Python y documentos

Una fructífera relación

Python y documentos

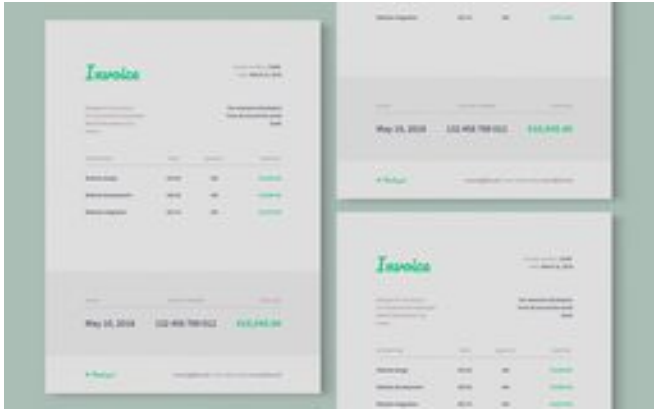
- Generar
  - **HTML**
  - **ReST**ructured Text, Markdown
  - Formularios PDF
  - LibreOfficeWriter y SVG
- Leer
- Modificar

Generar - **WeasyPrint**

Desde HTML

- Flask-WeasyPrint y Django-WeasyPrint
- command line
- weasyprint [options] <input> <output>
- Most CSS 2.1
- Mínimo Python 3.6

Generar - **WeasyPrint**



Generar - **WeasyPrint**

```
from weasyprint import HTML, CSS
html = HTML(string='<h1>The title</h1>')
css = CSS(string='@page { size: A3; margin: 1cm }')
html.write_pdf(
    '/tmp/example.pdf', stylesheets=[css])
```

Generar - **xhtml2pdf**

Desde HTML

- Preparado para Django
- Permite utilizar estilos especiales y CSS
- Mínimo Python 3.6

Generar - **xhtml2pdf**

- Páginas: Define tipo de página, márgenes, etc
- Frames: regiones dentro de las páginas donde fluye el contenido. Estáticos (cabeceras, pies) o dinámicos
- Soporta diferentes estilos en diferentes secciones.

Generar - **xhtml2pdf**

```
from xhtml2pdf import pisa

source_html = "<html><body><p>To PDF or not to
PDF</p></body></html>"

result_file = open("test.pdf", "w+b")

pisa.CreatePDF(source_html, dest=result_file)

result_file.close()
```

Generar - **rinohtype**

Desde ReST o Markdown

- Plantillas de documentos
- Páginas de estilo
- Preparado para Sphinx
- Soporta MarkDown, además de rst

Generar - **rinohtype**

```
from rinoh.frontend.rst import ReStructuredTextReader
from rinoh.templates import Article

with open('my_document.rst') as file:
    document_tree = ReStructuredTextReader().parse(file)

Article(document_tree).render('my_document')
```

Generar - **rst2pdf**

Desde ReST o Markdown

- usa estilos,
- puede incluir
  - HTML usando xhtml2pdf
  - Expresiones matemáticas usando mathplotlib
  - Realzado de código usando pygments

Generar - **rst2pdf**

PDF Documents From Your Text Editor

This tool uses a plain text format for content, then applies styles to make the nice document you see here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam maximus tortor sem, vel pellentesque leo fringilla et. Aliquam imperdiet nisi eget dui finibus sagittis. Nunc malesuada libero vel dignissim pharetra. Cras egestas vehicula quam, et accumsan arcu lacinia auctor. Integer imperdiet sagittis justo, vel varius nulla dapibus finibus. Cras rhoncus mattis pellentesque. Quisque vel sapien sed tellus convallis accumsan. Praesent volutpat sapien at lacinia scelerisque. Phasellus neque libero, consectetur in neque id, egestas elementum nisl.

Mauris eu dolor non massa auctor suscipit. Donec sit amet aliquet eros, id sodales leo. Duis erat ipsum, laoreet eget nulla at, euismod ullamcorper mi. Curabitur vel orci a libero ullamcorper finibus. Sed vel lectus sapien. Praesent mollis et dui at laoreet. Donec eleifend, nunc nec bibendum luctus, massa lorem vestibulum justo, a convallis nunc turpis ut urna. Proin venenatis erat et ante convallis efficitur. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In neque turpis, sollicitudin maximus egestas sed, finibus a odio. Nam eu eros id enim vehicula hendrerit at vel orci.

Code Samples for Documents or Presentations

We use an adapted set of styles based on Pygments. Here are some examples, you can apply the built-styles as you wish.

Python example: Flask minimal example

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

PHP example: Slim Framework minimal example

```php
<?php
use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;
use Slim\Factory\AppFactory;

require __DIR__ . '/../vendor/autoload.php';

$app = AppFactory::create();

$app->get('/hello/{name}', function (Request $request, Response $response, array $args)
    $name = $args['name'];
    $response->getBody()->write("Hello, $name");
    return $response;
});

$app->run();
```

Generar - **fdfgen**

Llenar formularios

- Necesitamos generar un formulario PDF (podría ser con Libre Office)
- Extraer el archivo .fdf con pdftk

  ```
  pdftk <pdf name> dump_data_fields
  ```

- Y allí podremos sencillamente combinar correspondencia

Generar - **fdfgen**

```
from fdfgen import forge_fdf

fields = [('name', 'John Smith'),
          ('telephone', '555-1234')]
fdf = forge_fdf("", fields, [], [], [])

with open("data.fdf", "wb") as fdf_file:
     fdf_file.write(fdf)
```

Generar - **secretary**

Desde aplicaciones

- Usando Libre Office
- Se crea un archivo tipo plantilla
- Luego se automatiza mezclando correspondencia
- Genera archivos de Libre Office, pueden convertise a PDF.

Generar - **certg**

Desde aplicaciones

- Usando Inkscape
- Puede incluirse imágenes
- Se crea un documento plantilla
- Similar a reemplazar correspondencia

Extraer información

- pdfminer.six
- Camelot
- pdf2docx

Extraer - **pdfminer.six**

- Extrae texto, imágenes,
- Etiquetas y soporta desencriptar

```
>>> text = extract_text('samples/simple1.pdf')
>>> print(text)
Hello
World
```

Extraer - **Camelot**

- Extrae tablas, utiliza OpenCV
- Genera dataframes pandas y exporta a muchos formatos
- Reporta la calidad estimada del resultado

Extraer información - **Camelot**

```
import camelot
tables = camelot.read_pdf('foo.pdf')
# json, excel, html, markdown, sqlite
tables.export('foo.csv', f='csv', compress=True)
```

Extraer - **pdf2docx**

Convertir pdf a docx

```
from pdf2docx import parse
parse(pdf_file=input_file,
      docx_with_path=output_file,
      pages=pages)
```

Modificar PDF

- PyMuPDF
- PyPDF3
- pdfrw

Modificar PDF - **PyMuPDF**

- Modifica, crea, reordena y borra páginas
- Junta y divide PDFs
- Encriptar y desencriptar
- Permite buscar y extraer información
- Crea miniaturas para visualizar con wxPython, Tkinter, PyQT 4 y 5
- La ayuda incluye muchas recetas.

Generar Thumbnails - **PyMuPDF**

```
import fitz
pdfIn = fitz.open("input_file.pdf")
for pg in range(pdfIn.pageCount):
    page = pdfIn[pg]
    mat = fitz.Matrix(zoom_x=2, zoom_y=2).preRotate(0)
    pix = page.getPixmap(matrix=mat, alpha=False)
    pix.writePNG(f"output_page{pg+1}.png")
pdfIn.close()
```

Modificar PDF - **PyPDF4**

- Modifica, crea, reordena y borra páginas
- Junta y divide PDFs
- Encriptar y desencriptar
- Agregar marcas de agua y superponer pdf
- Ejemplos más limitados

Agregar marcas de agua

```
watermark_reader = PdfFileReader(watermark_buffer)
pdf_reader = PdfFileReader(open(input_file, 'rb'),
strict=False)
pdf_writer = PdfFileWriter()


for page in range(pdf_reader.getNumPages()):
    page = pdf_reader.getPage(page)
    page.mergePage(watermark_reader.getPage(0))
    pdf_writer.addPage(page)
```

Python y documentos

María Andrea Vignau