

Creazione di PDF da file HTML in Python con Pyppeteer

In questo articolo, esploreremo un semplice script Python per generare file PDF a partire da file HTML usando la libreria Pyppeteer, un'implementazione di Puppeteer per Python. Puppeteer è uno strumento sviluppato da Google che permette di controllare un browser tramite codice, offrendo la possibilità di automatizzare molte operazioni, tra cui il rendering di pagine web e la creazione di PDF. Vediamo come funziona il codice e come possiamo adattarlo alle nostre esigenze.

Pyppeteer è una libreria che replica le funzionalità di Puppeteer (originariamente sviluppato per JavaScript) in Python, permettendo di controllare il browser Chrome o Chromium. La sua capacità di generare PDF, screenshot e automatizzare interazioni con le pagine è particolarmente utile in contesti come la generazione di report, l'archiviazione di pagine web e lo scraping.

Prima di eseguire lo script, è necessario installare Pyppeteer. È possibile installarla tramite pip:

```
pip install pyppeteer
```

Vediamo ora il codice Python che consente di convertire file HTML in PDF.

```
import asyncio
import os
from pyppeteer import launch
```

```

async def create_pdf_from_html(html, pdf_path):
    browser = await launch()
    page = await browser.newPage()

    await page.setContent(html)
    await page.pdf({'path': pdf_path, 'format': 'A4'})
    await browser.close()

async def main():
    html_files_dir = './html'
    pdf_files_dir = './pdf'
    for html_file in os.listdir(html_files_dir):
        if html_file.endswith('.html'):
            html_path = os.path.join(html_files_dir,
html_file)
            pdf_path = os.path.join(pdf_files_dir,
html_file.replace('.html', '.pdf'))
            html = open(html_path).read()
            await create_pdf_from_html(html, pdf_path)

asyncio.get_event_loop().run_until_complete(main())

```

Il codice è composto da due funzioni asincrone: `create_pdf_from_html` e `main`.

La funzione `create_pdf_from_html` è il cuore del programma. Accetta due parametri:

- `html`: il contenuto HTML da convertire in PDF.
- `pdf_path`: il percorso dove salvare il file PDF generato.

La funzione crea un'istanza del browser, carica la pagina HTML e genera un PDF nel formato A4.

```

async def create_pdf_from_html(html, pdf_path):
    browser = await launch() # Avvia un'istanza del
    browser
    page = await browser.newPage() # Crea una nuova
    pagina

    await page.setContent(html) # Imposta il contenuto
    HTML
    await page.pdf({'path': pdf_path, 'format': 'A4'})
    # Genera il PDF in formato A4
    await browser.close() # Chiude il browser

```

La funzione `main` gestisce l'elaborazione di più file HTML. Definisce le directory di input (`./html`) e output (`./pdf`), quindi esegue un ciclo su tutti i file HTML presenti nella directory `html_files_dir`. Per ogni file, legge il contenuto HTML e genera un file PDF nella directory di destinazione.

```

async def main():
    html_files_dir = './html' # Directory dei file
    HTML
    pdf_files_dir = './pdf' # Directory dei file PDF
    for html_file in os.listdir(html_files_dir): #
    Scorre i file nella cartella HTML
        if html_file.endswith('.html'): # Verifica
        l'estensione
            html_path = os.path.join(html_files_dir,
            html_file) # Percorso del file HTML
            pdf_path = os.path.join(pdf_files_dir,
            html_file.replace('.html', '.pdf')) # Percorso del
            PDF

```

```
        html = open(html_path).read() # Legge il
contenuto HTML
        await create_pdf_from_html(html, pdf_path)
# Converte in PDF
```

L'esecuzione dello script è gestita tramite l'uso di un loop di eventi asincrono. Il metodo `asyncio.get_event_loop().run_until_complete(main())` avvia l'esecuzione di `main` e completa tutte le operazioni asincrone definite.

Per far funzionare correttamente lo script, è importante organizzare i file in due cartelle:

- **./html**: dove vanno inseriti i file HTML che si desidera convertire in PDF.
- **./pdf**: dove verranno salvati i file PDF generati.

La generazione di PDF a partire da HTML è una tecnica utile in molte applicazioni, come:

- Generazione automatica di report.
- Archiviazione di pagine web per future consultazioni.
- Trasformazione di contenuti dinamici in documenti stampabili.

Questo approccio è particolarmente vantaggioso per chi lavora con contenuti dinamici su web e ha necessità di automatizzare la generazione di documenti in formati stabili e pronti per la stampa.

Conclusioni

In questo articolo abbiamo illustrato come usare Pyppeteer per creare PDF a partire da file HTML con Python. Abbiamo analizzato il codice in dettaglio, evidenziando come le funzionalità asincrone di Python e Pyppeteer

permettano di gestire facilmente operazioni di automazione del browser e di generazione di documenti.