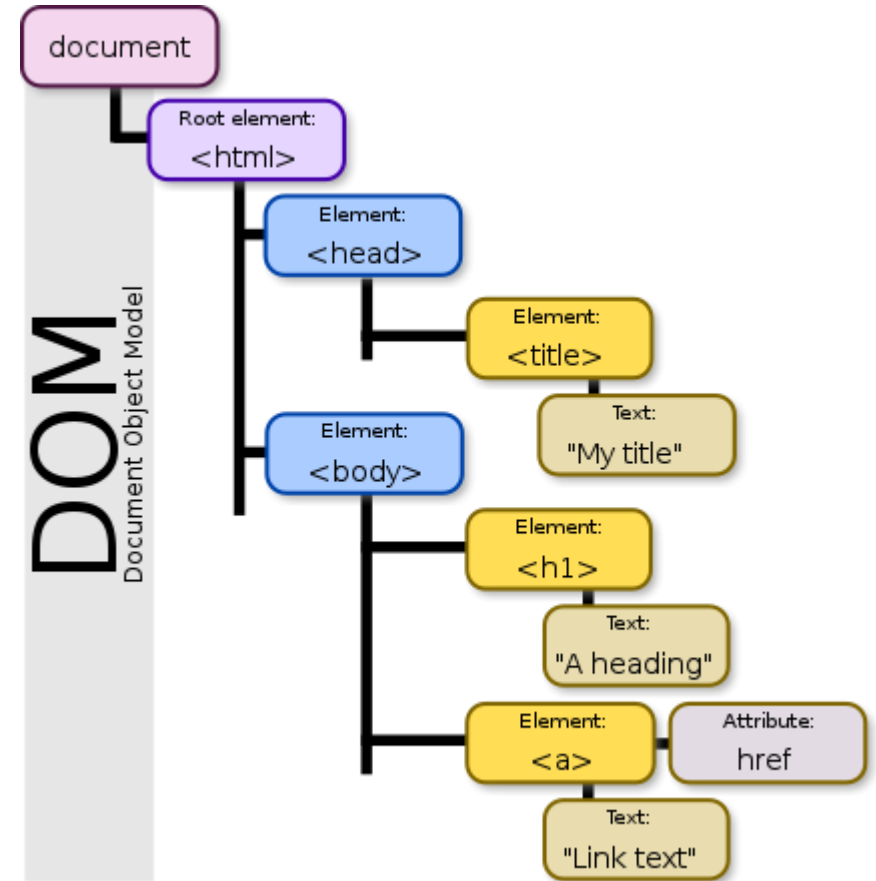
DOM HTML

Document Object Model:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Permite acesso a:

- Elementos HTML como objetos
- As propriedades dos elementos HTML
- Métodos para acessar todos os elementos HTML
- Eventos de elementos HTML



Boas práticas para scraping

Arquivo /robots.txt

- “The Robots Exclusion Protocol”
- Arquivo **público** que o dono do site define para dar instruções sobre o seu site para robôs da web.
- Fica localizado na raiz do site.
- O robô acessa esse arquivo para saber sobre quem pode pegar informações e que partes do site podem ser acessadas.
- Composto por linhas **User-Agent** (sobre usuários) e linhas **Disallow** (sobre páginas).
- É apenas uma convenção, um robô **não precisa** respeitar o arquivo se não quiser.

Arquivo /robots.txt

To exclude all robots from the entire server

```
User-agent: *  
Disallow: /
```

To exclude all robots from part of the server

```
User-agent: *  
Disallow: /cgi-bin/  
Disallow: /tmp/  
Disallow: /junk/
```

To allow all robots complete access

```
User-agent: *  
Disallow:
```

To exclude a single robot

```
User-agent: BadBot  
Disallow: /
```

To allow a single robot

```
User-agent: Google  
Disallow:  
  
User-agent: *  
Disallow: /
```

Mais informações: <http://www.robotstxt.org/>

Sitemaps

- Arquivos usados para fornecer informações sobre páginas, vídeos e outros recursos do site e indicar a relação entre eles.
- Em geral, é de interesse do dono do site fornecer um sitemap para ajudar em atividades de crawling (alô, Google!).
- Podemos visualizar o sitemap para entender melhor a estrutura do site e facilitar nossa coleta.

<https://www.sitemaps.org/protocol.html>

```
▼<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  ▼<url>
    <loc>http://web-scraping.appspot.com/places/default/view/Afghanistan-1</loc>
  </url>
  ▼<url>
    <loc>http://web-scraping.appspot.com/places/default/view/Aland-Islands-2</loc>
  </url>
  ▼<url>
    <loc>http://web-scraping.appspot.com/places/default/view/Albania-3</loc>
  </url>
  ▼<url>
    <loc>http://web-scraping.appspot.com/places/default/view/Algeria-4</loc>
  </url>
```

Vamos ver alguns exemplos

- <https://www.pucrs.br/>
- <https://g1.globo.com/>
- <https://en.wikipedia.org/>
 - Apenas o robots.txt

Leitura Indicada

- JARMUL, Katharine; LAWSON, Richard. **Python Web Scraping**. Packt Publishing Ltd, 2017.
 - Capítulo 1