WeasyPrint

PDF erstellen aus HTML und CSS

Detlef Lannert
PyDdf 18. 06. 2025

Getting started ...

Von der Kommandozeile aus benutzen:

```
$ uv tool install weasyprint
$ weasyprint --help
$ weasyprint muster.html muster.pdf
```

Aufruf in einem Programm

```
import io
from weasyprint import HTML as wHTML, CSS

html = '<!doctype html><html lang="en"> …'  # generated
weasy = wHTML(string=html)
css = CSS("some_stylefile.css")  # or from string
result = io.BytesIO()
weasy.write_pdf(result, stylesheets=[css])

content = result.getvalue()  # write it, send it , …
```

Einbettung in Python- (Web-) Anwendungen

- HTML und CSS können aus Dateien gelesen oder als Strings übergeben werden
- Ausgabe in Datei oder in `BytesIO`
- Templating (Jinja2) bietet sich an
- HTML ggf. aus ReST oder Markdown generieren (z. B. mit rst2html4 oder pandoc)
  → Schreiberleichterung
- **django-weasyprint** bietet angepasste CBVs für Django

Features von WeasyPrint

- unterstützt Bookmarks, Fußnoten, running headers etc.
- kann PDF/A, PDF/UA, Factur-X erstellen
- PDF-Forms, -Metadata, attached files
- Einbindung von Grafiken (JPEG, PNG, GIF, …, SVG)
  (mit Optimierung und Caching)
- Font-Handling über Pango (autom. eingebettet)
- eigene URL-Fetcher können eingebunden werden

Unterstützte Spezifikationen (Auswahl)

Genaue Spezifikationen und Einschränkungen finden sich **in der API-Doku**. Weitgehend implementiert sind beispielsweise:

- CSS 2.1 (besteht Acid-Test, aber ohne RTL/BiDi)
- Selectors Level 3 / 4
- CSS Text Module Level 3 / 4
- CSS Fonts Module Level 3 / 4

Unterstützte Spezifikationen (Forts.)

- CSS Paged Media Module Level 3 und Generated Content
- CSS Color Module Level 3
- einfache Multi-Column-Layouts
- Flexbox- und Grid-Layouts

Ressourcen

Homepage, Beispiele: **weasyprint.org**
Repo: **https://github.com/Kozea/WeasyPrint**
Doku: **https://doc.courtbouillon.org/weasyprint**

Mitbewerber

Andere (freie) Tools, um aus Python-Code heraus PDF zu generieren (garantiert unvollständige Liste):

- **ReportLab**
- **PyFPDF**
- **rinohtype**
- **rst2pdf**
- LaTeX via subprocess
- Browser mit Printfunktion

Vorteile / Annehmlichkeiten von WeasyPrint

- Python-Integration eingabe-/ausgabeseitig
- eigene URL-Auflösung möglich
- Printgestaltung unabhängig von Python-Kenntnissen
- Experimentieren und Debugging im Browser möglich
- gute Standardkonformität

Schwächen

- nicht rasend schnell
- einige (meist exotische …) CSS-Features fehlen noch

Anwendungsbeispiele

- Tabellendarstellung
- TOC-Generierung
- Newsletter, aus Django generiert, in Webversion (mit BS5) und Druckversion
- IT-Sicherheitskonzepte (mit BSI-Checks)
- Geschäftsbrief
- Tickets und Labels

Beispiel Geschäftsbrief

- absolute Positionierung des Briefkopfs, des Adressfelds, des Datums, der Seitenzahl
- normales Layout für `@page :first` und vereinfachtes Layout für die anderen (Folge-) Seiten
- Generierung von Adressangaben o.ä. aus Datenbank möglich
- Rechnung: Tabellen dynamisch einfügen

Fragen?