Python

The PyPI package `weblitz` is Weblitz for Python: HTML, a file or a web address to a PDF, an Excel, Word or PowerPoint file, or a picture, printed as Chrome prints them, with no browser to install, start or drive. The same package renders RDLC and RDL reports and Markdown. Its options are Playwright's `page.pdf()` options under their Python names.

Install

```
pip install --pre weblitz
```

Version 2.0.0rc2, a prerelease of 2.0, so pip is asked for prereleases: Python's spelling of 2.0.0-rc.2, the version of every package. Python 3.9 or later, on Linux with glibc (x64, Arm64), macOS (Arm64, x64) and Windows (x64, Arm64): a wheel each, holding the engine and the .NET runtime it runs on, so there is nothing else to install. Alpine and other musl systems are not supported yet; on a slim Linux image, add fontconfig and ICU (`libfontconfig1` and `libicu72`, the ICU package's number following the system's release).

Your first PDFs

```
import weblitz

# HTML text
result = weblitz.render_html("<h1>Invoice 42</h1><p>...</p>", format="A4",
print_background=True)
result.save("invoice.pdf")

# A web page, after its scripts ran; path writes the file too
weblitz.render_url("https://example.com/report", path="report.pdf",
                   margin={"top": "18mm", "right": "18mm", "bottom": "20mm", "left":
"20mm"})

# A file on disk: its CSS, pictures, fonts and scripts are read from its folder
from pathlib import Path
letters = weblitz.render_html(Path("templates/letters.html").read_text(),
                              base_url=Path("templates").resolve().as_uri() + "/")

# What could not be drawn or loaded is said, never raised
for note in result.notes:
    print(note.severity, note.code, note.message)
```

Every call answers a `Result` (`file`, `content_type`, `extension`, `pages`, `notes`, `trial`, and `save(path)`) or raises `WeblitzError`, whose `code` says why. The options are .NET's options under Python's names: `format`, `margin`, `landscape`, `print_background`, `page_ranges`, `header_template`, `base_url`, `output`, `scripts`, `network`, `locale`, `time_zone`, `now`...

Excel, Word, PowerPoint and pictures

```
weblitz.render_url("https://example.com/prices", output="xlsx").save("prices.xlsx")   #
or "docx", "pptx"
weblitz.render_url("https://example.com", output="png",
                   viewport={"width": 1280, "device_scale_factor": 2},
path="example.png")
pages = weblitz.render_html(html, output="png", capture="pages", format="A4")    #
pages.file is a ZIP
```

`output="png"` (or `"jpeg"`, `"webp"`, `"tiff"`) draws the page instead of printing it: pictures of a page.

What a page loads

A page loads nothing from the network by default but, for `render_url`, its own host. `network` opens more, with headers and cookies for pages behind a login, each going only to the hosts it names:

```
weblitz.render_url("https://app.example.com/reports/42", network={
    "allowed_hosts": ["app.example.com", "cdn.example.com"],
    "headers": [{"name": "Authorization", "value": f"Bearer {token}", "hosts":
["app.example.com"]}],
})
```

Reports and Markdown

The same package renders SSRS2's reports: `render_report`, `check_report`, `markdown_to` and `import_report`. A dataset's rows are a list of dicts or a pandas or polars DataFrame. ssrs2 (`pip install --pre ssrs2`) has the same functions for projects that only render reports; it runs on `weblitz`.

```
from pathlib import Path
import weblitz

rows = [{"Description": "Widget", "Price": 104.99, "Qty": 1}]
weblitz.render_report(Path("Invoice.rdlc"), datasets={"Items": rows},
path="invoice.pdf")
weblitz.render_report(Path("Invoice.rdlc"), format="docx", datasets={"Items":
dataframe})

problems = weblitz.check_report(Path("Invoice.rdlc"))   # its problems, without
rendering it
```

Workers, asyncio and timeouts

Calls go to a pool of worker processes kept warm, started on first use; any thread may make them. `weblitz.configure(...)` sets its options; `weblitz.Engine` is a pool of your own and `weblitz.AsyncEngine` the same for asyncio, where cancelling the awaiting task cancels the render. Times are in milliseconds.

```
weblitz.configure(workers=2, timeout=30_000)

async with weblitz.AsyncEngine(workers=4) as engine:
    result = await engine.render_html(html, format="A4", timeout=5_000)
```

On a server

A server that prints pages other people send should start its workers with SSRS2's server settings: a page then loads nothing but a URL's own host, private addresses and `file:` base URLs are refused, report code is always restricted, and a render stops at 2,000 pages or 120 seconds. `SSRS2_WORKER_SETTINGS` changes them, as JSON (the Node.js page shows it).

```
weblitz.configure(env={"SSRS2_WORKER_POLICY": "service"})
```

Licence

The SSRS2 licence covers Weblitz and SSRS2's reports. Give the key as `license=` or as the environment's `SSRS2_LICENSE`. Without one, PDF and image pages carry a trial watermark and a result's `trial` is true. Buy a licence