Web Scraping - Playwright

InfoUma 2024/25 **Andrea Marchetti**

Cos’è Playwright

Sistema di Automazione di browser (**Chrome, Edge, Firefox, Safari**)

Funzionamento **headless** o con **interfaccia grafica**

Perfetto per **web scraping** e **test automatici**

Supporto a **Python**, **JavaScript**, **Java**, e altri linguaggi

Cos’è Playwrigth

Posso scrivere un **programma** che simula il comportamento umano per

- aprire una finestra del browser,
- navigare in una pagina web
- compilare campi di input
- fare click sui pulsanti
- gestire le finestre di dialogo

Storia di Playwright

- **2017**: **Google** sviluppa **Puppeteer**, una libreria per automatizzare esclusivamente Chrome.
- **2020**: **Microsoft** introduce **Playwright**, derivato da Puppeteer, che supporta più browser (Chrome, Edge, Firefox, Safari).
- Rapidamente adottato dalla community grazie alle prestazioni elevate e alla semplicità d’uso.

Playwright Vs Selenium

Prestazioni elevate: È generalmente più veloce e meno pesante di Selenium.

Facilità d'uso: Configurazione più semplice e intuitiva.

Gestione automatica di caricamenti e attese: Riduce la necessità di espliciti wait per elementi.

Multi-browser integrato: Non richiede setup aggiuntivi per browser diversi.

Installazione

Assicurati di avere Python 3.7+ installato

Installa Visual Studio Code da code.visualstudio.com

Installa l’estensione Python in VS Code

Installazione

Da Vs Code apri un terminale

```
python -m venv .venv # Crea un ambiente virtuale chiamato ".venv"
.venv\Scripts\activate # Attiva l'ambiente
pip install playwright # installa Playwright
playwright install chromium # installa Chrome/Edge
```

Usa l'ambiente virtuale in VS Code

- Apri il file .venv/bin/python (Mac/Linux) o venv\Scripts\python.exe (Windows).
- Premi Ctrl + Shift + P e digita "Python: Select Interpreter".
- Scegli l'interprete Python dell'ambiente virtuale (venv).

Installazione

```
pip install playwright
playwright install
```

Il primo comando installa la libreria Python per interagire con Playwright.

Il secondo comando installa i browser necessari al funzionamento di Playwright.

Test installazione

```
from playwright.sync_api import sync_playwright

with sync_playwright() as p:
    browser = p.chromium.launch(headless=True)
    page = browser.new_page()
    page.goto("https://example.com")
    print("Titolo della pagina:", page.title())
    browser.close()
```

Demo

Dataroom di Milena Gabanelli

https://www.corriere.it/dataroom-milena-gabanelli/

Primo Esempio

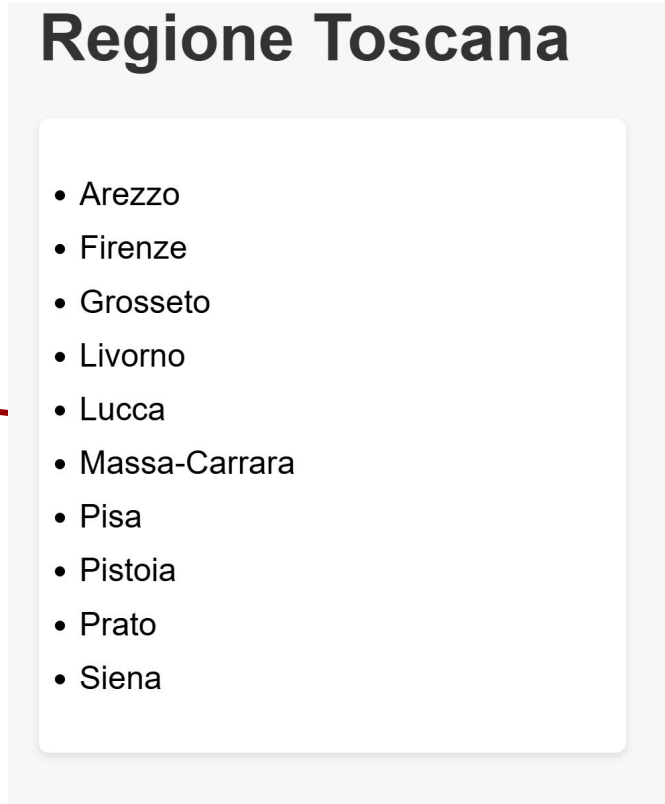
```python
from playwright.sync_api import sync_playwright

def scrape_page(url):
    with sync_playwright() as p:
        browser = p.chromium.launch(headless=False)
        page = browser.new_page() # Apre una nuova pagina
        page.goto(file_path) # Vai su URL

        # trova tutti i tag <li> nella pagina
        lista_elementi = page.locator('li')

        # Cicla e stampa il testo di ogni elemento <li>
        for i in range(lista_elementi.count()):
            provincia = lista_elementi.nth(i).inner_text()
            print(f"Elemento {i+1}: {provincia}")

        # chiusura del browser
        browser.close()
```



Note

Il metodo `locator('li')` seleziona tutti gli elementi <li> presenti sulla pagina.

Utilizzando il metodo `.count()` ottieni il numero totale degli elementi trovati.

Puoi accedere ad ogni elemento individualmente con `.nth(indice)`.

Modalità Headless

```
browser = p.chromium.launch(headless=False|True)
```

headless = True

utile in fase iniziale per testare cosa fa l'applicazione

headless=False

una volta che l'applicazione è pronta

Metadati della pagina

```
page.title()                                # Titolo della pagina

page.url                                    # URL della pagina

page.content()                              # sorgente html della pagina

page.screenshot(path="screenshot.png") # Salva uno screenshot della pagina
```

Tutti i modi per individuare elementi HTML con Playwright

Modi x individuare elementi HTML

Sintassi Generale

```
page.locator('selettore')
```

Modi x individuare elementi HTML

1 Selettori CSS

```
page.locator('div.classe')
page.locator('ul#idlista.classe > li.classe')
```

Modi x individuare elementi HTML

2 Selettori XPath

```
page.locator('//ul[@id="Toscana"]/li')
page.locator('//li')
```

Modi x individuare elementi HTML

3 Selezione per testo

```
page.locator("text=Firenze")
page.locator("li", has_text="Firenze")
page.get_by_text("Firenze", exact="True")
```

Modi x individuare elementi HTML

4 Selezione tramite ruoli ARIA (Accessibilità)

```
# es. Ruolo: button, link, checkbox, radio, listitem ecc.
page.get_by_role('listitem', name='Firenze')
```

Modi x individuare elementi HTML

5 Selezione per placeholder (per input)

per campi input con placeholder definito

```
<!-- Input
<div class
   <label
   <input
</div>
```


```
nome@example.com">
```

```
page.get_by_placeholder('nome@example.com')
```

Modi x individuare elementi HTML

6 Selezione tramite label

per elementi associati a una label esplicita

```
page.get_by_label('Provincia')
```

Modi x individuare elementi HTML

7 Selezione tramite Alt-text (per immagini)

per elementi immagine <img>

```
page.get_by_alt_text('Logo')
```

Modi x individuare elementi HTML

8 Selezione per attributi personalizzati (data-attributes)

```
<li data-provincia="Firenze">Firenze</li>
```

```
page.locator('li[data-provincia="Firenze"]')
```

Modi x individuare elementi HTML

9 Selezione del primo, ultimo, n-esimo

```
page.locator('ul#Toscana > li').first
page.locator('ul#Toscana > li').last
page.locator('ul#Toscana > li').nth(3)   # terzo elemento
```

Modi x individuare elementi HTML

10 Concatenazione

```
page.locator('ul#Toscana').locator(':text("Firenze")')
page.get_by_text('Firenze').locator('xpath=..')  # genitore
```

Conclusioni

Selettore CSS: standard, semplice e leggibile.

XPath: potente, ma meno leggibile, per selezioni complesse.

Concatenazioni: per elementi annidati

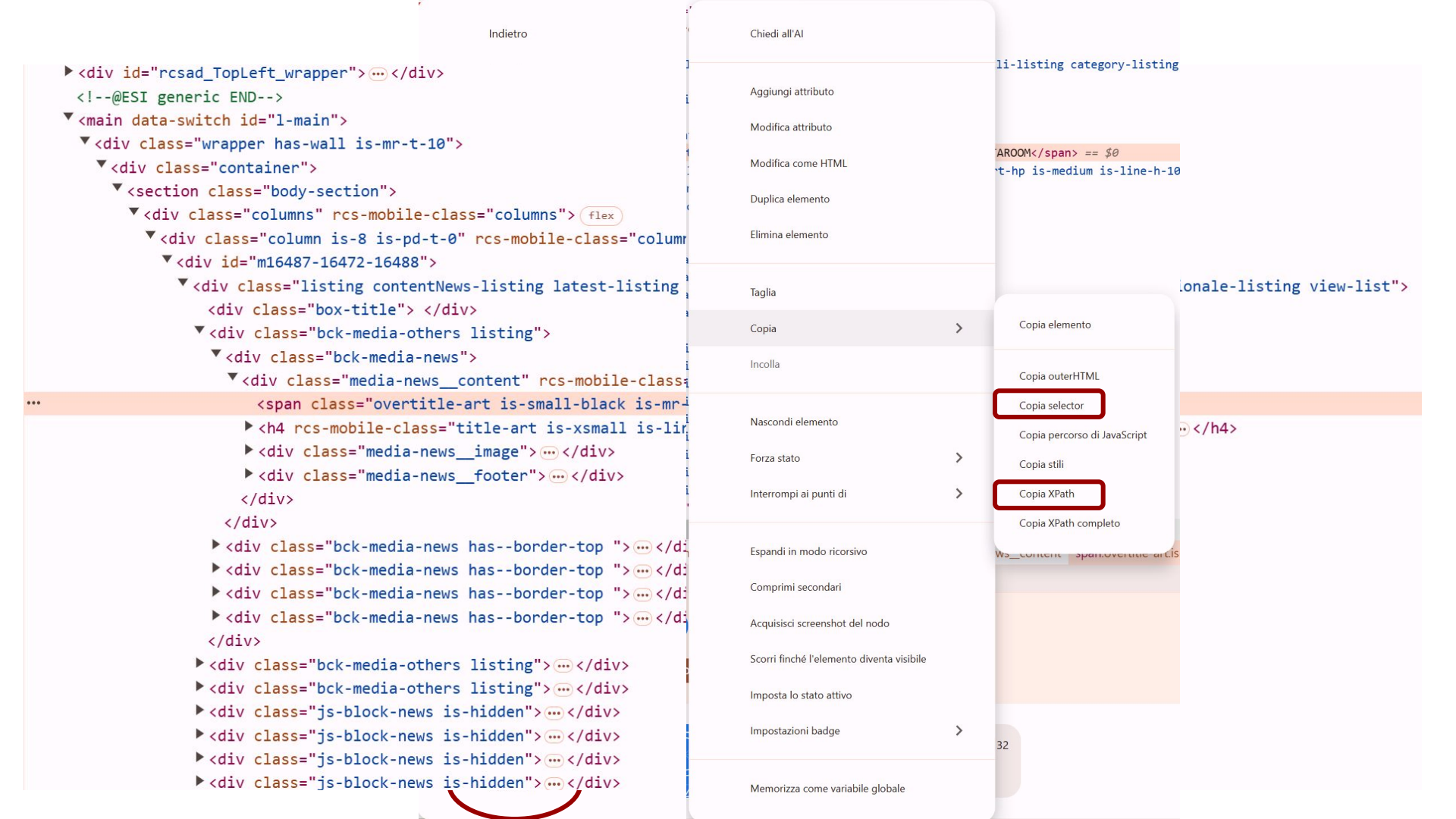
Documentazione CSS Selector, XPath

CSS Selectors Reference w3schools

Try CSS Selector Test di espressioni XPath

XPath Syntax w3schools

XPath Test di espressioni XPath


Indietro
Chiedi all'AI
Aggiungi attributo
Modifica attributo
Modifica come HTML
Duplica elemento
Elimina elemento
Taglia
Copia
Incolla
Nascondi elemento
Forza stato
Interrompi ai punti di
Espandi in modo ricorsivo
Comprimi secondari
Acquisisci screenshot del nodo
Scorri finché l'elemento diventa visibile
Imposta lo stato attivo
Impostazioni badge
Memorizza come variabile globale
Copia elemento
Copia outerHTML
Copia selector
Copia percorso di JavaScript
Copia stili
Copia XPath
Copia XPath completo
...
<!--@ESI generic END-->
<main data-switch id="l-main">

<section class="body-section">
flex
<div class="column is-8 is-pd-t-0" rcs-mobile-class="columm

<div class="listing contentNews-listing latest-listing

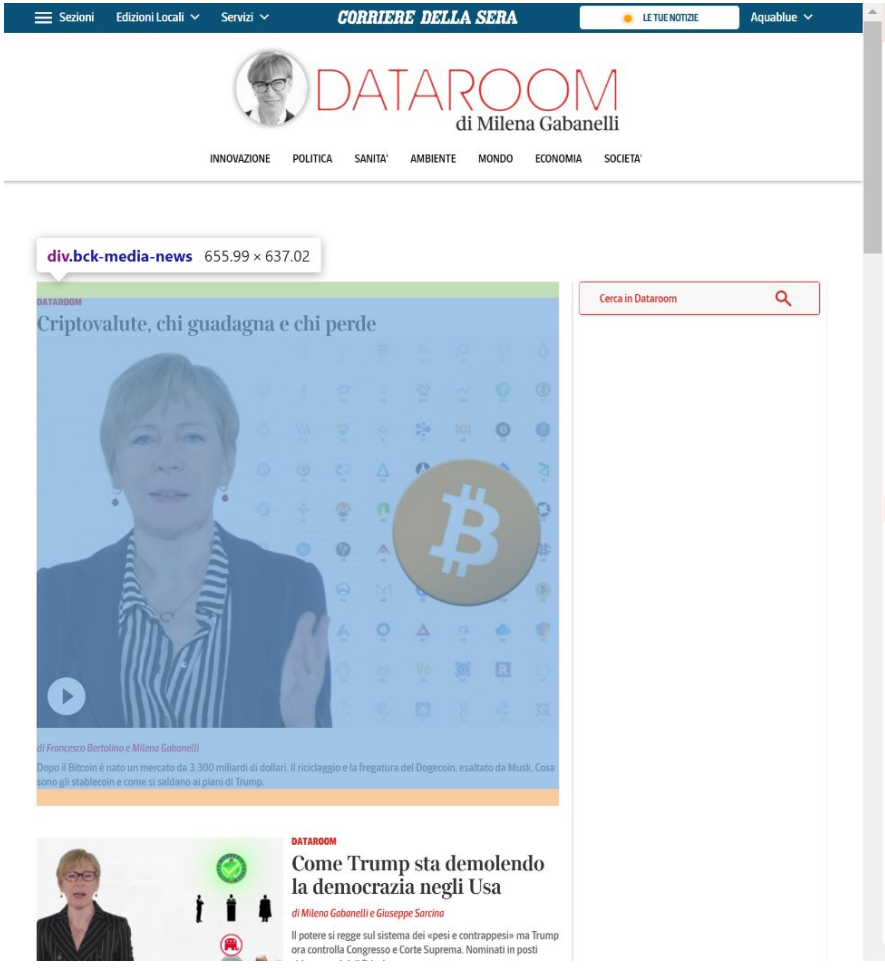
<div class="media-news__content" rcs-mobile-class
<span class="overtitle-art is-small-black is-mr-
<h4 rcs-mobile-class="title-art is-xsmall is-lin
...
...

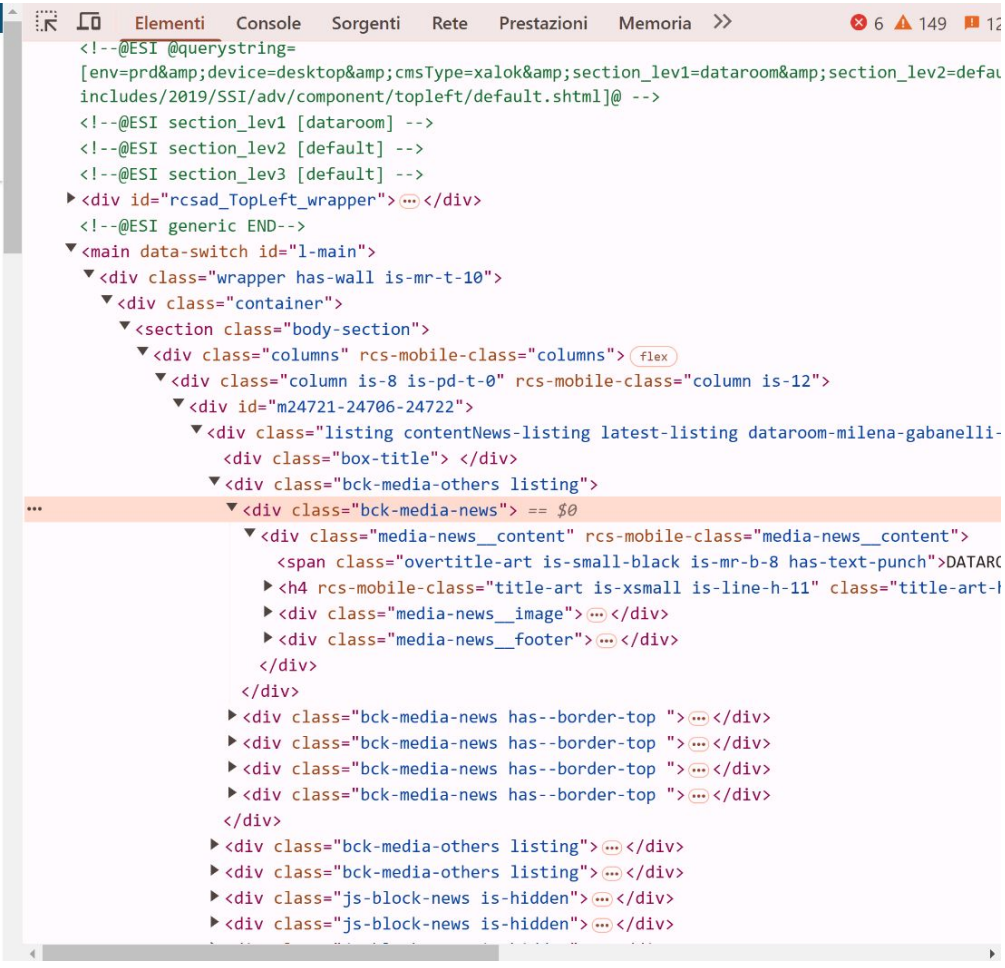
...</di
...</di
...</di
...</di

...
...
...
...
...
...
li-listing category-listing
AROOM == $0
t-hp is-medium is-line-h-10
onale-listing view-list">
</h4>
32

#m24721-24706-24722 > div > div:nth-child(2) > div:nth-child(1) > div > span

//*[@id="m24721-24706-24722"]/div/div[2]/div[1]/div/span


Sezioni
Edizioni Locali
Servizi
CORRIERE DELLA SERA
LE TUE NOTIZIE
Aquablue
DATAROOM
di Milena Gabanelli
INNOVAZIONE
POLITICA
SANITA'
AMBIENTE
MONDO
ECONOMIA
SOCIETA'
div.bck-media-news 655.99 × 637.02
DATAROOM
Criptovalute, chi guadagna e chi perde
di Francesco Bertolino e Milena Gabanelli
Dopo il Bitcoin è nato un mercato da 3.300 miliardi di dollari. Il riciclaggio e la fregatura del Dogecoin, esaltato da Musk. Cosa sono gli stablecoin e come si saldano ai piani di Trump.
Cerca in Dataroom
DATAROOM
Come Trump sta demolendo la democrazia negli Usa
di Milena Gabanelli e Giuseppe Sarcina
Il potere si regge sul sistema dei «pesi e contrappesi» ma Trump ora controlla Congresso e Corte Suprema. Nominati in posti


Elementi
Console
Sorgenti
Rete
Prestazioni
Memoria
6
149
<!--@ESI @querystring=
[env=prd&device=desktop&cmsType=xalok&section_lev1=dataroom&section_lev2=defau
includes/2019/SSI/adv/component/topleft/default.shtml]@ -->
<!--@ESI section_lev1 [dataroom] -->
<!--@ESI section_lev2 [default] -->
<!--@ESI section_lev3 [default] -->
...
<!--@ESI generic END-->
<main data-switch id="1-main">

<section class="body-section">
flex

<div class="listing contentNews-listing latest-listing dataroom-milena-gabanelli-

 == $0

DATARO
<h4 rcs-mobile-class="title-art is-xsmall is-line-h-11" class="title-art-
...
...

...
...
...
...

...
...
...
...

Interazione con Form elements

Inserisce "Testo di esempio".
page.locator("input#text-input").**fill**("Testo di esempio")

Seleziona l'opzione con value "opzione2".
page.locator("select#select").**select_option**("opzione2")

Seleziona il radio button con id "radio2".
page.locator("input#radio2").**check**()

Clicca sul pulsante di submit per inviare il form.
page.locator("button[type='submit']").**click**()

Spiegazione Demo

Dataroom di Milena Gabanelli

DataRoom

```
from playwright.sync_api import sync_playwright

url = "https://www.corriere.it/dataroom-milena-gabanelli/"

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False)
    page = browser.new_page() # Apre una nuova pagina
    page.goto(url) # Vai su URL

    # Accetta cookie (se appare popup)
    try:
        page.click('button:text("ACCETTA E CONTINUA")')
    except:
        pass  # ignora se il popup non appare
```

DataRoom

```
# Trova tutti i tag <div> con classe "bck-media-news"
news = page.locator('div.bck-media-news')

# Cicla su news e crea un record per ogni new
results = []
for i in range(news.count()):
    results.append({
        'titolo': news.nth(i).locator('h4'             ).inner_text().strip(),
        'url'   : news.nth(i).locator('h4>a'           ).get_attribute('href'),
        'autori': news.nth(i).locator('span.author-art').inner_text().strip(),
        'desc'  : news.nth(i).locator('p.subtitle-art' ).inner_text().strip()
    })
```

DataRoom

```
    import pandas as pd

    # Salva i risultati su file CSV con pandas
    df = pd.DataFrame(results)
    df.to_csv('notizie.csv', index=False)

    browser.close() # Chiude il browser
```

Risorse utili e documentazione

Documentazione ufficiale:
playwright.dev/python

Repository GitHub:
github.com/microsoft/playwright-python