

Documentación de UnoGenerator

Version: 0.3.0

1 Introducción

UnoGenerator uses Libreoffice UNO API python bindings to generate documents. So in order to use, you need to launch a --headless libreoffice instance. You can easily launch server.sh script with bash.

UnoGenerator has standard templates to help you with edition, although you can use your own templates. Puedes editar éste o crear el tuyo. This document has been created with 'standard.odt' files that you can find inside this python module.

1.1 Instalación

Puedes usar pip para instalar este paquete python:

```
pip install unogenerator
```

1.2 Ejemplo Hola Mundo

This is a Hello World example. You get the example in odt, docx and pdf formats:

```
from unogenerator import ODT_Standard
doc=ODT_Standard()
doc.addParagraph("Hello World", "Heading 1")
doc.addParagraph("Easy, isn't it", "Standard")
doc.save("hello_world.odt")
doc.export_docx("hello_world.docx")
doc.export_pdf("hello_world.pdf")
doc.close()
```

2 ODT

ODT files can be quickly generated with UnoGenerator. There is a predefined template in code called 'standard.odt' to help you with edition.

2.1 Llamando al constructor ODT

You can call ODT constructor in this ways:

- ODT with standard template (Recomended). There is a predefined template in code called 'standard.odt', inside this python module, to help you with edition, although you can use your own ones. With this mode you can create new documents

```
from unogenerator import ODT_Standard
doc=ODT_Standard()
```

- ODT with template or file (Recomended). With this mode you can read your files to overwrite them or use your file as a new template to create new documents

```
from unogenerator import ODT
doc=ODT('yourdocument.odt')
```

- ODT without template. With this mode you can write your files with Libreoffice default styles. If you want to create new ones, you should write them using Libreoffice API code

```
from unogenerator import ODT
doc=ODT()
```

2.2 Estilos

To call default Libreoffice paragraph styles you must use their english name. You can see their names with this method:

```
doc.print_styles()
```

2.3 Tablas

We can create tables with diferent font sizes and formats:

Concepto	Valor
Texto	Esto es un texto
Fecha y hora	2021-10-01 22:00:09.137794
Fecha	2021-10-01
Float	12.121
Divisa	-12.12 €

Porcentaje	33.33 %
------------	---------

Concepto	Valor
Texto	Esto es un texto
Fecha y hora	2021-10-01 22:00:09.137794
Fecha	2021-10-01
Float	12.121
Divisa	-12.12 €
Porcentaje	33.33 %

2.4 Listas y listas numeradas

Lista simple

- Prueba hola no. Prueba hola no. Prueba hola no. Prueba hola no. Prueba hola no. Prueba hola no. Prueba hola no.
- Adios
- Bienvenido

2.5 Imágenes

Este es un ejemplo de imagen as char:  . Ahora sigo escribiendo sin problemas.

Como puedes ver, puedo reutilizar la imagen cien veces. El tamaño del fichero no aumentará porque uso referencias.                                                                                                                                                                     

El siguiente párrafo es generado con el método illustration



2.6 Buscar y reemplazar

Below this paragraph is a paragraph with a % REPLACEME % (Without white spaces) text and it's going to be replaced after all document is been generated

This paragraph was set at the end of the code after a find and replace command.

3 ODS

```
def                                TTO                                READ
                                demo_ods_standard_read():
                                doc=ODS("unogenerator_ods_standard.ods")
                                doc.setActiveSheet(0)
                                print(doc.getValuesByRow("4",          standard=True))
                                print(doc.getValuesByRow("4",          standard=False))
                                doc.close()
                                return "demo_ods_standard took {}".format(datetime.now()-doc.init)
```