

Generación de formatos PDF, CSV, XML, ...

Antonio Espín Herranz

Contenidos

- Generar contenido en CSV.
- Generar contenido en XML.
- Generar contenido en PDF.
- Librería django-import-export
 - Exportar a CSV
 - Exportar a JSON
 - Exportar a Excel
 - Filtrar datos

Generar contenido en CSV

- Para generar un fichero CSV, lo primero importar el módulo de csv.
 - **import csv**
- En la función de la vista, creamos un objeto HttpResponse indicando content_type con text/csv. Tenemos que indicar el tipo mime.
- Y con la propiedad 'Content-Disposition' el fichero adjunto.
 - **response = HttpResponse(content_type="text/csv")**
 - **response['Content-Disposition'] = 'attachment; filename=nombre_fichero.csv'**
- Creamos un writer pasando el objeto response.
 - **writer = csv.writer(response)**
- A partir de aquí, podemos escribir filas en el csv con el método writerow. Como parámetro recibe una lista:
 - **fila = ['campo1','campo2','...']**
 - **writer.writerow(fila)**
- Una vez hayamos escrito todas las filas, podemos devolver el objeto response.
 - **return response**

Generar contenido XML

- En una **función** de la **vista**:
 - Accedemos al modelo.
 - Y devolvemos **render** con la **request**, el **nombre de la plantilla**, el **contexto** con los datos que queremos pasar y una propiedad:
content_type="xml/text" indicando el tipo **mime**.
- Se puede diseñar una plantilla similar al HTML pero en este caso generando etiquetas de XML.

Generar contenido XML

- En el fichero, se indica la cabecera de XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<registros>
```

```
    {% for r in registros %}
```

```
        <registro>
```

```
            <etiqueta>{{r.propiedad1}}</etiqueta>
```

```
            <etiqueta>{{r.propiedad2}}</etiqueta>
```

```
            <etiqueta>{{r.propiedad3}}</etiqueta>
```

```
        </registro>
```

```
    {% endfor %}
```

```
</registros>
```

Generar contenido en PDF

- **Instalar: reportlab**
- Desde una consola de MS-Dos:
pip install reportlab
- Probar si funciona, entrar en la consola de python y comprobar si no genera errores el comando: **import reportlab**

Generar contenido en PDF

- Importar el **módulo canvas**. Dentro de este módulo disponemos de la **clase Canvas** que será el objeto que necesitamos para generar un PDF.

```
from reportlab.pdfgen import canvas
```

- Similar al XML, cuando creamos el objeto response en la vista, tenemos que indicar el tipo mime, en este caso: **application/pdf**
response = HttpResponse(content_type='application/pdf')
- En la propiedad Content-Disposition se indica el nombre del fichero:
response['Content-Disposition'] = 'attachment; filename=hello.pdf'
- Creamos el objeto Canvas pasando un objeto response.
p = canvas.Canvas(response)

Generar contenido en PDF

- Una vez creado el objeto Canvas ya podemos activar colores, escribir contenido dentro del fichero, etc.
- Una vez que hemos terminado de escribir todo el contenido se cierra el objeto, se graba y retornamos el objeto response.

p.showPage()

p.save()

return response

Ejemplo: Hello.pdf

```
from reportlab.pdfgen import canvas
```

```
def exportarpdf(request):
```

```
    response = HttpResponse(content_type='application/pdf')
```

```
    response['Content-Disposition'] = 'attachment;  
        filename=hello.pdf'
```

```
    p = canvas.Canvas(response)
```

```
    p.drawString(50, 800, "Hola Mundo")
```

```
    p.showPage()
```

```
    p.save()
```

```
    return response
```

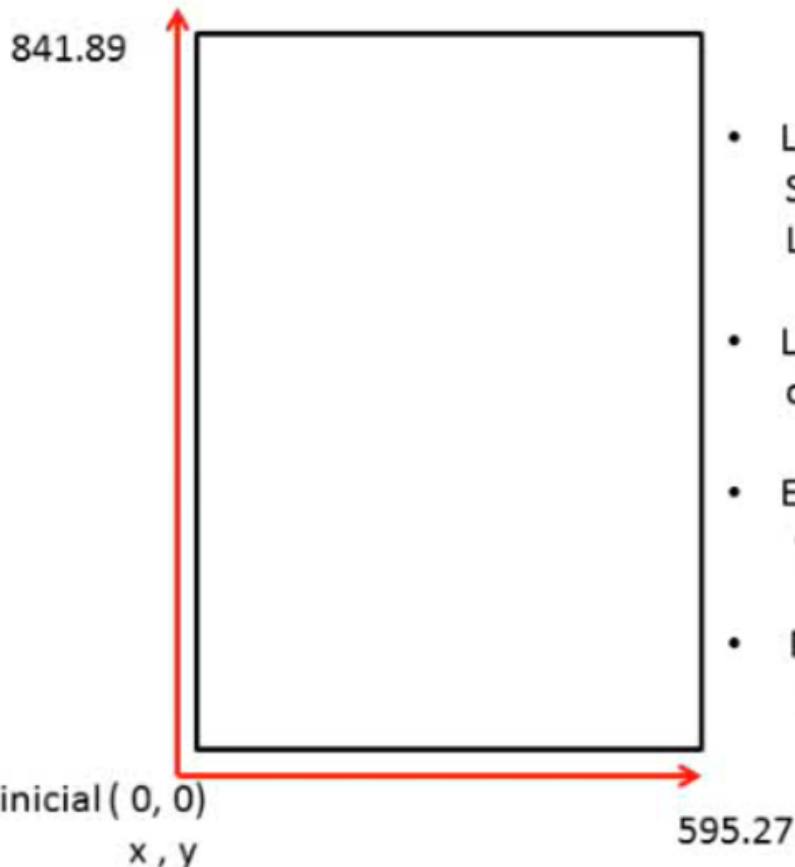
Generar contenido en PDF

- Las coordenadas **(0,0)** del documento se ubican en la **esquina inferior izquierda**.
- El constructor de Canvas se inicializa con:

```
def __init__(self,filename,  
             pagesize=(595.27,841.89), bottomup = 1,  
             pageCompression=0,  
             encoding=rl_config.defaultEncoding, verbosity=0  
             encrypt=None):
```

Generar contenido en PDF

Medidas iniciales por defecto de un documento pdf creado por reportlab.



- Las coordenadas iniciales por defecto Son (0,0) , y la ubicación de este punto en La hoja es la esquina inferior izquierda.
- La dimensión inicial por defecto del documento es : (595.27 X 841.89).
- El valor de la coordenada x se incrementa y se desplaza a la derecha.
- Mientras que el valor de la coordenada y se incrementa y se desplaza hacia arriba.

Tamaños de la página

- Por defecto, el tamaño es A4 con resolución de 72 PPI.
- Dentro del módulo: **pagesizes**
- Podemos importar el tamaño:
`from reportlab.lib.pagesizes import letter, A4, A5, A3`
- Y cuando creamos el Canvas se puede indicar con el parámetro: **pagesize**
- `p = canvas.Canvas(response, pagesize=letter)`
- El tamaño letter:
 - w: 612.0, h: 792.0

Tamaño	72 PPI
4A0	4768 x 6741
2A0	3370 x 4768
A0	2384 x 3370
A1	1684 x 2384
A2	1191 x 1684
A3	842 x 1191
A4	595 x 842
A5	420 x 595
A6	298 x 420
A7	210 x 298
A8	147 x 210
A9	105 x 147
A10	74 x 105

Orientación página

- Por defecto, el documento se crea en vertical.
- Para ponerlo en **horizontal**, importamos el módulo **landscape**
 - from reportlab.lib.pagesizes import letter, A4, landscape
 - Y se indica al crear el objeto Canvas.
p = canvas.Canvas(response, **pagesize=landscape(A4)**)

Métodos de Canvas

- Dibujar **líneas**: `x1,y1, x2,y2`
- Un rectángulo:
 - `p.line(20,20,20,820)` # Vertical izq.
 - `p.line(20,820,575,820)` # Horizontal sup.
 - `p.line(575,20,575,820)` # Vertical der.
 - `p.line(20,20,575,20)` # Horizontal inf.

Métodos de Canvas

- Rectángulos: x, y, width, height

`p.rect(40, 40, 515, 760)`

- Rectángulos rellenos:

`p.setFillColorRGB(1,1,0)`

- Color de relleno, Red, Green, Blue con valores entre 0.0 y 1.0. Negro: 0,0,0 Blanco: 1,1,1

`p.rect(40, 40, 515, 760, fill=1)`

Métodos de Canvas

- Texto: x, y, texto
 - `p.drawString(x, y, “contenido”)`
 - `canvas.drawRightString(x, y, text)`
 - `canvas.drawCentredString(x, y, text)`
- Imágenes:
 - Para cargar un logo en el documento PDF
 - `p.drawImage(“fichero_imagen”, x, y, width, height)`

Párrafos

- Para añadir párrafos de texto al PDF necesitamos importar:
 - from reportlab.platypus import **Paragraph**
- Estilos.
 - from reportlab.lib.styles import **getSampleStyleSheet**

Ejemplo

- Se pueden obtener los estilos de la librería.
`styles = getSampleStyleSheet()`
- `texto = "Contenido del párrafo"`
- Crear un párrafo y le añade el estilo Normal
`p = Paragraph(p text, style=styles["Normal"])`
- La `c` que se pasa por parámetro a los métodos es el Canvas.
`p.wrapOn(c, 50*mm, 50*mm) # size of 'textbox' for linebreaks etc.`
`p.drawOn(c, 20*mm, 0*mm) # position of text / where to draw`
- Podemos trabajar con unidades, mm, inch, cm
`from reportlab.lib.units import inch, mm, cm`
- Si no indicamos nada son pixeles.
- Tener en cuenta que el (0,0) es la esquina inferior izquierda.

Tablas

- Para las tablas necesitamos importar:
from reportlab.platypus import Table, TableStyle
- La tabla se carga con una lista de listas.
 - Puede ser interesante implementar un método `to_list()` en los objetos del modelo.
 - Cuando recuperemos el `querySet` con `MiClase.objects.all()` podemos cargarlos en una lista de listas.

Tablas

- Se puede crear el objeto response y el objeto Canvas (como antes)
- Después crear la tabla con `t = Table(datos)`
 - Siendo datos una lista de listas.
- Y por otro lado están los estilos de la tabla: colores, alineación, grid, innergrid.
 - Para ello necesitamos un objeto de tipo `TableStyle`.
 - Las celdas a las queremos aplicar un estilo lo hacemos mediante tuplas que indican las coordenadas.

TableStyle

```
t = Table(datos)
t.setStyle(TableStyle(
[
    ('ALIGN',(1,1),(-2,-2),'RIGHT'),
    ('TEXTCOLOR',(1,1),(-2,-2),colors.red),
    ('VALIGN',(0,0),(0,-1),'TOP'),
    ('TEXTCOLOR',(0,0),(0,-1),colors.blue),
    ('ALIGN',(0,-1),(-1,-1),'CENTER'),
    ('VALIGN',(0,-1),(-1,-1),'MIDDLE'),
    ('TEXTCOLOR',(0,-1),(-1,-1),colors.green),
    ('INNERGRID', (0,0), (-1,-1), 0.25, colors.black),
    ('BOX', (0,0), (-1,-1), 0.25, colors.black),
])
))
```

00	01	02	03	04
10	11	12	13	14
20	21	22	23	24
30	31	32	33	34

Notas

- Al final tenemos que grabar y mostrar la página como antes.
- Falta decirle al objeto table que se pinte en el objeto c (Canvas).
- Este primer método calcula las dimensiones de la tabla.
table.wrapOn(c, width, height)
- Indica donde se empieza a pintar, el problema es la coordenada y, que empieza por abajo.
table.drawOn(c, 2*cm, 6*cm)

django-import-export

- Instalar:
 - `pip install django-import-export`
- En el fichero de settings:
 - `INSTALLED_APPS = (... 'import_export',)`
 - Activar la variable de configuración, por defecto es `False`:
 - `IMPORT_EXPORT_USE_TRANSACTIONS = True`

django-import-export

- Esta librería trabaja con el concepto de resources.
 - Muy similar a como trata django los modelos.

```
from django.db import models
```

```
class Person(models.Model):  
    name = models.CharField(max_length=30)  
    email = models.EmailField(blank=True)  
    birth_date = models.DateField()  
    location = models.CharField(max_length=100, blank=True)
```

```
from import_export import resources
```

```
from .models import Person
```

```
class PersonResource(resources.ModelResource):  
    class Meta:  
        model = Person
```


Exportar a CSV

```
from django.http import HttpResponse  
from .resources import PersonResource
```

```
def export(request):  
    person_resource = PersonResource()  
    dataset = person_resource.export()  
    response = HttpResponse(dataset.csv,  
        content_type='text/csv')  
    response['Content-Disposition'] = 'attachment;  
        filename="persons.csv"'  
return response
```

Exportar a JSON

```
from django.http import HttpResponse  
from .resources import PersonResource
```

```
def export(request):
```

```
    person_resource = PersonResource()
```

```
    dataset = person_resource.export()
```

```
    response = HttpResponse(dataset.json,  
                             content_type='application/json')  
    response['Content-  
Disposition'] = 'attachment; filename="persons.json"'
```

```
return response
```

Exportar a Excel

```
from django.http import HttpResponse  
from .resources import PersonResource
```

```
def export(request):  
    person_resource = PersonResource()  
    dataset = person_resource.export()  
    response = HttpResponse(dataset.xls,  
        content_type='application/vnd.ms-excel')  
    response['Content-Disposition'] = 'attachment;  
        filename="persons.xls"  
    return response
```

Con filtrado

- `person_resource = PersonResource()`
- `queryset =`
`Person.objects.filter(location='Helsinki')`
- `dataset = person_resource.export(queryset)`

Enlaces

- Tutorial de **reportlab**
 - <https://www.blog.pythonlibrary.org/2010/03/08/a-simple-step-by-step-reportlab-tutorial/>
- **django-import-export:**
 - Documentación:
 - https://django-import-export.readthedocs.io/en/latest/getting_started.html#creating-import-export-resource
 - Tutorial:
 - <https://simpleisbetterthancomplex.com/packages/2016/08/11/django-import-export.html>