dj-easy-pdf Documentation

Release 0.1.2.dev1

William Otieno

Jul 04, 2022

CONTENTS

Django PDF rendering, the easy way but with support for newer versions of Django.

Developed by William Otieno.

CHAPTER

ONE

OVERVIEW

This app makes rendering PDF files in Django really easy. It can be used to create invoices, bills and other documents from simple HTML markup and CSS styles. You can even embed images and use custom fonts.

The library provides both Class-Based View that is almost a drop-in replacement for Django's `TemplateView` as well as helper functions to render PDFs in the backend outside the request scope (i.e. using Celery workers).

CHAPTER

TWO

QUICKSTART

1. Include `dj-easy-pdf`, `xhtml2pdf` in your `requirements.txt` file. If you are on Python 3 you need to install the latest version of Reportlab and the beta version of xhtml2pdf:

```
$ pip install xhtml2pdf
```

2. Add `easy_pdf` to `INSTALLED_APPS`.

3. Create HTML template for PDF document and add a view that will render it:

```
{% extends "easy_pdf/base.html" %}

{% block content %}
    <div id="content">
        <h1>Hi there!</h1>
    </div>
{% endblock %}
```

```
from easy_pdf.views import PDFTemplateView

class HelloPDFView(PDFTemplateView):
    template_name = 'hello.html'
```

4. You can also use a mixin to output PDF from Django generic views:

```
class PDFUserDetailView(PDFTemplateResponseMixin, DetailView):
    model = get_user_model()
    template_name = 'user_detail.html'
```

5. If you're using Function Based views then:

```
from easy_pdf.rendering import render_to_pdf_response

def pdf_output(request):
    context = {}
    return render_to_pdf_response(request, "output.html", context)
```

CHAPTER

THREE

DOCUMENTATION

The full documentation is at dj-easy-pdf.readthedocs.io.

CHAPTER

FOUR

DEPENDENCIES

`django-easy-pdf` depends on:

- `django>=3.2`
- `xhtml2pdf`
- `reportlab`

CHAPTER

FIVE

LICENSE

`dj-easy-pdf` is released under the MIT license.

CHAPTER

SIX

OTHER RESOURCES

- GitHub repository - https://github.com/WilliamOtieno/dj-easy-pdf
- PyPi Package site - https://pypi.python.org/pypi/dj-easy-pdf
- Docs - https://dj-easy-pdf.readthedocs.io/

CHAPTER

SEVEN

COMMERCIAL SUPPORT

This app and many other help us build better software and focus on delivering quality projects faster. We would love to help you with your next project so get in touch by dropping an email at jimmywilliamotieno@gmail.com

CHAPTER

EIGHT

CONTENT

8.1 Installation

Add `dj-easy-pdf=<version>` and `git+https://github.com/chrisglass/xhtml2pdf.git` to your `requirements.txt` file or install it directly from the command line by invoking:

```
$ pip install dj-easy-pdf xhtml2pdf reportlab
```

8.2 Usage

8.2.1 Prepare HTML Templates

Create a Django HTML template with embedded CSS style. You can use special style attributes to format the PDF output.

For more information on the supported HTML and CSS rules see docs at https://github.com/xhtml2pdf/xhtml2pdf/blob/master/doc/source/usage.rst

You can also use custom embeddable resources like images and fonts. Put them inside Django's `STATIC_ROOT` directory and make sure they are available locally on the server even if you are serving your static files from S3 or other CDN.

For now only local resources are supported.

```
{% extends "easy_pdf/base.html" %}

{% block extra_style %}
    <style type="text/css">
        body {
            font-family: "Helvetica", "sans-serif";
            color: #333333;
        }
    </style>
{% endblock %}

{% block content %}
    <div id="content">
        <div class="main">
            <h1>Hi there!</h1>
        </div>
```

(continues on next page)

(continued from previous page)

```
    </div>
{% endblock %}
```

8.2.2 Create PDF rendering views

This part is easy. The PDF rendering view inherits from `TemplateResponseMixin` so it works in the same way as Django's `TemplateView`. Just point it to a HTML template and define `get_context_data()` method to pass any extra variables to the template:

```
from easy_pdf.views import PDFTemplateView

class HelloPDFView(PDFTemplateView):
    template_name = 'hello.html'

    def get_context_data(self, **kwargs):
        return super(HelloPDFView, self).get_context_data(
            pagesize='A4',
            title='Hi there!',
            **kwargs
        )
```

Then add the view to your url config and start serving PDF files rendered from the HTML template.

```
urlpatterns = [
    url(r'^hello.pdf$', HelloPDFView.as_view())
]
```

You can also use a mixin to output PDF from Django generic views:

```
.. code-block:: python
```

class PDFUserDetailView(PDFTemplateResponseMixin, DetailView):
model = get_user_model() template_name = 'user_detail.html'

8.2.3 Rendering PDF outside of Django views

See *PDF rendering functions*.

8.3 API Overview

8.3.1 Views

PDFTemplateResponseMixin

class easy_pdf.views.**PDFTemplateResponseMixin**

Bases: `TemplateResponseMixin`

A mixin class that implements PDF rendering and Django response construction.

pdf_filename = None

Optional name of the PDF file for download. Leave blank for display in browser.

pdf_kwargs = None

Additional params passed to `render_to_pdf_response()`

get_pdf_filename()

Returns *pdf_filename* value by default.

If left blank the browser will display the PDF inline. Otherwise it will pop up the "Save as.." dialog.

Return type
`str()`

get_pdf_kwargs()

Returns *pdf_kwargs* by default.

The kwargs are passed to `render_to_pdf_response()` and `xhtml2pdf.pisa.pisaDocument()`.

Return type
`dict`

get_pdf_response(*context*, ***response_kwargs*)

Renders PDF document and prepares response.

Returns
Django HTTP response

Return type
`django.http.HttpResponse`

render_to_response(*context*, ***response_kwargs*)

Return a response, using the *response_class* for this view, with a template rendered with the given context.

Pass response_kwargs to the constructor of the response class.

PDFTemplateView

class easy_pdf.views.**PDFTemplateView**(***kwargs*)

Bases: *PDFTemplateResponseMixin*, `ContextMixin`, `View`

Concrete view for serving PDF files.

```
class HelloPDFView(PDFTemplateView):
    template_name = "hello.html"
```

get(*request*, **args*, ***kwargs*)

Handles GET request and returns HTTP response.

8.3.2 PDF rendering functions

easy_pdf.rendering.**render_to_pdf**(*template*, *context*, *using=None*, *request=None*, *encoding='utf-8'*, ***kwargs*)

Create PDF document from Django html template.

Parameters

- **template** (*str*) – Path to Django template
- **context** (*dict*) – Template context
- **using** – Optional Django template engine
- **request** (django.http.HttpRequest) – Django HTTP request

Returns
rendered PDF

Return type
bytes

Raises
PDFRenderingError, UnsupportedMediaPathException

easy_pdf.rendering.**render_to_pdf_response**(*request*, *template*, *context*, *using=None*, *filename=None*, *encoding='utf-8'*, ***kwargs*)

Renders a PDF response using given `request`, `template` and `context`.

If `filename` param is specified then the response `Content-Disposition` header will be set to `attachment` making the browser display a "Save as.." dialog.

Parameters

- **request** (django.http.HttpRequest) – Django HTTP request
- **template** (*str*) – Path to Django template
- **context** (*dict*) – Template context
- **using** – Optional Django template engine

Return type
django.http.HttpResponse

Other lower-level helpers

easy_pdf.rendering.**html_to_pdf**(*content*, *dest*, *encoding='utf-8'*, *link_callback=fetch_resources*, ***kwargs*)

Converts html `content` into PDF document.

Parameters
content (*unicode*) – html content

Returns
PDF content

Return type
bytes

Raises
PDFRenderingError

easy_pdf.rendering.**fetch_resources**(*uri*, *rel*)

Retrieves embeddable resource from given `uri`.

For now only local resources (images, fonts) are supported.

Parameters
uri (*str*) – path or url to image or font resource

Returns
path to local resource file.

Return type
str

Raises
`UnsupportedMediaPathException`

easy_pdf.rendering.**make_response**(*content*, *filename=None*, *content_type='application/pdf'*)

Wraps content into HTTP response.

If `filename` is specified then `Content-Disposition: attachment` header is added to the response.

Default `Content-Type` is `application/pdf`.

Parameters

- **content** (*bytes*) – response content
- **filename** (*str*) – optional filename for file download
- **content_type** (*str*) – response content type

Return type
`django.http.HttpResponse`

easy_pdf.rendering.**encode_filename**(*filename*)

Encodes filename part for `Content-Disposition: attachment`.

```
>>> print(encode_filename("abc.pdf"))
filename=abc.pdf
>>> print(encode_filename("aa bb.pdf"))
filename*=UTF-8''aa%20bb.pdf
>>> print(encode_filename(u"zażółć.pdf"))
filename*=UTF-8''za%C5%BC%C3%B3%C5%82%C4%87.pdf
```

8.3.3 Exceptions

8.4 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

8.4.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/WilliamOtieno/dj-easy-pdf/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

dj-easy-pdf could always use more documentation, whether as part of the official dj-easy-pdf docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/WilliamOtieno/dj-easy-pdf/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

8.4.2 Get Started!

Ready to contribute? Here's how to set up *dj-easy-pdf* for local development.

1. Fork the *dj-easy-pdf* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-easy-pdf.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv django-easy-pdf
$ cd django-easy-pdf/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8:

```
$ flake8 easy_pdf
```

To get flake8, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

8.4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
2. The pull request should work for Python 3.4+ plus (if there are compatible 3rd party packages available).

8.5 Credits

8.5.1 Development Lead

- William Otieno <jimmywilliamotieno@gmail.com> <https://github.com/WilliamOtieno>

8.6 History

8.6.1 0.2.1 (2022-06-15)

- Beta Release. Support for Django 3.2+ and 4.0

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX