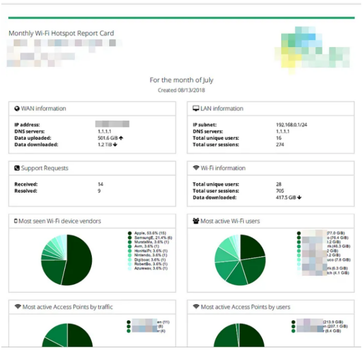
Create PDF from Webpage in Python

Nikita Tonkoshkur · Follow
2 min read · Aug 4, 2021



So, are you here because you need to convert web pages to PDFs and do it fast while being able to maintain page styles and JavaScrtip rendered content?

Recently I faced this task. I quickly understood that using something like **pdfkit** was not an option for me. It does not give full control over the contents of the page and the way the result PDF would look like. Also, if the page that you want to convert to PDF is rendered via JavaScript you won't see the same result as you would through your browser. And **pdfkit** is kinda slow .

My solution includes using Selenium to achieve fast and controllable PDF generation. If that detail does not scare you, let's start.

. . .

Prerequisites:
You will need to have Chrome browser installed on your machine, it can be Mac OS, Windows, or Linux.
Also, you need to install *webdriver-manager* and *selenium* packages. *webdriver-manager* takes care of fetching the latest version of chromedriver, so don't need to worry about it. This approach works for years in production for me with no issues with chromedriver and chrome being incompatible.

```
pip install webdriver-manager selenium
```

. . .

This approach is based on using Chrome`s built-in print function and invoking it through browser API.

I will provide a full code snippet and explain what is going on:

All the heavy lifting is done by `_generate_pdfs` function, which

1. Iterates over the URLs
2. Calls `_get_pdf_from_url` on them. This function uses build-in Chrome devtools and invokes `Page.printToPDF` on them by calling `_send_devtools` helper function.
3. Returns PDF file as bytes.

Example use:

```
pdf_file = PdfGenerator(['https://medium.com']).main()

# save pdf to file
with open('medium_site.pdf', "wb") as outfile:
    outfile.write(pdf_file[0].getbuffer())
```

After you've set everything up, you can start configuring PDF generator for your needs.

1. You can adjust print options by changing `print_options` class attribute. Check printToPDF Chrome devtools documentation for all the options.
2. You can change chromedriver options within the `main` . In the example code, I've set `webdriver_options.add_argument('--headless')` so that chromedriver would open in headless mode(not actually opening the browser)
3. You can change chromedriver properties within `_get_pdf_from_url` options. For example, you can set the screen resolution `self.driver.set_window_size(1920, 1080)`
4. You can adjust the timeout between opening the page and creating a PDF. For example, to create a PDF of medium.com, I had to set `time.sleep(2)` to 1 second, to allow JavaScript to render the page.

You can use `_get_pdf_from_url` separately by building your code around it, even change options on the fly depending on the URL.

This PDF generator works quite fast, for my use case I can generate 500 PDF reports in under 20 minutes on the production server running Ubuntu.

Python Pdf Selenium Programming Pdf Generation