SEVEN
publicações acadêmicas

Web Scraping: Metodologia de busca e recuperação de dados em ciência da informação

Crossref doi
https://doi.org/10.56238/sevened2023.008-002

Gilnei Machado
Professor Doutor, Docente no Programa de Pós-Graduação em Ciência da Informação da Universidade Estadual de Londrina

RESUMO
A Web Scraping é um processo de coleta automatizada de dados em sites da internet através da ação de "bots" ou programas. A mesma se mostra importante nos dias atuais, pois os bancos de dados estão cada vez maiores e, no geral, temos urgência de obtenção de informações. A técnica apresentada permite extrair textos, dados numéricos, imagens, arquivos e tabelas, disponibilizados tanto na página inicial do site quanto em suas várias abas. O capítulo tem como objetivo apresentar as potencialidades da utilização da Web scraping através do Python na coleta de dados presentes em sites da web e sua importância para a recuperação de informações na Ciência da Informação. Nos utilizamos de linhas de comando escritas por Lisa Tagliaferri, buscando verificar se estas linhas de comando funcionam e se conseguimos, de fato, obter as informações desejadas. O site utilizado para a recuperação das informações desejadas acerca de nomes de artistas e dos links que direcionam aos seus nomes foi o Web archive, que apresenta o registro de todos os artistas cujas obras se encontram no National Gallery of Art, dos EUA. Como resultado, percebemos que as linhas de comando são extremamente úteis na obtenção das informações, uma vez que, permitem, em um tempo curto, obter um grande conjunto de informações. Como conclusões, vimos que a raspagem de dados em sites da web é perfeitamente realizável a partir do Python e seus códigos e que as informações recuperadas foram plenamente satisfatórias.

Palavras-chave: Recuperação de informação, Raspagem de dados, Programação na Ciência da Informação, Python.

1 INTRODUÇÃO

A *Web Scraping* ou raspagem de Dados, como é também conhecida na língua portuguesa, é um processo de coleta automatizada de dados em sites da internet (MITCHELL, 2018) através da ação de "*bots*" ou programas, para posterior tratamento destes dados e retirada das informações desejadas.

A raspagem de dados pode ser realizada de forma manual, quando se trata de uma pequena quantidade de dados, porém, em tempos de "*Big Data*", é praticamente impossível realizar a atividade desta maneira, por isso que, na *Web scraping* a programação é algo de extrema importância, principalmente pelo fato dela possibilitar a automatização da consulta aos servidores onde os dados estão armazenados, sendo bem definida como um método automático de extração de dados (BARBOSA; CAVALCANTI, 2020).

A técnica, ora apresentada, não permite apenas extrair textos ou dados numéricos dos sites analisados, mas também auxilia na extração de informações apresentadas na forma de imagens,

arquivos e tabelas, disponibilizados tanto na página inicial do site quanto em suas várias abas, ou seja, em todas as áreas do site.

Todo o processo de *web scraping* pode ser realizado através do Python ou do R, especialmente as etapas de programação, raspagem e a mineração dos dados para a extração de elementos e comportamentos relevantes aos interesses da pesquisa. No Python são utilizados os módulos *Beautiful Soup* e *Requests*.

Cabe destacar que, não é o objetivo deste capítulo ensinar a usar o Python, o R ou mesmo ensinar uma técnica que permita fazer a raspagem de dados em toda e qualquer página da web. O objetivo aqui é debater acerca da importância da técnica ou do método de *Web Scraping*, para extração de dados de forma automatizada da internet e a transformação destes dados em informações relevantes e úteis (MOOERS, 1951). Sendo assim, o capítulo tem como objetivo apresentar as potencialidades da utilização da *Web scraping* através do Python na coleta de dados presentes em *sites* da *web* e sua importância para a recuperação de informações na Ciência da Informação.

2 METODOLOGIA

Sendo este um debate teórico-conceitual-prático, nos utilizamos de análise e pesquisa bibliográfica a respeito do assunto, buscando elucidar as dúvidas relacionadas a todos os elementos conceituais básicos.

Por outro lado, a parte prática demandou estudo da linguagem utilizada em Python e as formas de escrever a programação para a realização da *Web Scraping* através dos módulos *Beautiful Soup* e *Requests*.

A linguagem de programação Python é amplamente utilizada na comunidade de ciência de dados e, portanto, possui um ecossistema de módulos e ferramentas que utilizáveis em vários projetos. Neste capítulo nos concentraremos nos módulos *Beautiful Soup* e *Requests*.

Beautiful Soup é uma biblioteca de ferramentas Python que permite um retorno rápido de resultados em projetos de *web scraping*. Atualmente disponível como *Beautiful Soup 4* (BS4) e compatível com Python 2.7 e Python 3. *Beautiful Soup* cria uma árvore de análise a partir de documentos HTML e XML. Por sua vez a biblioteca Requests serve para fazer solicitações e requisições em uma base de dados.

A raspagem de dados na internet é totalmente ética e legal, em caso de uso correto da informação coletada e em caso de coleta em sites que não expressam por escrito a proibição a esta prática. O questionamento legal ao procedimento pode advir do mal uso das informações, coleta de dados privados e particulares e vendas a terceiros das informações obtidas.

3 RESULTADOS E DISCUSSÃO

3.1 REALIZANDO WEB SCRAPING COM O PYTHON

Como destacado anteriormente, o objetivo deste capítulo não é ensinar Python, mas cabe lembrar que o mesmo foi usado como base para o desenvolvimento desta pesquisa e que ele facilita muito o trabalho de quem quer ou precisa raspar dados da web. Sendo assim, destaca-se que antes de qualquer passo a ser realizado em *web scraping* é necessário fazer a instalação e a configuração deste software, além da importação e instalação dos módulos ou bibliotecas *Requests* e *Beautifull Soup*, caso você decida realizar o processo de raspagem com o auxílio do Python, é claro.

Além da instalação do software e da importação das bibliotecas é necessário pensar nas linhas de comando ou no programa a ser usado e, para esta etapa do capítulo nos utilizamos de linhas de comando escritas por Lisa Tagliaferri[1] (2023). Aqui o objetivo é verificar se estas linhas de comando funcionam e se conseguimos, de fato, obter as informações desejadas.

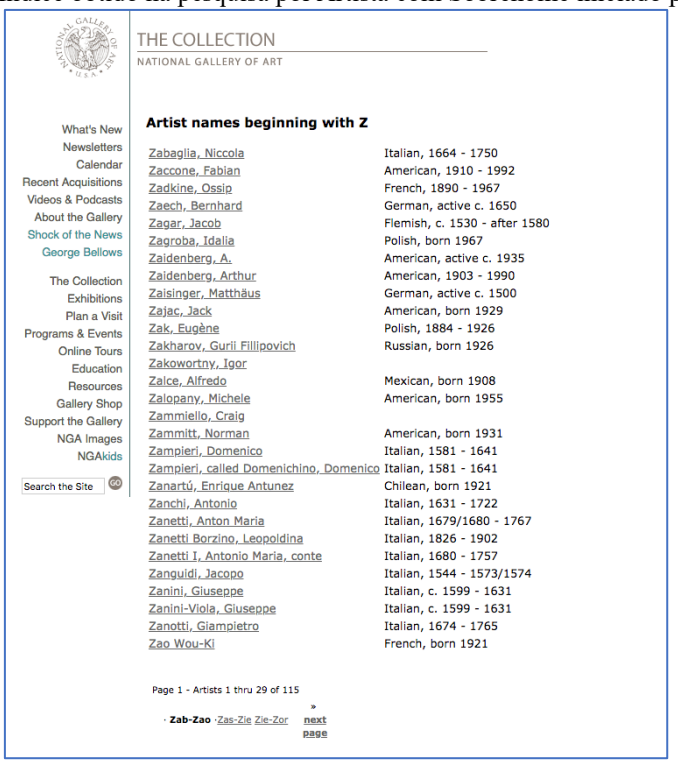
Coletaremos e analisaremos os dados de uma página da web, objetivando gravar as informações em um arquivo do tipo texto ou CSV. A página a ser utilizada é o site oficial da *National Gallery of Art*[2] dos Estados Unidos. O *National Gallery* é um museu de arte localizado na cidade de Whashington, DC, onde estão armazenadas mais de 120.000 peças de arte de mais de 13.000 artistas, que datam desde a Renascença até os dias atuais.

Apesar da grande quantidade de obras e de artistas presentes no museu, inicialmente não precisaremos extrair muitos dados do site, pois o objetivo aqui é demonstrar a efetividade da ferramenta na extração destes dados e não na extração da totalidade dos dados. Com isso, reduziremos o escopo de pesquisa à apenas um artista e dentre todas as letras do alfabeto, decidimos ficar com os artistas cujo sobrenome começam com a letra “Z”.

Dos artistas com sobrenome iniciado por “Z”, o que aparece em primeiro lugar na pesquisa realizada no *Web Archive*[3], onde as obras e artistas do museu estão cadastradas, é Niccola Zabaglia[4] (Figura 01), sendo o último artista citado Václav Zykmund[5]. Ao todo foram encontrados 116 artistas com sobrenome iniciado pela letra “Z”, distribuídos em quatro páginas de resultados (Ex: Zab-Zao, Zas-Zie, Zie-Zor) (Figura 01).

[1] https://www.digitalocean.com/community/users/ltagliaferri, acessado em 20 de novembro de 2023.
[2] https://www.nga.gov/
[3] Índice disponível em: https://web.archive.org/ ou https://archive.org/ ou ainda em https://web.archive.org/web/20170131230332/https://www.nga.gov/collection/an.shtm do “Internet Archive”.
[4] https://web.archive.org/web/20121007172955/http://www.nga.gov/collection/anZ1.htm
[5] https://web.archive.org/web/20121010201041/http://www.nga.gov/collection/anZ4.htm

Figura 01: Índice obtido na pesquisa por Artista com Sobrenome iniciado pela letra Z



THE COLLECTION
NATIONAL GALLERY OF ART

What's New
Newsletters
Calendar
Recent Acquisitions
Videos & Podcasts
About the Gallery
Shock of the News
George Bellows

The Collection
Exhibitions
Plan a Visit
Programs & Events
Online Tours
Education
Resources
Gallery Shop
Support the Gallery
NGA Images
NGAkids

Search the Site GO

Artist names beginning with Z

Zabaglia, Niccola	Italian, 1664 - 1750
Zaccone, Fabian	American, 1910 - 1992
Zadkine, Ossip	French, 1890 - 1967
Zaech, Bernhard	German, active c. 1650
Zagar, Jacob	Flemish, c. 1530 - after 1580
Zagroba, Idalia	Polish, born 1967
Zaidenberg, A.	American, active c. 1935
Zaidenberg, Arthur	American, 1903 - 1990
Zaisinger, Matthäus	German, active c. 1500
Zajac, Jack	American, born 1929
Zak, Eugène	Polish, 1884 - 1926
Zakharov, Gurii Fillipovich	Russian, born 1926
Zakowortny, Igor	
Zalce, Alfredo	Mexican, born 1908
Zalopany, Michele	American, born 1955
Zammiello, Craig	
Zammitt, Norman	American, born 1931
Zampieri, Domenico	Italian, 1581 - 1641
Zampieri, called Domenichino, Domenico	Italian, 1581 - 1641
Zanartú, Enrique Antunez	Chilean, born 1921
Zanchi, Antonio	Italian, 1631 - 1722
Zanetti, Anton Maria	Italian, 1679/1680 - 1767
Zanetti Borzino, Leopoldina	Italian, 1826 - 1902
Zanetti I, Antonio Maria, conte	Italian, 1680 - 1757
Zanguidi, Jacopo	Italian, 1544 - 1573/1574
Zanini, Giuseppe	Italian, c. 1599 - 1631
Zanini-Viola, Giuseppe	Italian, c. 1599 - 1631
Zanotti, Giampietro	Italian, 1674 - 1765
Zao Wou-Ki	French, born 1921

Page 1 - Artists 1 thru 29 of 115

· **Zab-Zao** ·Zas-Zie Zie-Zor » next page

Fonte: Disponível em: https://web.archive.org/web/20121007172955/http://www.nga.gov/collection/anCOM1.htm, acesso em 20 de novembro de 2023.

3.1.1 Passo 1: Importação das bibliotecas

Uma vez que decidimos em qual site realizar a raspagem dos dados e o mais importante, quais dados raspar, cabe prosseguir para aquilo que denominamos de passo 1, que envolve a instalação e configuração do Python e suas bibliotecas.

Para a instalação do Python basta acessar a página https://www.python.org/downloads/ e escolher a versão de interesse (Windows, Linux Mac) fazer o download e proceder a instalação.

O Python se utiliza de IDEs para a inserção das linhas de comando. Dentre as várias IDEs existentes, sugerimos a instalação do Visual Studio, disponível em https://code.visualstudio.com/download.

Uma vez instalado o Python e o Visual Studio, é hora de importarmos as bibliotecas do Python, sendo a primeira delas a Biblioteca *Requests* e a segunda a *BeautifullSoup*. Esse passo deve ser realizado por meio de linhas de comando digitadas no Visual Studio (Figura 02).

Uma biblioteca, nada mais é, que um conjunto de módulos e comandos em Python, organizados e disponibilizados previamente para que se possa importar e economizar tempo na hora de programar.

Uma vez importadas estas duas bibliotecas precisaremos identificar ou atribuir o URL da página inicial, criando uma variável denominada *page,* usando o método *requests.get()* conforme representado na Figura 02.

Figura 02: Importação de Bibliotecas do Python.

```
import requests
from bs4 import BeautifulSoup


# Coletando a primeira página de artistas
page = requests.get('https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ1.htm')
```

Fonte: O Autor (2023).

Após a importação do *beautifullSoup* deve-se criar um objeto ou árvore de análise. Este objeto deverá tomar como argumento o texto da página a ser analisada, ou seja, *page.text = html.parser*. Em outras palavras, é criado um arquivo html, o qual é analisado para a retirada do conteúdo da resposta do servidor (Figura 03).

Figura 03: Criando um objeto BeautifulSoup

```
import requests
from bs4 import BeautifulSoup


page = requests.get('https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ1.htm')

# Criando um objeto BeautifulSoup
soup = BeautifulSoup(page.text, 'html.parser')
```

Fonte: O Autor (2023).

Com a página de interesse coletada é possível dar seguimento ao processo e dar sequência com a extração dos textos de interesse na página da web. Destaca-se que isso só é possível graças ao fato da página ter sido coletada, analisada e configurada como objeto do *BeautifullSoup*.

3.1.2 Passo 2: Extraindo texto de uma página da web

Neste momento nos interessa buscar o nome dos artistas e os links mais relevantes do site, porém, também podem ser coletados outros dados, como a nacionalidade dos artistas e os períodos da história em que viveram.

A questão neste momento para quem não está familiarizado com a temática talvez seja: Como fazer isso? Onde encontramos tais informações? A resposta a estas questões é bem simples. As informações podem ser encontradas no DOM do site ou página da Web.

3.1.2.1 E o que é o DOM?

DOM é a sigla para “Modelo de Objeto de Documento”, que é uma interface de programação para documentos HTML, XML e SVG que fornece uma representação estruturada do documento. Este modelo define os métodos que permitem acesso à estrutura, estilo e conteúdo do site, sendo formado por nós e objetos com várias propriedades e métodos, que fornecem a conexão entre páginas web a scripts ou linguagens de programação.

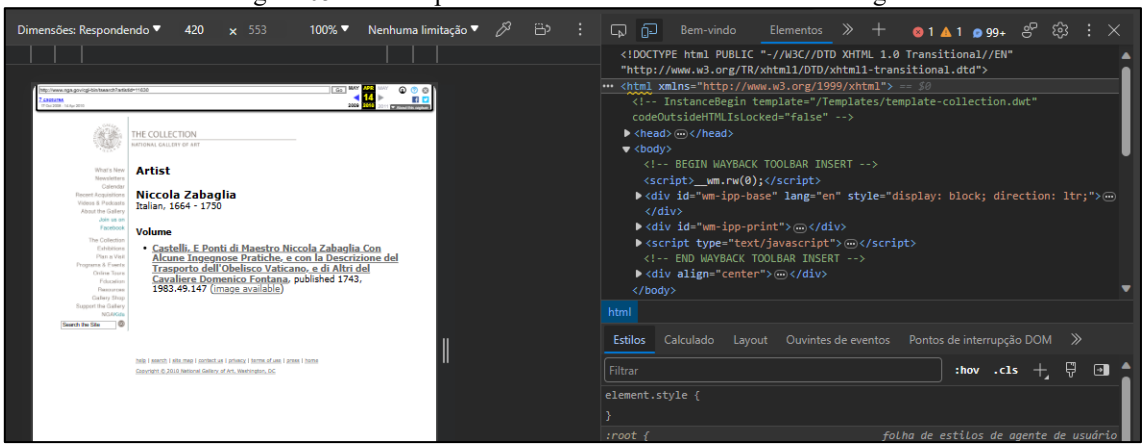
O acesso ao DOM de cada site é realizado por meio do navegador, bastando para isso ter o site aberto e clicar com o botão direito do mouse em uma área qualquer do mesmo e escolher a opção “Inspecionar” ou “Inspecionar Documento”. Na lista de nomes dos artistas, objeto de análise deste capítulo, poderíamos clicar em qualquer nome para obter as informações desejadas, conforme Figura 04.

Figura 04: Representação do acesso ao DOM por meio do item inspecionar

Fonte: o autor (2023).

Depois de clicar no item Inspecionar do menu, as ferramentas de interesse dos desenvolvedores irão aparecer no navegador (Figura 05), o que permitirá acesso às *Tags* ligadas aos nomes presentes na lista de artistas e o html do site.

Figura 05: Item inspecionado: nome do artista Niccola Zabaglia



Fonte: O autor (2023).

Quando inspecionamos a partir do nome de Niccola Zabaglia é possível verificar que a tabela com o nome dos artistas presentes na letra Z está dentro da * * das *tags* onde temos uma classe denominada *“BodyText”*. Associado a este nome também há uma *tag* de link, pois tem uma página que explica quem é o artista (Figura 05).

Na raspagem de dados, interessa extrair o nome dos artistas presentes na tabela, descritos no arquivo *"BodyText "*, para isto deve-se voltar ao Python e ao *BeautifullSoup* com o comandos *find()* e *find_all()*. O programa indicado para isto está demonstrado na Figura 06.

Figura 06: Extração de texto do Site.

```
import requests
from bs4 import BeautifulSoup


# Coletar e analisar a primeira página
page = requests.get('https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ1.htm')
soup = BeautifulSoup(page.text, 'html.parser')

# Extrair todo o texto de BodyText div
artist_name_list = soup.find(class_='BodyText')

# Extrair texto de todas as instâncias de <a> tag de dentro de BodyText div
artist_name_list_items = artist_name_list.find_all('a')
```

Fonte: O autor (2023).

Logo após encontrar o nome dos artistas presentes na lista, precisaremos realizar um *"forloop"*[6] (loop) sobre esses nomes ou variável (artist_name_list_items). A apresentação deles será realizada por meio do *prettify()* (Figura 07), que é usado para transformar a árvore de análise do *BeautifulSoup* em uma *string* unicode formatada.

Figura 07: Criando um looping for.

```
...
artist_name_list = soup.find(class_='BodyText')
artist_name_list_items = artist_name_list.find_all('a')

# Criar loop para imprimir os nomes de todos os artistas
for artist_name in artist_name_list_items:
    print(artist_name.prettify())
```

Fonte: O autor (2023).

A partir deste momento podemos executar o programa, a fim de obter o retorno das páginas onde a informação está armazenada. Estes dados já foram registrados no documento criado (Nome_do_arquivo).

Figura 08: Execução do Arquivo com as informações das quatro páginas

```
1. python nga_z_artists.py (esse é o nome do arquivo[7])
2.
```

Fonte: O autor (2023).

[6] Um loop for é usado para iterar uma sequência (que é uma lista, uma tupla, um dicionário, um conjunto ou uma string). Este looping será realizado tantas vezes, até que seja satisfeita a necessidade do programador.

[7] nga_z_artists.py = Arquivo Python com a lista de artistas com sobrenome iniciado em "Z" do *National Gallery of Art* (nga).

Uma vez realizado o procedimento receberemos o resultado presente na Figura 09.

O que vemos na saída neste momento é o texto completo e as *tags r*elacionadas a todos os nomes dos artistas dentro das *<a>tags* encontradas na * tag* da primeira página, bem como algum texto de link adicional na parte inferior. Como não queremos essas informações extras, vamos trabalhar para removê-las na próxima seção.

Até agora, conseguimos coletar todos os dados de texto do link em uma * * seção da página da web. No entanto, não interessa no momento, links que não façam referência aos nomes de artistas, então vamos trabalhar para remover essa parte.

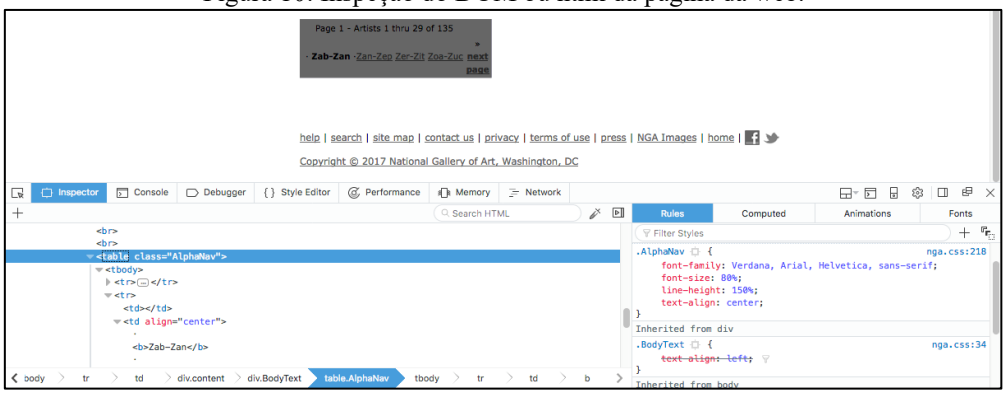
Figura 09: Resultado da Execução do Arquivo com as informações

```
Output
<a href="/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=11630">
 Zabaglia, Niccola
</a>
...
<a href="/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=3427">
 Zao Wou-Ki
</a>
<a href="/web/20121007172955/https://www.nga.gov/collection/anZ2.htm">
 Zas-Zie
</a>

<a href="/web/20121007172955/https://www.nga.gov/collection/anZ3.htm">
 Zie-Zor
</a>

<a href="/web/20121007172955/https://www.nga.gov/collection/anZ4.htm">
 <strong>
  next
  <br/>
  page
 </strong>
</a>
```

Fonte: O autor (2023).

Para remover os links da página, vamos clicar novamente com o botão direito e inspecionar o DOM. Veremos que os links na parte inferior da * * seção estão contidos em uma tabela HTML *<table class ="AlphaNav">* (Figura 10).

Figura 10: Inspeção do DOM ou html da página da web.



Fonte: O autor (2023).

Podemos, portanto, usar Beautiful Soup para encontrar a *AlphaNav classe* e usar o *decompose()* método para remover uma *tag* da árvore de análise e então eliminá-la junto com seu conteúdo. Usaremos a variável *last_links* para fazer referência a esses links inferiores e adicioná-los ao arquivo do programa (Figura 11):

Figura 11: Remoção de Botões de Links

```
import requests
from bs4 import BeautifulSoup


page = requests.get('https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ1.htm')

soup = BeautifulSoup(page.text, 'html.parser')

# Remove bottom links
last_links = soup.find(class_='AlphaNav')
last_links.decompose()

artist_name_list = soup.find(class_='BodyText')
artist_name_list_items = artist_name_list.find_all('a')

for artist_name in artist_name_list_items:
    print(artist_name.prettify())
```

Fonte: O autor (2023).

Agora, ao executarmos o programa com o Python receberemos a saída exposta na Figura 12.

Neste ponto, vemos que a saída não inclui mais os links na parte inferior da página web e agora exibe apenas os links associados aos nomes dos artistas. Até agora, direcionamos os links especificamente para os nomes dos artistas, mas temos dados extras de *tags* que não são interessantes para a análise e devem ser eliminados. Isso é o que será realizado na próxima seção de comandos.

Figura 12: Arquivo de saída após a execução

```
Output
<a href="/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=11630">
 Zabaglia, Niccola
</a>
<a href="/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=34202">
 Zaccone, Fabian
</a>
...
<a href="/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=11631">
 Zanotti, Giampietro
</a>
<a href="/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3427">
 Zao Wou-Ki
</a>
```

Fonte: O autor (2023).

3.1.3 Passo 03: Extraindo o conteúdo de uma tag

Para acessar apenas os nomes dos artistas, desejaremos direcionar o conteúdo das <a>tags em vez de imprimir a tag do link inteira. Podemos fazer isso com *Beautiful Soup's .contents*, que permitirá revisar o *forloop* para que, em vez de imprimir o link inteiro e sua tag, imprimamos apenas a lista com os nomes completos dos artistas:

Com os comandos da Figura 13 estamos *iterando* a lista de nomes chamando o número de índice de cada item. Podemos executar o programa com o Python para visualizar a lista de nomes e receberemos de volta uma lista (Figura 14), com todos os nomes dos artistas disponíveis na primeira página da letra Z.

Se quisermos também capturar os links associados a todos estes nomes poderemos extrair essas URLs a partir de <a>tags de uma página usando get('href'), que é um método do método *Beautiful Soup* (Figura 15).

Pela saída dos links, sabemos que a URL inteira não está sendo capturada, então concatenaremos a *string* do link com a frente da *string* da URL (neste caso https://web.archive.org/) (Figura 15).

Figura 13: Retirada dos Filhos de Tag – nomes dos artistas

```
import requests
from bs4 import BeautifulSoup


page = requests.get('https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ1.htm')

soup = BeautifulSoup(page.text, 'html.parser')

last_links = soup.find(class_='AlphaNav')
last_links.decompose()

artist_name_list = soup.find(class_='BodyText')
artist_name_list_items = artist_name_list.find_all('a')

# Usando .contents para retirar <a> os filhos da tag para artist_name em artist_name_list_items: </a>
    names = artist_name.contents[0]
    print(names)
```

Fonte: O autor (2023).

Figura 14: Saída: lista de nomes dos artistas com sobrenome em “Z”

```
Output
Zabaglia, Niccola
Zaccone, Fabian
Zadkine, Ossip
...
Zanini-Viola, Giuseppe
Zanotti, Giampietro
Zao Wou-Ki
```

Fonte: O autor (2023).

Figura 15: Uso do ‘href’

```
...
for artist_name in artist_name_list_items:
    names = artist_name.contents[0]
    links = 'https://web.archive.org' + artist_name.get('href')
    print(names)
    print(links)
```

Fonte: O autor (2023).

Quando executarmos o programa presente na Figura 15, receberemos como saída os nomes dos artistas e as URLs dos links que apresentam mais informações sobre os artistas (Figura 16).

Figura 16: Saída após a execução do ‘href’: nomes e links

```
Output
Zabaglia, Niccola
https://web.archive.org/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=11630
Zaccone, Fabian
https://web.archive.org/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=34202
...
Zanotti, Giampietro
https://web.archive.org/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=11631
Zao Wou-Ki
https://web.archive.org/web/20121007172955/https://www.nga.gov/cgi-bin/tsearch?artistid=3427
```

Fonte: O autor (2023).

Embora estejamos extraindo informações do site, no momento elas estão apenas sendo apresentadas em nossa janela de terminal, porém esse formato de apresentação não nos interessa muito,

apesar de valiosa. A este capítulo demonstrativo interessa capturar e usar esses dados em outro lugar e isso só será possibilitado a partir da gravação destas informações em um arquivo do tipo texto (CSV).

3.1.4 Passo 04: Gravando os dados em um arquivo CSV

Coletar e visualizar dados que são apresentados apenas em uma janela de terminal IDE Python não é muito prático nem útil. Em contrapartida, arquivos de valores separados por vírgula (CSV) permitem armazenar dados em tabelas de texto simples, além de serem um formato comum para planilhas e bancos de dados. Para este procedimento, em primeiro lugar precisamos importar o módulo integrado do *Python csv* (Figura 17).

Figura 17: Importando um arquivo CSV

```
import csv
```

Fonte: O autor (2023).

Logo após a importação do CSV, pode-se criar e abrir um arquivo chamado *z-artist-names.csv* (ou qualquer nome que queira dar). Na criação deste arquivo serão utilizadas a variável ‘f’ e o modo 'w'. Também devem ser escritos os títulos das linhas superiores: *Name e Link* os quais serão passados ao método *writerow()* como uma lista (Figura 18).

Figura 18: Criando um arquivo CSV com Nome e link dos artistas.

```
f = csv.writer(open('z-artist-names.csv', 'w'))
f.writerow(['Name', 'Link'])
```

Fonte: O autor (2023).

Finalmente, dentro do *forloop*, escreveremos cada linha com os nomes dos artistas e os links a eles associados (Figura 19).

Figura 19: Criação de linhas com nome e link dos artistas

```
f.writerow([names, links])
```

Fonte: O autor (2023).

Você pode ver as linhas para cada uma dessas tarefas na Figura 20.

Figura 20: Adicionando o nome de cada artista e o link associado à uma linha

nga_z_artists.py

```
import requests
import csv
from bs4 import BeautifulSoup


page = requests.get('https://web.archive.org/web/20121007172955/http://www.nga.gov/collection/anZ1.htm')

soup = BeautifulSoup(page.text, 'html.parser')

last_links = soup.find(class_='AlphaNav')
last_links.decompose()

# Criar um arquivo para gravar e adicionar linha de cabeçalhos
f = csv.writer(open('z-artist-names.csv', 'w'))
f.writerow(['Name', 'Link'])

artist_name_list = soup.find(class_='BodyText')
artist_name_list_items = artist_name_list.find_all('a')

for artist_name in artist_name_list_items:
    names = artist_name.contents[0]
    links = 'https://web.archive.org' + artist_name.get('href')


    # Adicione o nome de cada artista e o link associado a uma linha
    f.writerow([names, links])
```

Fonte: O autor (2023).

Se executarmos o programa neste momento, não receberemos nenhuma saída como resposta, em vez disso, um arquivo será criado no diretório em que você está trabalhando, chamado z-artist-names.csv (ou qualquer nome que você queira dar).

No teste realizado para este capítulo foi criado um arquivo CSV legível em Excel. Cabe aqui lembrar aos leitores que estamos realizando esta raspagem em um site onde estão registradas as informações do National Gallery of Arts e que isso pode provocar a presença de links não executáveis na saída do programa, como é o caso do link da saída para o artista "Zadkine, Ossip" presente na Figura 21.

Figura 21: Nomes e links dos artistas presentes na saída do terminal

z-artist-names.csv

```
Name, Link
"Zabaglia, Niccola",https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=11630
"Zaccone, Fabian",https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=34202
"Zadkine, Ossip",https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3475w
...
```

Fonte: O autor (2023).

No arquivo CSV essas informações poderão se parecer com uma planilha (Figura 22).

Com a criação do arquivo CSV, poderemos usar os dados de maneira mais significativa, já que as informações coletadas agora estão armazenadas em nosso computador.

Figura 22: Planilha criada a partir do arquivo CSV.

Name	Link
Zabaglia, Niccola	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=11630
Zaccone, Fabian	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=34202
Zadkine, Ossip	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3475
Zaech, Bernhard	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=25135
Zagar, Jacob	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=2298
Zagroba, Idalia	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=23988
Zaidenberg, A.	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=8232
Zaidenberg, Arthur	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=34154
Zaisinger, Matthäus	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=4910
Zajac, Jack	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3450
Zak, Eugène	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=1986
Zakharov, Gurii Fillipovich	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3451
Zakowortny, Igor	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=20099
Zalce, Alfredo	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3452
Zalopany, Michele	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=34309
Zammiello, Craig	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=27191
Zammitt, Norman	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=5846
Zampieri, Domenico	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3941
Zampieri, called Domenichino, Domenico	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3941
Zanartú, Enrique Antunez	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3453
Zanchi, Antonio	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=35173
Zanetti, Anton Maria	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=11133
Zanetti Borzino, Leopoldina	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3455
Zanetti I, Antonio Maria, conte	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3454
Zanguidi, Jacopo	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=961
Zanini, Giuseppe	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=11597
Zanini-Viola, Giuseppe	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=11597
Zanotti, Giampietro	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=11631
Zao Wou-Ki	https://web.archive.org/web/20121007172955/http://www.nga.gov/cgi-bin/tsearch?artistid=3427

Fonte: O autor (2023).

3.1.5 Passo 05: Recuperando Informações de páginas relacionadas

Criamos um programa para extrair informações da primeira página da lista de artistas cujos sobrenomes começam com a letra **Z**, entretanto, existem 4 páginas no total desses artistas disponíveis no site. Para coletar todas essas páginas, podemos realizar mais iterações com *forloops*. Isso revisará a maior parte do código que escrevemos até agora, mas empregará conceitos semelhantes.

Para começar, devemos criar uma lista que irá conter as páginas (Figura 23).

Figura 23: criação da lista contendo as páginas da web referente aos artistas

```
pages = []
```

Fonte: O autor (2023).

Preencheremos esta lista inicializada com um *forloop* (Figura 24).

Figura 24: Percorrendo as páginas de 1 a 5 para confecção da lista de páginas.

```
for i in range(1, 5):
    url = 'https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ' + str(i) + '.htm'
    pages.append(url)
```

Fonte: O autor (2023).

Como existem 4 páginas para a letra Z, construímos o *forloop* presente na Figura 24 com um intervalo de 1 a 5 para que ele percorra cada uma das 4 páginas. Para este site em específico, os URLs começam com a string *https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ* e depois são

seguidos por um número para a página (que será o número inteiro i do *forloop* que convertemos em uma *string*) e terminam com *.htm*.

Concatenaremos essas *strings* e, em seguida, anexaremos o resultado à *pages* lista. Além deste *loop*, devemos usar um outro *loop* que percorrerá cada uma das páginas. O código neste *forloop* será semelhante ao código que criamos até agora, pois está executando a tarefa que concluímos para a primeira página dos artistas da letra Z, isso será realizado para cada uma das quatro páginas e como colocamos o programa original no segundo *forloop*, agora temos o *loop* original como um *loop aninhado for*[8].

No código (Figura 25), você deve ver que o primeiro *forloop* está iterando pelas páginas e o segundo *forloop* está coletando dados de cada uma dessas páginas e, em seguida, adicionando os nomes dos artistas e links linha por linha em cada linha de cada página.

Esses dois *forloops* vêm abaixo das *import* instruções, a criação e gravador do arquivo CSV (com a linha para escrever os cabeçalhos do arquivo) e a inicialização da *pages* variável (atribuída a uma lista). Os dois *forloops* ficarão assim (Figura 25):

Figura 25: Execução dos dois *forloops* percorrendo as 4 páginas da web

```
pages = []

for i in range(1, 5):
    url = 'https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ' + str(i) + '.htm'
    pages.append(url)

for item in pages:
    page = requests.get(item)
    soup = BeautifulSoup(page.text, 'html.parser')

    last_links = soup.find(class_='AlphaNav')
    last_links.decompose()

    artist_name_list = soup.find(class_='BodyText')
    artist_name_list_items = artist_name_list.find_all('a')

    for artist_name in artist_name_list_items:
        names = artist_name.contents[0]
        links = 'https://web.archive.org' + artist_name.get('href')

        f.writerow([names, links])
```

Fonte: O autor (2023).

Dentro do contexto maior do arquivo de programação, o código completo fica assim (Figura 26):

[8] É o looping que ocorre sem interrupções. Exemplo: os meses do ano. Começam em janeiro e vão até dezembro, depois começa tudo novamente.

Figura 26: Código Completo

nga_z_artists.py

```
import requests
import csv
from bs4 import BeautifulSoup


f = csv.writer(open('z-artist-names.csv', 'w'))
f.writerow(['Name', 'Link'])

pages = []

for i in range(1, 5):
    url = 'https://web.archive.org/web/20121007172955/https://www.nga.gov/collection/anZ' + str(i) + '.htm'
    pages.append(url)


for item in pages:
    page = requests.get(item)
    soup = BeautifulSoup(page.text, 'html.parser')

    last_links = soup.find(class_='AlphaNav')
    last_links.decompose()

    artist_name_list = soup.find(class_='BodyText')
    artist_name_list_items = artist_name_list.find_all('a')

    for artist_name in artist_name_list_items:
        names = artist_name.contents[0]
        links = 'https://web.archive.org' + artist_name.get('href')

        f.writerow([names, links])
```

Fonte: O autor (2023).

Pode demorar um pouco para criar o arquivo CSV, uma vez que o programa é longo, mas uma vez criado, a produção estará completa, mostrando os nomes dos artistas e seus links associados desde Zabaglia, Niccola até Zykmund, Václav.

4 CONSIDERAÇÕES FINAIS

Os exercícios realizados ao longo deste capítulo, utilizando-se das linhas de código criadas por Lisa Tagliaferri (2023), permitiram verificar que a raspagem de dados em sites da web é perfeitamente realizável a partir do Python e seus códigos.

Os resultados obtidos foram plenamente satisfatórios, uma vez que conseguimos obter, de forma automática, as informações relativas aos artistas com sobrenome iniciado em Z e que tem obras na *National Gallery of Art*. Este procedimento poderia ser realizado para qualquer lista de artistas, isto é, para qualquer letra entre A e Z. não é aconselhável a realização da pesquisa para artistas isolados, devido ao trabalho que dá e à facilidade que seria entrar no site e digitar o nome deste artista.

A recuperação dos nomes e links para estes nomes de artistas permite o acesso à outras informações destes artistas como, por exemplo, época em que viveram, obras criadas e local de nascimento.

Antes de pensar em realizar o procedimento de web scraping, torna-se necessário levar em consideração os servidores dos quais obteremos as informações. É essencial verificar se o site possui termos de serviço ou termos de uso relacionados a web scraping, ou seja, se eles permitem ou proíbem de forma explicita (escrita) ou se o site possui uma API que permite capturar dados, antes de extraí-los.

Deve-se lembrar que a busca constante e incessante de dados de um site fará com que o mesmo o entenda como um robô ou um malware que deve ser bloqueado e certamente seu IP será bloqueado. Por isso não devemos acessar continuamente os servidores para coletar dados, não devemos sobrecarregar os servidores de outra pessoa.

Uma boa prática de web scraping, e de evitar ser bloqueado, é incluir um cabeçalho que contenha seu nome e e-mail para que um site possa identificá-lo e entrar em contato caso tenha alguma dúvida. Um exemplo de cabeçalho que você pode usar com a biblioteca *Python Requests* consta na Figura 27:

Figura 27: Criação de um cabeçalho para identificação pessoal durante o Scraping

```
import requests

headers = {
    'User-Agent': 'Your Name, example.com',
    'From': 'email@example.com'
}

url = 'https://example.com'

page = requests.get(url, headers = headers)
```

Fonte: O autor (2023).

Neste momento, cabe lembrar que o uso das ferramentas do *BeautifullSoup* e do *Requests* é uma boa maneira de realizar o web scraping. E que o salvamento dos resultados em um arquivo de texto CSV legível em Excel ou outro aplicativo permite a análise mais aprofundada dos dados e informações recolhidas.

REFERÊNCIAS

BARBOSA, A. B. G.; CAVALCANTI, A. B. Web Scraping e Análise de Dados. Anais do V CONAPESC, Campina Grande: Realize Editora, 10 dez. 2020. Disponível em: https://www.editorarealize.com.br/index.php/artigo/visualizar/73189. Acesso em: 13 nov. 2023.

MITCHELL, R. E. *web scraping with Python: collecting more data from the modern web*. Second Edition e. Sebastopol, AC: O'Reilly Media, 2018.

MOOERS, C. N. Zatocoding applied to mechanical organization of knowledge. American Documentation, [*s.l.*], v. 2, n. 1, p. 20-32, 1951. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.5090020107. Acesso em: 25 novembro de 2023.

NGA - NATIONAL GALLERY OF ARTS. Disponível em: https://www.nga.gov/, acesso em 25 de novembro de 2023.

TAGLIAFERRI, L. *How To Scrape Web Pages with Beautiful Soup and Python 3*. https://www.digitalocean.com/community/tutorials/how-to-scrape-web-pages-with-beautiful-soup-and-python-3, Acesso em 20 de novembro de 2023.