

Web Scraping

Web Scrapping

Gabriel Felipe

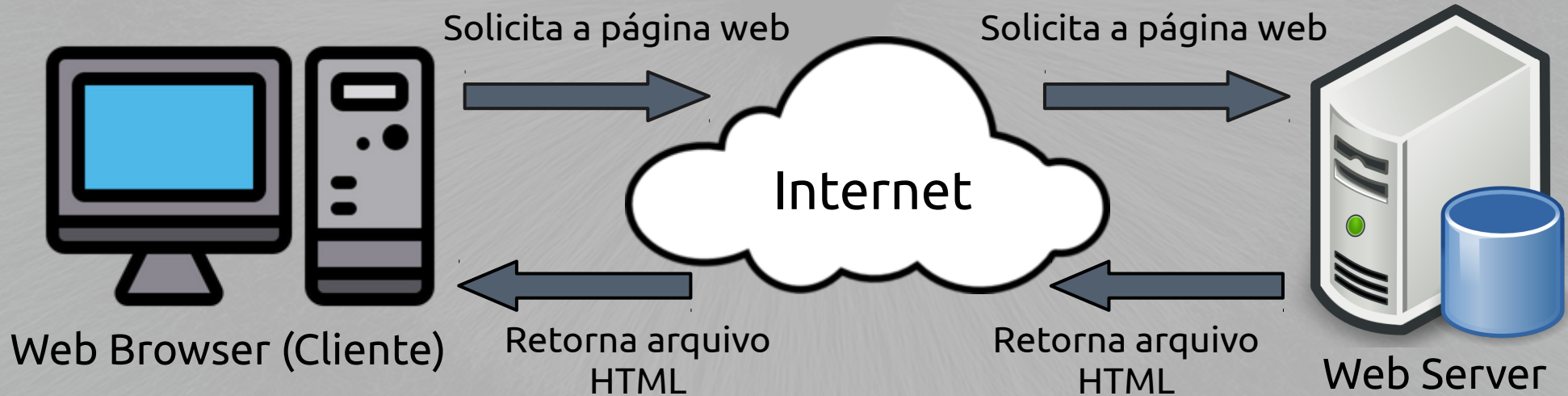
akiradev.netlify.app

Introdução

- **Web Scraping** é a coleta de dados usada para extrair dados de sites.
- O software de web scraping pode acessar diretamente a **World Wide Web** usando o **Hypertext Transfer Protocol** ou através de um navegador web.
- Embora o web scraping possa ser feito manualmente por um usuário de software, o termo normalmente se refere a processos automatizados implementados usando um **bot** ou **web crawler**.
- É considerado uma forma de cópia na qual dados específicos são coletados e copiados da web, normalmente em um banco de dados local central ou planilha, para recuperação ou análise posterior.

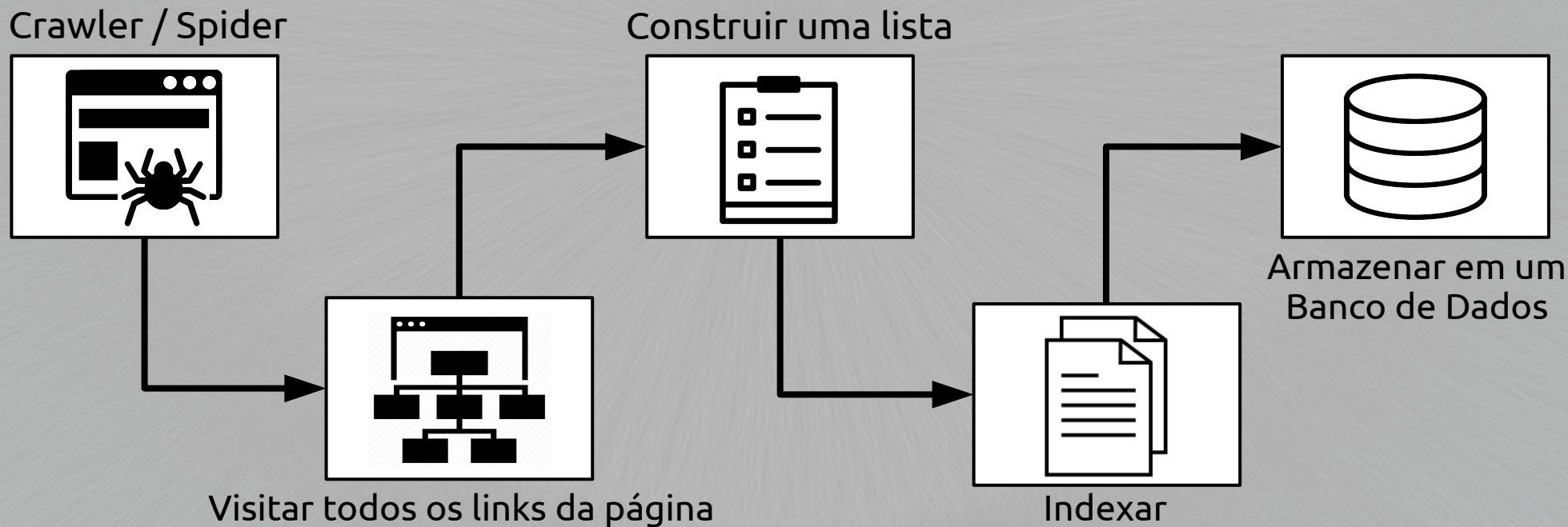
Introdução

- O web scraping de uma página da web envolve buscá-la e extraí-la. Buscar é o download de uma página (que um navegador faz quando um usuário visualiza uma página).



Web Crawling

- Portanto, o **web crawling** é um componente principal do web scraping, para buscar páginas para processamento posterior. Depois de coletadas as páginas, a extração pode ocorrer.



Web Scraping

- O conteúdo de uma página pode ser analisado (**parsed**), pesquisado, reformatado, seus dados copiados em uma planilha ou carregados em um banco de dados.




Web Scrapers

- Os web scrapers geralmente extraem algo de uma página, para fazer uso dele para outra finalidade em outro lugar.
- Um exemplo seria localizar e copiar nomes e números de telefone, ou empresas e seus URL's, ou endereços de e-mail para uma lista (coleta de contato).
- Web Scraping é usado para scraping de contato e como um componente de aplicativos usados para indexação da web, mineração da web e mineração de dados, monitoramento de mudanças de preços online e comparação de preços, raspagem de análises de produtos (para observar a concorrência), coleta de listagens de imóveis, monitoramento de dados meteorológicos, detecção de alteração de site, pesquisa, rastreamento de presença e reputação online e integração de dados da web.

Páginas Web

- As páginas Web são construídas usando linguagens de marcação baseadas em texto (**HTML** e **XHTML**) e frequentemente contêm uma grande quantidade de dados úteis na forma de texto.

Ciência da Computação



659.27 x 150

Ciência da computação é o estudo de computadores e computação, incluindo seus fundamentos teóricos e algorítmicos, hardware e software e seus usos para processamento de informações. A disciplina de ciência da computação inclui o estudo de algoritmos e estruturas de dados, design de computadores e redes, modelagem de dados e processos de informação, e inteligência artificial.

A **ciência da computação** extrai alguns de seus fundamentos da matemática e da engenharia e, portanto, incorpora técnicas de áreas como teoria das filas, probabilidade e estatística e design de circuitos eletrônicos. A ciência da computação também faz uso intenso de testes de hipóteses e experimentação durante a conceituação, design, medição e refinamento de novos algoritmos, estruturas de informação e arquiteturas de computador.

Elements Console Sources Network Performance Memory

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Ciência da Computação</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="Website de Ciência da Computação">
    <meta name="author" content="Gabriel Felipe">
    <link rel="stylesheet" type="text/css" href="styles/style.css">
    <script src="https://kit.fontawesome.com/abb211c846.js"></script>
  </head>
  <body cz-shortcut-listen="true">
    <div>
      <h1 id="titulo">Ciência da Computação</h1>
      
    ...
    <p>
      <b>Ciência da computação</b>
      " é o estudo de computadores e computação, incluindo seus fundamentos teóricos e algorítmicos, hardware e software e seus usos para processamento de informações. A disciplina de ciência da computação inclui o estudo de algoritmos e estruturas de dados, design de computadores e redes, modelagem de dados e processos de informação, e inteligência artificial."
    </p>
  </div>
</body>
</html>
```

html body div p

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls + [Q]

element.style {

}

p {

text-align: justify;

font-size: 1.13rem;

background-color: #ebebcb;

padding: 10px;

border: 2px solid black;

}

p {

display: block;

margin-block-start: 1em;

margin-block-end: 1em;

margin-inline-start: 0px;

margin-inline-end: 0px;

}

user agent stylesheet

HTML Básico

- HTML (**H**ypertext **M**arkup **L**anguage) é o código usado para estruturar uma página da web e seu conteúdo.
- Por exemplo, o conteúdo pode ser estruturado em um conjunto de parágrafos, uma lista de pontos com marcadores ou usando imagens e tabelas de dados.
- HTML é uma linguagem de marcação que define a estrutura do seu conteúdo.
- O HTML consiste em uma série de **elementos**, que você usa para incluir, ou agrupar, diferentes partes do conteúdo para fazê-lo aparecer de uma determinada maneira ou agir de uma determinada maneira.

HTML Básico

- As **tags** delimitadoras podem criar um hiperlink de palavra ou imagem para outro lugar, podem colocar palavras em itálico, podem tornar a fonte maior ou menor e assim por diante.
- Por exemplo, considere a seguinte linha de conteúdo:

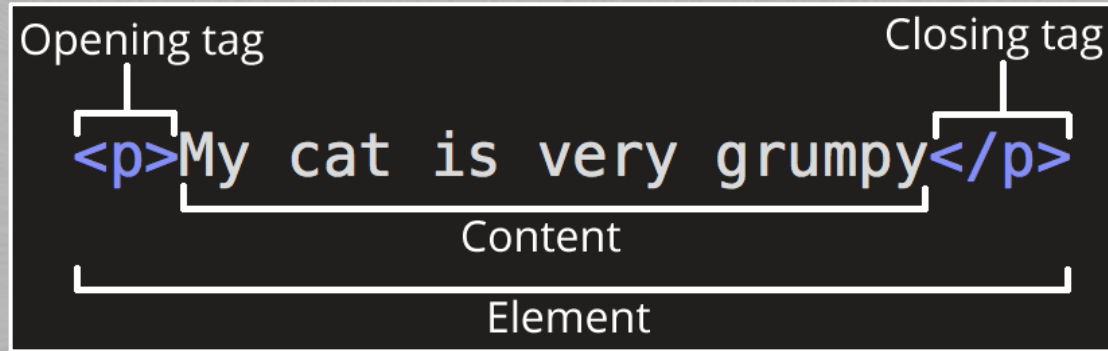
Everything you can imagine is real.

- Se quiséssemos que a linha ficasse sozinha, poderíamos especificar que é um parágrafo, incluindo-o entre tags de parágrafo:

```
<p>Everything you can imagine is real.</p>
```

Anatomia de um Elemento HTML

- Vamos explorar a tag parágrafo um pouco mais.



- As principais partes do nosso elemento são as seguintes:
 1. **Tag de abertura:** consiste no nome do elemento (neste caso, p), envolto em colchetes angulares de abertura e fechamento. Isso indica onde o elemento começa ou começa a ter efeito, neste caso, onde o parágrafo começa.
 2. **Tag de fechamento:** é a mesma que a tag de abertura, exceto que inclui uma barra antes do nome do elemento. Isso indica onde o elemento termina, neste caso, onde o parágrafo termina. Deixar de adicionar uma tag de fechamento é um dos erros padrão do iniciante e pode levar a resultados estranhos.
 3. **Conteúdo:** é o conteúdo do elemento, que, neste caso, é apenas texto.
 4. **Elemento:** a tag de abertura, a tag de fechamento e o conteúdo juntos constituem o elemento.

Anatomia de um Elemento HTML

- Os elementos também podem ter **atributos** semelhantes a estes:



```
<p class="editor-note">My cat is very grumpy</p>
```

- Os atributos contêm informações extras sobre o elemento que você não deseja que apareça no conteúdo real. Aqui, **class** é o nome do atributo e **editor-note** é o valor do atributo. O atributo **class** permite que você dê ao elemento um identificador não exclusivo que pode ser usado para selecioná-lo (e quaisquer outros elementos com o mesmo valor de class) com informações de estilo e outras coisas.

Anatomia de um Elemento HTML

- Um atributo sempre deve ter o seguinte:
 1. Um espaço entre ele e o nome do elemento (ou o atributo anterior, se o elemento já tiver um ou mais atributos).
 2. O nome do atributo seguido por um sinal de igual.
 3. O valor do atributo agrupado abrindo e fechando aspas.

Nota: Os valores de atributos simples que não contêm espaços em branco ASCII (ou qualquer um dos caracteres " ' ` = < >) podem permanecer sem aspas, mas é recomendado que você cite todos os valores de atributos, pois isso torna o código mais consistente e compreensível.

Aninhando Elementos

- Você também pode colocar elementos dentro de outros elementos, isso é chamado de aninhamento (**nesting**). Se quiséssemos destacar a palavra “imagine” de nossa frase, poderíamos envolver a palavra “imagine” em um elemento ****, o que significa que a palavra deve ser fortemente enfatizada:

```
<p>Everything you can <strong>imagine</strong> is real.</p>
```

- No entanto, você precisa se certificar de que seus elementos estão aninhados corretamente. No exemplo acima, abrimos primeiro o elemento **<p>** e, em seguida, o elemento ****; portanto, temos que fechar primeiro o elemento **** e, em seguida, o elemento **<p>**. O exemplo a seguir está **incorreto**:

```
<p>Everything you can <strong>imagine is real.</p></strong>
```


Elementos Vazios

- Alguns elementos não têm conteúdo e são chamados de elementos vazios.
- Pegue o elemento **** por exemplo:

```

```

- Ele contém dois atributos, mas não há tag de fechamento **** e nenhum conteúdo interno. Isso ocorre porque um elemento de imagem não envolve o conteúdo para afetá-lo. Seu objetivo é inserir uma imagem na página HTML no local em que aparece.



Anatomia de um Documento HTML

- A seguir temos a anatomia de um documento HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Página de Exemplo</title>
  </head>
  <body>
    
  </body>
</html>
```

Anatomia de um Documento HTML

- No exemplo anterior temos o seguinte:
 - ♦ **<!DOCTYPE html>** - **doctype**. É um preâmbulo obrigatório. Nas brumas do tempo, quando o HTML era jovem (por volta de 1991/92), os doctypes deveriam atuar como links para um conjunto de regras que a página HTML tinha que seguir para ser considerada um bom HTML, o que poderia significar verificação automática de erros e outras coisas úteis. No entanto, atualmente, eles não fazem muito e são basicamente necessários para garantir que seu documento se comporte corretamente.
 - ♦ **<html></html>** - o elemento **<html>**. Este elemento envolve todo o conteúdo da página inteira e às vezes é conhecido como o elemento raiz.
 - ♦ **<head></head>** - o elemento **<head>**. Este elemento atua como um contêiner para todas as coisas que você deseja incluir na página HTML que não seja o conteúdo que você está mostrando aos visualizadores da página. Isso inclui coisas como palavras-chave e uma descrição de página que você deseja que apareça nos resultados de pesquisa, CSS para definir o estilo de nosso conteúdo, declarações de conjunto de caracteres e muito mais.

Anatomia de um Documento HTML

- No exemplo anterior temos o seguinte:
 - ♦ **<meta charset="utf-8">** - Este elemento define o conjunto de caracteres que seu documento deve usar para UTF-8, que inclui a maioria dos caracteres da grande maioria das línguas escritas. Essencialmente, agora ele pode lidar com qualquer conteúdo textual que você possa colocar nele. Não há razão para não definir isso e pode ajudar a evitar alguns problemas mais tarde.
 - ♦ **<title></title>** - o elemento **<title>**. Isso define o título da sua página, que é o título que aparece na guia do navegador em que a página é carregada. Também é usado para descrever a página quando você a marca/adiciona aos favoritos.
 - ♦ **<body></body>** - o elemento **<body>**. Ele contém todo o conteúdo que você deseja mostrar aos usuários da web quando eles visitam sua página, seja texto, imagens, vídeos, jogos, faixas de áudio reproduzíveis ou qualquer outra coisa.

Imagens

- Vamos voltar nossa atenção para o elemento **** novamente:

```

```

- Como dissemos antes, ele incorpora uma imagem em nossa página na posição em que aparece. Ele faz isso por meio do atributo **src** (source), que contém o caminho para o nosso arquivo de imagem.
- Veja que também incluímos um atributo **alt** (alternative). Neste atributo, você especifica um texto descritivo para usuários que não podem ver a imagem, possivelmente pelos seguintes motivos:
 1. Eles são deficientes visuais. Usuários com deficiência visual significativa costumam usar ferramentas chamadas leitores de tela para ler o texto alternativo para eles.
 2. Algo deu errado fazendo com que a imagem não fosse exibida, isso pode incluir o caminho da imagem incorreto ou algum problema de conexão.

Marcando Texto

- Os elementos de título permitem que você especifique que certas partes do seu conteúdo são títulos - ou subtítulos. Da mesma forma que um livro tem o título principal, os títulos dos capítulos e os subtítulos, um documento HTML também pode.
- O HTML contém 6 níveis de título, **<h1>** - **<h6>**, embora você normalmente use apenas 3 a 4 no máximo:

```
<h1>Meu Título Principal</h1>  
<h2>Meu subtítulo principal</h2>  
<h3>Meu segundo subtítulo</h3>  
<h4>Meu terceiro subtítulo</h4>
```


Marcando Texto

- Muito do conteúdo da web são **listas** e o HTML tem elementos especiais para elas. Marcar listas sempre consiste em pelo menos 2 elementos. Os tipos de lista mais comuns são listas **ordenadas** e **não ordenadas**:
 1. Listas não ordenadas são para listas em que a ordem dos itens não importa, como uma lista de compras. Eles são agrupados em um elemento ****.
 2. Listas ordenadas são para listas em que a ordem dos itens é importante, como uma receita. Eles são agrupados em um elemento ****.
- Cada item dentro das listas é colocado dentro de um elemento **** (item da lista).

```
<p>Linguagens de Programação</p>
```

```
<ul>
```

```
<li>Python</li>
```

```
<li>JavaScript</li>
```

```
<li>C++</li>
```

```
</ul>
```

Marcando Texto

- Os links são muito importantes - eles são o que torna a web uma web! Para adicionar um link, precisamos usar um elemento simples - `<a>` - "a" sendo a forma abreviada de "anchor". Vejamos um exemplo de link:

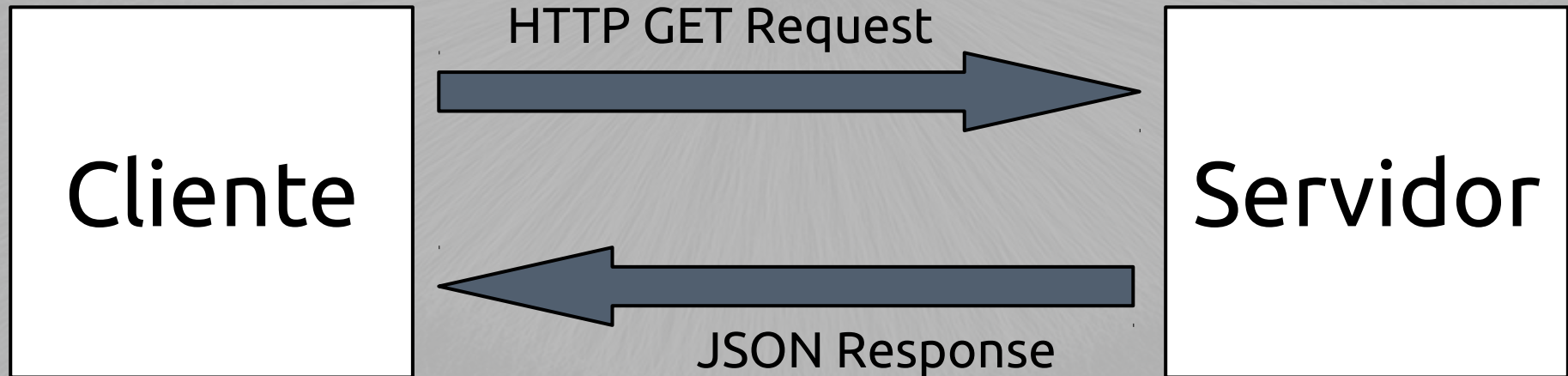
```
<a href="http://cc33z.herokuapp.com">Curso de Comp</a>
```

- Observe que: estamos envolvendo o texto "Curso de Comp" em um elemento `<a>` e estamos usando um atributo **href** para fazer referência ao site **http://cc33z.herokuapp.com**

href pode parecer uma escolha bastante obscura para um nome de atributo em primeiro lugar. Se você estiver tendo problemas para se lembrar, lembre-se de que significa **hypertext reference**.

Formas de Web Scraping

- As novas formas de web scraping envolvem o monitoramento de feeds de dados de servidores web.
- Por exemplo, **JSON** é comumente usado como um mecanismo de armazenamento de transporte entre o **cliente** e o **servidor** da web.



História

- A história do web scraping remonta quase à época em que a World Wide Web nasceu.
 - ◆ Após o nascimento da **World Wide Web** em 1989, o primeiro robô da web, **World Wide Web Wanderer**, foi criado em junho de 1993, com o objetivo apenas de medir o tamanho da web.
 - ◆ Em dezembro de 1993, foi lançado o primeiro mecanismo de busca na web baseado em crawler, o JumpStation. Como não havia tantos sites disponíveis na web, os mecanismos de pesquisa na época costumavam contar com seus administradores humanos de sites para coletar e editar os links em um formato específico. Em comparação, JumpStation trouxe um novo salto, sendo o primeiro motor de busca WWW que contou com um robô da web.
 - ◆ Em 2000, surgiu o primeiro Web API e crawler de API. API significa Application Programming Interface. É uma interface que torna muito mais fácil desenvolver um programa, fornecendo os blocos de construção. Em 2000, a Salesforce e o eBay lançaram sua própria API, com a qual os programadores puderam acessar e baixar alguns dos dados disponíveis ao público. Desde então, muitos sites oferecem API's da web para as pessoas acessarem seu banco de dados público.

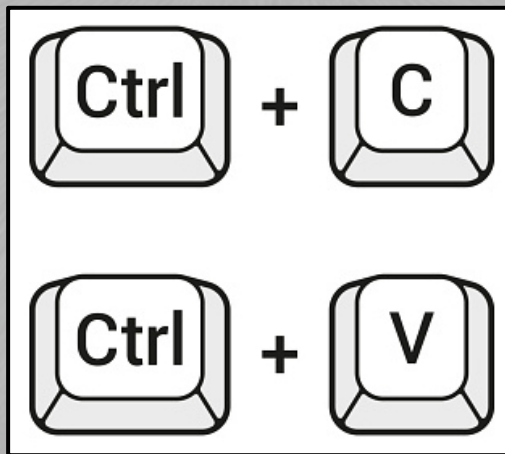
Técnicas

- Web scraping é o processo de extração automática de dados ou coleta de informações da World Wide Web.
- É um campo com desenvolvimentos ativos que compartilham um objetivo comum com a visão da web semântica, uma iniciativa ambiciosa que ainda requer avanços no processamento de texto, compreensão semântica, inteligência artificial e interações homem-computador.
- As soluções atuais de web scraping vão desde o ad-hoc, que exige esforço humano, até sistemas totalmente automatizados que são capazes de converter sites inteiros em informações estruturadas, com limitações.



Copiar e Colar Manualmente

- A forma mais simples de web scraping é copiar e colar manualmente os dados de uma página da web em um arquivo de texto ou planilha.
- Às vezes, mesmo a melhor tecnologia de web scraping não pode substituir o exame manual de um humano e copiar e colar, e às vezes essa pode ser a única solução viável quando os sites de web scraping explicitamente configuram barreiras para impedir a automação da máquina.



Correspondência de Padrão Textual

- Uma abordagem simples, porém poderosa, para extrair informações de páginas da web pode ser baseada no comando **grep** do UNIX ou nos recursos de correspondência de **expressões regulares** de linguagens de programação (por exemplo, **Perl** ou **Python**).

```
VALID_EMAIL_REGEX = /\A[\w+\-\.]+\@[a-z\d\-\\.]+\.[a-z]+\z/i
validates_email presence: true, length: { maximum: 255 }, REGEX },
```

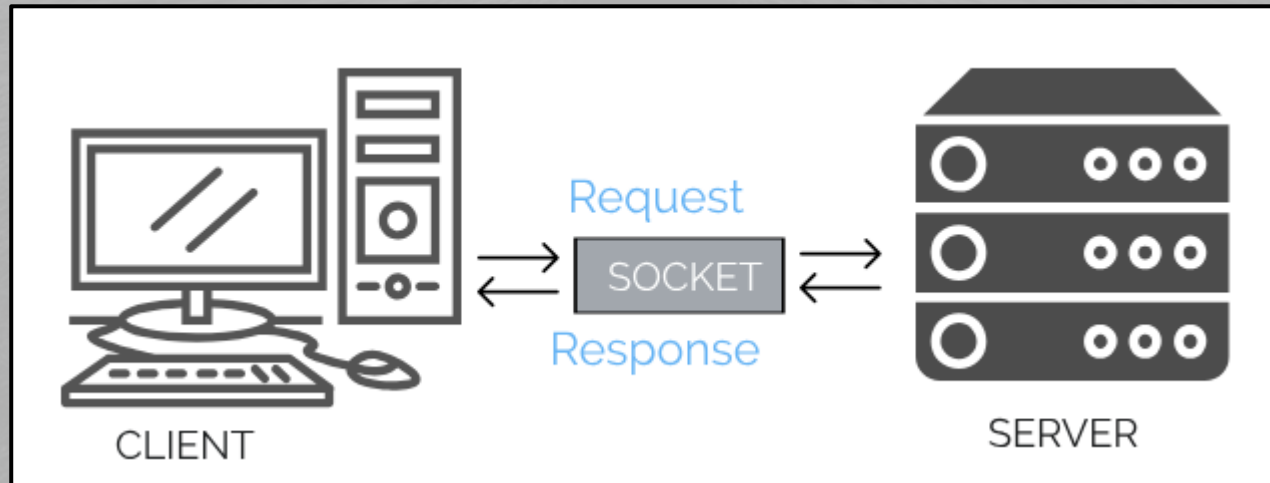
RegExp: `\A[\w+\-\.]+\@[a-z\d\-\\.]+\.[a-z]+\z`

Sample: `example@jetbrains.com|`

Matches!

Programação HTTP

- Páginas da web estáticas e dinâmicas podem ser recuperadas postando solicitações **HTTP** para o servidor da web remoto usando programação de **socket**.

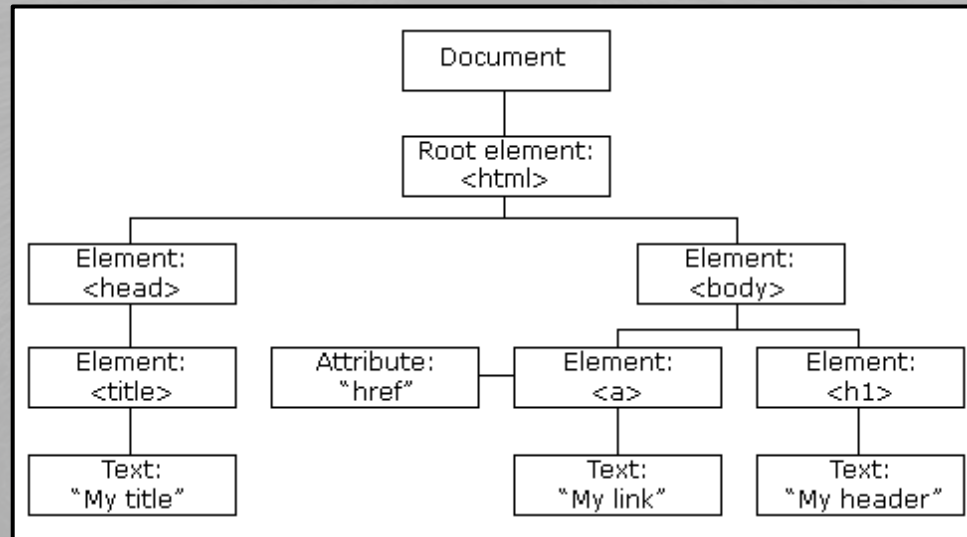


HTML Parsing

- Muitos sites têm grandes coleções de páginas geradas dinamicamente a partir de uma fonte estruturada subjacente, como um banco de dados. Dados da mesma categoria são normalmente codificados em páginas semelhantes por um script ou modelo comum. Na mineração de dados, um programa que detecta esses modelos em uma fonte de informação específica, extrai seu conteúdo e o traduz em uma forma relacional, é chamado de wrapper. Os algoritmos de geração de wrapper assumem que as páginas de entrada de um sistema de indução de wrapper estão em conformidade com um modelo comum e que podem ser facilmente identificadas em termos de um esquema comum de URL. Além disso, algumas linguagens de consulta de dados semiestruturados, como **XQuery** e **HTQL**, podem ser usadas para analisar páginas HTML e para recuperar e transformar o conteúdo da página.

DOM Parsing

- Ao incorporar um navegador da Web completo, como o Internet Explorer ou o controle de navegador Mozilla, os programas podem recuperar o conteúdo dinâmico gerado por scripts do lado do cliente. Esses controles do navegador também analisam páginas da web em uma árvore DOM, com base na qual os programas podem recuperar partes das páginas. Linguagens como o **Xpath** podem ser usadas para analisar a árvore DOM resultante.

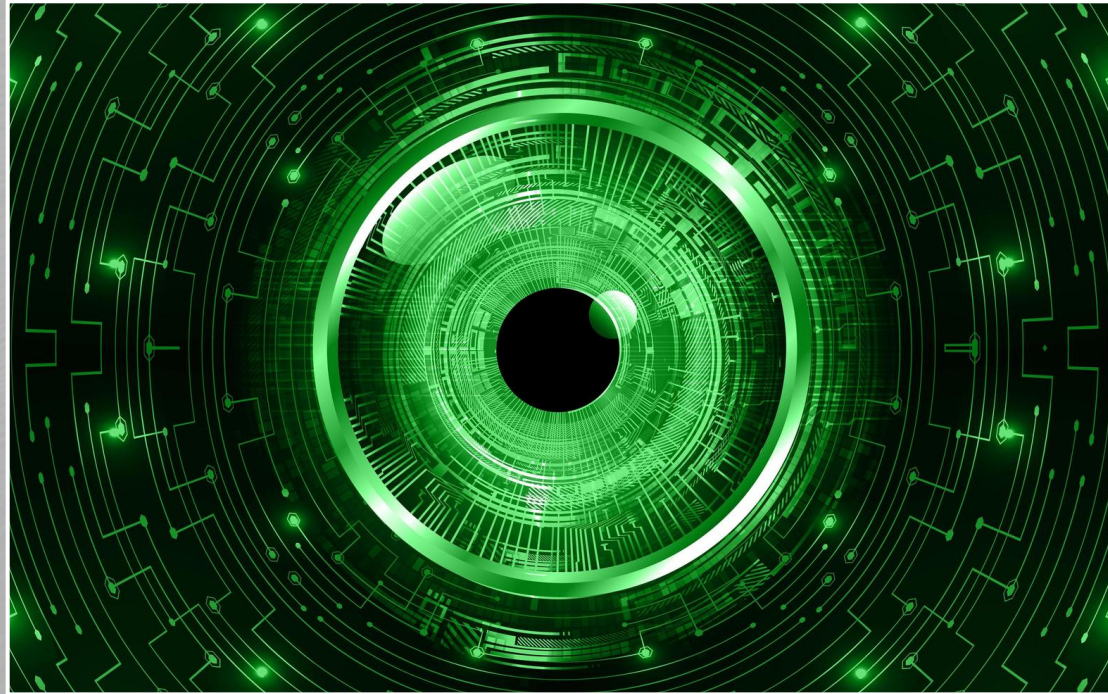


Aggregação Vertical

- Existem várias empresas que desenvolveram plataformas de colheita específicas verticais. Essas plataformas criam e monitoram uma infinidade de "bots" para verticais específicas sem "man in the loop" (sem envolvimento humano direto) e sem trabalho relacionado a um site de destino específico. A preparação envolve estabelecer a base de conhecimento para toda a vertical e, em seguida, a plataforma cria os bots automaticamente. A robustez da plataforma é medida pela qualidade das informações que ela recupera (geralmente o número de campos) e sua escalabilidade (a rapidez com que pode escalar para centenas ou milhares de sites). Essa escalabilidade é usada principalmente para atingir a **Long Tail** de sites que os agregadores comuns consideram complicados ou muito trabalhosos para coletar conteúdo.

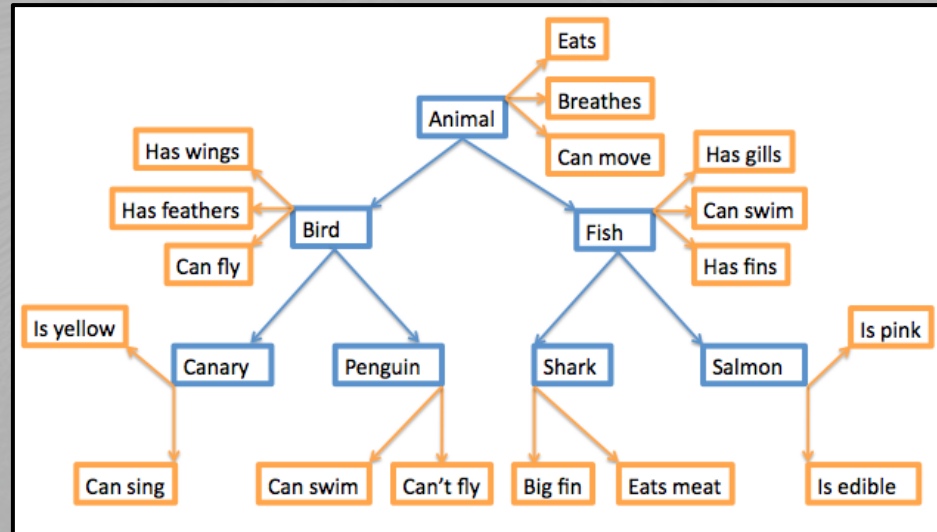
Análise de Visão Computacional da Página Web

- Existem esforços usando **machine learning** e **visão computacional** que tentam identificar e extrair informações de páginas da web interpretando as páginas visualmente como um ser humano faria.



Reconhecimento de Anotação Semântica

- As páginas que estão sendo copiadas podem incluir **metadados** ou marcações semânticas e anotações, que podem ser usadas para localizar fragmentos de dados específicos. Se as anotações forem incorporadas às páginas, como o **Microformat** faz, essa técnica pode ser vista como um caso especial de análise DOM. Em outro caso, as anotações, organizadas em uma camada semântica, são armazenadas e gerenciadas separadamente das páginas web, para que os scrapers possam recuperar o esquema de dados e as instruções dessa camada antes de fazer o scraping das páginas.

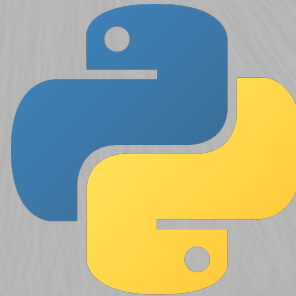


Software

- Existem muitas ferramentas de software disponíveis que podem ser usadas para personalizar soluções de web scraping. Este software pode tentar reconhecer automaticamente a estrutura de dados de uma página ou fornecer uma interface de gravação que elimine a necessidade de escrever manualmente o código de web scraping ou algumas funções de script que podem ser usadas para extrair e transformar conteúdo e interfaces de banco de dados que podem armazenar os dados coletados em bancos de dados locais. Alguns softwares de web scraping também podem ser usados para extrair dados diretamente de uma API.



Scrapy



Impedindo Web Scraping

- Existem métodos que alguns sites usam para evitar o web scraping, como detectar e impedir que os bots rastreiem (visualizem) suas páginas. Em resposta, existem sistemas de web scraping que contam com o uso de técnicas de análise DOM, visão computacional e processamento de linguagem natural para simular a navegação humana para permitir a coleta de conteúdo da página da web para análise offline.



Métodos para Impedir Scraping

- O administrador de um site pode usar várias medidas para interromper ou desacelerar um bot. Algumas técnicas incluem:
 - ◆ Bloquear um endereço IP manualmente ou com base em critérios como geolocalização e DNSRBL. Isso também bloqueará toda a navegação desse endereço.
 - ◆ Desativar qualquer API de serviço da web que o sistema do site possa expor.
 - ◆ Às vezes, os bots declaram quem são (usando strings de **user agent**) e podem ser bloqueados com base no arquivo **robots.txt**; 'googlebot' é um exemplo. Outros bots não fazem distinção entre eles e um humano que usa um navegador.
 - ◆ Os bots podem ser bloqueados monitorando o excesso de tráfego.
 - ◆ Às vezes, os bots podem ser bloqueados com ferramentas para verificar se é uma pessoa real acessando o site, como um CAPTCHA. Às vezes, os bots são codificados para quebrar explicitamente padrões CAPTCHA específicos ou podem empregar serviços de terceiros que utilizam trabalho humano para ler e responder em tempo real aos desafios do CAPTCHA.

Métodos para Impedir Scraping

- O administrador de um site pode usar várias medidas para interromper ou desacelerar um bot. Algumas técnicas incluem:
 - ◆ Serviços comerciais anti-bot: as empresas oferecem serviços anti-bot e anti-scraping para websites. Alguns firewalls de aplicativos da web também têm recursos de detecção de bot limitados. No entanto, muitas dessas soluções não são muito eficazes.
 - ◆ Localizar bots com um honeypot ou outro método para identificar os endereços IP de crawlers automatizados.
 - ◆ Ofuscação usando sprites CSS para exibir dados como números de telefone ou endereços de e-mail, ao custo da acessibilidade aos usuários de leitores de tela.
 - ◆ Como os bots dependem da consistência no código de front-end de um site de destino, adicionar pequenas variações ao HTML/CSS em torno de dados importantes e elementos de navegação exigiria um envolvimento mais humano na configuração inicial de um bot e, se feito de forma eficaz, pode tornar o site de destino é muito difícil de raspar devido à capacidade reduzida de automatizar o processo de varredura.
 - ◆ Os sites podem declarar se o crawling é permitido ou não no arquivo **robots.txt** e permitir acesso parcial, limitar a taxa de crawling, especificar o momento ideal para crawling e muito mais.

Tutorial Completo

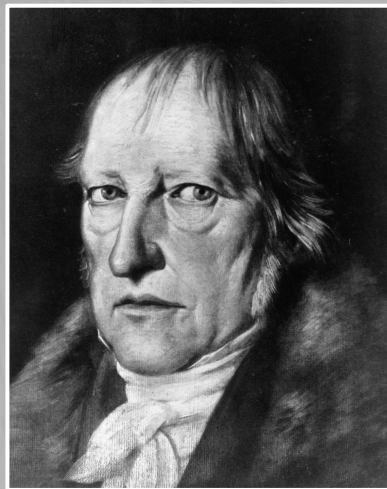


- <https://github.com/the-akira/Python-Web-Scraping>

Bons Estudos!

“Nothing great in the world was accomplished without passion.” **Georg Wilhelm Friedrich Hegel**

“To be independent of public opinion is the first formal condition of achieving anything great.” **Georg Wilhelm Friedrich Hegel**



Referências

- https://en.wikipedia.org/wiki/Web_scraping
- https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics