Generar un PDF de un post en Django

5 de Diciembre de 2016 a las 17:56

Estoy migrando mi blog de *Wordpress* a *Django* y deseo que tengo las mismas funcionalidades en *Django*. Una de ellas, es la posibilidad de generar un *PDF* de un post del blog, para realizar esto tenia instalado en *Wordpress* un plugin, esta funcionalidad no quería perderla en la migración.

La implementación debería ser la misma que tenia en *Wordpress*, al final de cada post habrá un enlace que al pulsarlo genera el *PDF*, mostrara los datos mas importante del post; título, fecha y contenido, también se añadirá una cabecera y un pie de página.

En Django hay diversos paquetes que permiten trabajar con PDF, tenemos:

- *Reportlab*: posiblemente el mas famoso, permite crear *PDF* desde cero, ya sean diseños simples como complejos.
- *WeasyPrint*: permite generar un *PDF* de un html.
- *Wkhtmltopdf*: igual que anterior genera *PDF* desde un html.

Después de varias búsquedas y pruebas para generar un *PDF*, de los tres anteriores, el que mejor me funciono fue *Wkhtmltopdf*, fue el único que me permitió generar un *PDF* correctamente, cumpliendo los requisitos que tenia. Con los otros dos tuve diversos problema que no conseguí solucionarlos, eso no significa que no sean buenas opciones para trabajar con *PDF*, simplemente en mi caso no conseguí que funcionaran como yo quería.

INSTALACIÓN

Primero deberemos instalar *wkhtmltopdf,* existen paquetes para las principales distribuciones de Linux, utilizando el gestor de paquetes es fácil instalarlo.

NOTA

Para utilizar determinadas opciones de configuración, wkhtmltopdf debe estar parcheado, el paquete disponible en los repositorios no lo esta. Mas adelante se vera como conseguir una versión parcheada.

Segundo es utilizar *wkhtmltopdf* en *django,* la manera mas fácil es utilizar una paquete llamado *django-wkhtmltopdf* , utilizando comando *pip*.

```
pip install django-wkhtmltopdf
```

Como requerimientos tenemos la librería *libfontconfig*, que debe estar instalado en el sistema, soporta *python* 2.6 como 3.3 o superior.

Una vez instalada,agregar la aplicación *'wkhtmltopdf'* en el apartado *INSTALLED_APPS* dentro del fichero *settings.py* del proyecto de Django.

Django-*wkhtmltopdf* puede ser personalizado con diferentes opciones, en el fichero settings.py podemos añadir una variable donde especificaremos diferentes opciones para generar *PDF*, esta variable *WKHTMLTOPDF_CMD_OPTIONS* que es un diccionario donde almacenara las diversas opciones de configuración para los *PDF* generados.

```
WKHTMLTOPDF_CMD_OPTIONS = {
        'opción' : valor
}
```

También se puede incluir las opciones en el método *PDFTemplateResponse,* que mas adelante se vera.

GENERACIÓN DE PDF

Ahora veremos un ejemplo práctico para integrar *django-wkhtmltopdf* en un proyecto. Para generar un *PDF* necesitaremos:

- Una URL.
- Una vista, especificada en la url que sera la encargada de hacer el "trabajo sucio" y generar el *PDF*.
- Una plantilla que será utilizada como modelo para el *PDF*.

Primero configuraremos la url.

```
urlpatterns = [
.....
 url(r'^pdf/(?P\S+)$', views.MyPDFView.as_view(), name='pagina_detalle'),
 ]
```

Tenemos una vista,que es del tipo *clase basada en vista (CBV),* con nombre *MyPDFView* y la ruta para especificar el post que se convertirá a *PDF*.

Para la vista,es un poco mas extenso.

```
class MyPDFView(DetailView):
 model = Entry
 template = 'pdf_export.html'
 context = {'titulo': 'Rooteando Blog'}

 def get(self, request, *args, **kwargs):
    self.context['entry'] = self.get_object()
    response = PDFTemplateResponse(request=request,
                                   template=self.template,
                                   filename="postPDF.pdf",
                                   context=self.context,
                                   footer_template="footer.html",
                                   show_content_in_browser=True,
                                   cmd_options={'margin-top': 15,
                                                'margin-bottom': 20,
                                                'default-header': True,
                                                'header-left': 'Rooteando Blog',
                                                'footer-line': True,
                                                },
```

```
                    )
    return response
```

MyPDFView es del tipo *DetailView* que es un vista para mostrar en detalle los campos de un objeto. Primero especificamos el modelo de donde obtendremos los campos,*Entry,* la plantilla a utilizar para mostrar los datos(*pdf_export.html)*, mas adelante se verá esta plantilla, y una variable de contexto, que es un diccionario. Esta variable almacenara el texto "*Rooteando Blog*", cualquier dato que queramos usar en la plantilla sera mediante su variable de contexto.

A continuación tendremos el método *get* que permitirá obtener el objeto que es enviado por medio de una petición, *request, .* incluido como parámetro del método.En este caso, se refiere alun post utilizado para convertirlo, mediante un enlace que habrá situado al final de cada post.

En el cuerpo del método, se añade una variable, *entry,* al contexto que almacenara el objeto que se a utilizar, que será obtenido por medio de la función *get:_object()*. Por último, se incluye una llamada función *PDFTemplateResponse*, incluida en *django-wkhtmltopdf*, que permite generar un PDF de un plantilla dada, como parámetros tenemos:

- *request*: corresponde a una petición.
- *template*: la plantilla que será utilizada para generar el PDF, se utilizara la especificada anteriormente.
- *filename:* especifica el nombre del fichero PDF generado.
- *context:* la variable de contexto, declarada anteriormente.
- *footer_template*: permite crear un pie de página(footer) , que ha sido definido en una plantilla.
- *show_content_in_browser*: el PDF será mostrado en el mismo navegador utilizando el visor que incluye el navegador.
- *cmd_option:* opciones de configuración del documento.
    - *margim-top*: define el margen superior del documento.
    - *margim-bottom*: define el margen inferior del documento.
    - *default-header:* genera un encabezado, con el número de página y una línea de separación.
    - *header-left*: incluye en el encabezado un texto especificado alineado a la izquierda.
    - *footer-line:* genera una línea de separación en el pie de página.

La función *PDFTemplateResponse* devuelve el PDF generado a partir de una plantilla especificada, pdf_export.html,que es almacenado en la variable *response* que es devuelta por *MyPDFView*.

Como se ha explicado anteriormente, para la generación se utiliza una plantilla donde se especifican que datos serán utilizados. Anteriormente se ha obtenido un objeto de la clase *Entry* con todos los campos del post.

A continuación se muestra la plantilla *pdf_export.html.*

```
<!DOCTYPE html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
```

```
<body>
   </br>

<h1>{{ entry.title }}</h1>

   <i>{{entry.modified}}</i>
      {{entry.body|safe}}
</body>
[/sourcecode]
```

Dentro del *head* tenemos una línea que nos permite reconocer las tildes y la 'ñ', configurando el *charset* a *utf-8.* Dentro de *body,* especificamos los campo del post que queremos incluir, *title*,*modified* y *body*. En este último, se ha añadido el filtro *safe* para que muestre el texto sin las etiquetas html. Para escribir un post se utiliza *CKEditor*, es importante utilizar esta etiqueta para que no muestre el html que genera el editor.

Por último se añade a cada post una enlace para que genere el PDF cuando el usuario pulse. Este enlace se inserta en la plantilla que se utiliza para generar un post.

En mi caso he utilizando el siguiente código.

```
<a href="{% url "pagina_detalle"  slug=object.slug %}"> Generar PDF de {{object.title}}</a>
```

Este enlace envía a la URL especificada con el *slug* del post que esta viendo en ese momento.

Con esto se genera un PDF bastante simple de un post, se pueden añadir mas opciones que permitirán darle un aspecto mas profesional. La librería *WkhtmltoPDF* permite generar PDF mas complejos con tablas o imágenes, en su documentación explica como hacerlo.

WKHTMLTOPDF PARCHEADO

Antes se ha mencionado que determinadas funcionalidades, como los encabezados y pie de página, para la generación del PDF requiere un *wkhtmltopdf* parcheado.

En mi caso, para una Fedora, no encontré un paquete parcheado y utilice el binario disponible en la página de *wkhtmltopdf.* aquí. Descargo la versión correspondiente y busco el binario, que se encontrara en una carpeta denominada *bin,* copio esa versión en la carpeta donde se encuentra la versión instalada en el sistema, en Fedora se encuentra en la carpeta '*/usr/bin/*'. Con esto sobrescribimos la versión instalada en el sistema con el binario que se ha descargado que esta parcheado.

Con estos pasos, tenemos la versión parcheada en el sistema y podemos utilizar todas las funcionalidades disponibles de *wkhtmltopdf.* Es muy recomendable que la versión instalada en el sistema y el binario descargado sean de la misma versión.