django-pdf-overlay

Release 1.0.5

Nov 26, 2021

Contents

Source code http://github.com/iarp/django-pdf-overlay

Documentation https://django-pdf-overlay.readthedocs.io/en/latest/

CHAPTER 1

Overview

Designed to make it easy for developers working with PDF's to create views, pass model data, and have an easy to use GUI for field CRUD and layout.

1. Supply a PDF document in the django-pdf-overlay admin screen.
2. Create fields that match what you need filled out on the document.
3. Using the layout tool, move the fields to their respective locations on the document.
4. In your view, add similar to the following:

```
# In this example I will load a user and pass it
# to the PDF which has user.username as a field.
from django.contrib.auth import get_user_model
u = get_user_model().objects.get(pk=1)

from django_pdf_overlay.models import Document
doc = Document.objects.get(name='My Document')

# Here we render the page(s) on the PDF
doc.render_pages(user=u)

# You can call render_pages multiple times to generate a single
# PDF containing multiple copies of the base document.
u2 = get_user_model().objects.get(pk=2)
doc.render_pages(user=u2)

# If you wish to generate an actual file that you can store
# in a model or somewhere on your system.
file = doc.render_as_document(filename='users_1_2.pdf')

# Or if you want the document to auto-download to the user
return doc.render_as_response(filename='users_1_2.pdf')
```

CHAPTER 2

Contents

2.1 Installation

2.1.1 Requirements

- Python 2.7, 3.3, 3.4, 3.5, 3.6, 3.7
- Django (1.11+)
- PyPDF2
- reportlab
- django-bootstrap4
- ImageMagick
  - Only required if you want auto-generated layout images.
  - If you do not want to or cannot install ImageMagick see GENERATE_LAYOUT_IMAGE
  - **Theres an issue with Imagemagick disallowing PDF interactions, you need to edit the policy.xml file (linux** location: /etc/ImageMagick-6/policy.xml) and change the line *<policy domain="coder" rights="none" pattern="PDF" />* changing "none" to "readlwrite".

2.1.2 Django

Python package:

```
pip install django-pdf-overlay
```

settings.py:

```
INSTALLED_APPS = (
    ...
    'django_pdf_overlay',
    'bootstrap4',
    ...
)
```

urls.py:

```
urlpatterns = [
    ...
    path('django-pdf-overlay/', include('django_pdf_overlay.urls', namespace='django-
↪pdf-overlay')),
    ...
]
```

2.1.3 Development Settings

django-pdf-overlay requires your project to have media settings configured for the layout images to appear properly.

During development (*DEBUG=True*) that means you may not see the layout image at all.

The settings provided below are meant to be used in development ONLY, do not use these on production!

settings.py:

```
if DEBUG:
    MEDIA_URL = '/media/'
    MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

urls.py:

```
from django.conf import settings

if settings.DEBUG:
    from django.urls import re_path
    from django.views.static import serve
    urlpatterns = urlpatterns + [
        re_path(r'^media/(?P<path>.*)$', serve, {'document_root': settings.MEDIA_ROOT,
↪}),
    ]
```

2.1.4 Post-Installation

Run migrate to create the necessary tables:

```
./manage.py migrate
```

Start your server and then head to the Django PDF Overlay admin area (e.g. http://127.0.0.1:8000/django-pdf-overlay/)

Django PDF Overlay Admin uses built-in django permissions, superusers will always have full access, everyone else will require permissions.

Create a new group called *PDF Editors* and add the wanted Permissions for any user added to this group.

2.2 Configuration

2.2.1 Available Settings

DJANGO_PDF_OVERLAY_GENERATE_LAYOUT_IMAGE (=True) Enable or disable whether or not the system will auto-generate layout images from PDF.

Useful when you cannot or don't want to install ImageMagic.

Disabling this does require you to manually create the image and attach it to the page accordingly before field layout is possible.

DJANGO_PDF_OVERLAY_MAGICK_LOCATION The location of ImageMagick's convert utility. Value must be a list, if you need to add options each option must be its own string entry see Windows Default for example.

- Linux Default: *['/usr/bin/convert']*
- Windows Default *['magick.exe', 'convert']*

See: https://imagemagick.org/script/download.php for installation.

DJANGO_PDF_OVERLAY_MAGICK_DENSITY (='300') When creating the images for field layout purposes, what level of pixel quality do you want?

DJANGO_PDF_OVERLAY_MAGICK_FLATTEN (=True) Pass -flatten to magick conversion? True or False.

DJANGO_PDF_OVERLAY_LOCAL_DOCUMENT_STORAGE (=os.path.join(BASE_DIR, 'media', 'django_pdf_overlay', 'd The document uploaded MUST be located locally to the server itself. Where do we place these files?

Regardless of default storage and media settings in django, this location takes precedence as the file must be on the server.

DJANGO_PDF_OVERLAY_FIELD_VALUE_JOINS (=', .-_') When you chain object attributes in "obj name" on a field, you can select what value to join the chained values on. Default is a space, supply a string, list, tuple, or set to customize this.

- Comma
- Space
- Period
- Dash
- Underscore

DJANGO_PDF_OVERLAY_FIELD_CHAIN_SPLITTER (='|') Value used to split chained object.attributes on fields obj name value. Default is Pipe.

DJANGO_PDF_OVERLAY_FIELD_DATETIME_SPLITTER (=':') Value used to split object.attribute from its datetime formatting.

This value MUST be different than DJANGO_PDF_OVERLAY_FIELD_CHAIN_SPLITTER.

DJANGO_PDF_OVERLAY_COMMANDS (='django_pdf_overlay.commands.DefaultCommands') Dot notated path to the Commands class allowing you to alter certain methods used within the program.

2.3 Fields Configuration

2.3.1 Available Options

name Visual name for the layout page.

default (=”) If no value was found on the model instance, do you have a default you want to supply?

To print the current datetime as a field on the document, leave obj name blank and supply dt:<datetime format> (e.g. dt:%Y-%m-%d), the field will print YYYY-mm-dd.

obj name (=”) Where to find the data from the parameters passed to *render_pages* in your code. Format: object.attribute

Leave blank if using datetime in the default field as noted above.

When you call *render_pages* you will be passing model instances:

```
doc.render_pages(user=user)
```

If you wish to access a field on the user instance, supply *user.<attribute>* to obj name (e.g. user.username or user.email).

It is possible to chain object.attributes (see Field Chaining below).

If the value of the object.attribute is a date or datetime object, you must supply a format to be used as such:

```
user.date_joined:%Y-%m-%d
```

You can chain that too:

```
user.date_joined:%Y-%m-%d|user.first_name|user.last_name
```

font size (=12) Size of font used on the render.

font (=’Helvetica’) The font used must be installed on the server doing the rendering.

font color (=’black’) English name, hashed hex code, OR comma separated RGB values (e.g. 230,230,250)

2.3.2 render_pages parameters explained

As noted above, obj name field should match the format *object.attribute* where object is supplied as a named parameter to *render_pages*. (e.g. *render_pages(user=user)*).

It will also match against dict keys using the same dot notation (user.username). So you can supply your own dict of values instead of a model instance.

Dot notation only works at the first level user.username and no farther.

If the attribute is callable, it will be called with no parameters supplied.

Example 1:

```
# Document has a field with obj name of my_dict.val1

my_dict = {
    'val1': 'Here in dict',
}

doc.render_pages(my_dict=my_dict)
```

Example 2:

```
# Document has a field with user.username

user = User.objects.get(pk=1)

doc.render_pages(user=user)
```

2.3.3 Field Chaining

You can chain object.attributes together by supplying multiples separated by a pipe(|):

```
>>> user.first_name|user.last_name
John Doe
```

The resulting values from both attributes will be joined by a space.

You can change the character used for joining the chained values by supplying it after the chained object attributes:

```
>>> user.first_name|user.last_name|-
John-Doe
```

See DJANGO_PDF_OVERLAY_FIELD_VALUE_JOINS on configuration for all possible join values permitted by default.

2.4 Commands

You can override the following class and its methods to change certain behavior

Create your custom class extending DefaultCommands and supply the dot notation path to your class with `DJANGO_PDF_OVERLAY_COMMANDS` in your project settings:

- `django_pdf_overlay.commands.DefaultCommands`
  - `get_pdf_to_image_command(self, document, page)` returns a list of prepared commands to be used in the `execute` method below.
  - `execute(self, document, page, commands)` Executes the commands given on the system. Commands is the output from get_pdf_to_image_command above. No return needed. document and page are the objects being worked on.
  - `get_layout_image_filename(self, document, page)` Returns a string containing the filename of the layout image about to be saved to page.layout. Default is `{document filename}_{page number}.jpg`
  - `convert_to_image(self, document, page)` Combines all methods above and does the actual processing of data.

2.5 Example

my_proj/overrides.py:

```
from django_pdf_overlay.commands import DefaultCommands

class OverriddenCommands(DefaultCommands):
    def execute(self, document, page, commands):
        # You can now change the system call used from our default of
        # subprocess.Popen to whatever you want.
```

my_proj/settings.py:

```
DJANGO_PDF_OVERLAY_COMMANDS = 'my_proj.overrides.OverriddenCommands'
```

CHAPTER 3

Indices and tables

- genindex
- search