Django Privates Documentation

Release 1.3.0

Sergei Maertens

Dec 17, 2021

Contents:

Django-privates makes it easy to work with login-protected FileFields, all the way through your application.

CHAPTER 1

Features

- Default private media storage, configurable via settings
- Model field using the default storage
- Easy admin integration
- File serving through sendfile (supports nginx, apache, runserver,...)

1.1 Quickstart

1.1.1 Installation

Install from PyPI with pip:

```
pip install django-privates
```

And add the app to your `INSTALLED_APPS` for admin integration:

```
INSTALLED_APPS = [
    ...

    'privates',

    ...
]
```

Settings

You need to set the following settings:

- `PRIVATE_MEDIA_ROOT`: equivalent of `MEDIA_ROOT` - the base location where private media files will be saved.

- PRIVATE_MEDIA_URL: equivalent of MEDIA_URL. Note that your webserver must be configured appropriately for this, see the sendfile documentation.
- SENDFILE_BACKEND: depends on your webserver, see the sendfile documentation.
- SENDFILE_ROOT: should be set to PRIVATE_MEDIA_ROOT.
- SENDFILE_URL: should be set to PRIVATE_MEDIA_URL.

1.1.2 Usage

Models

To use private media files, the simplest way is to use the model field:

```
from django.db import models

from privates.fields import PrivateMediaFileField


class Invoice(models.Model):
    ...

    pdf = PrivateMediaFileField(blank=True)

    ...
```

This uses the underlying privates.storages.private_media_storage.

Additionally, there is support for ImageField through privates.fields.PrivateMediaImageField.

Admin

There is built in admin integration via a mixin. This takes care of replacing the default widget so you can open the private media files, and perform permission checks.

```
from django.contrib import admin

from privates.admin import PrivateMediaMixin

from .models import Invoice


@admin.register(Invoice)
class InvoiceAdmin(PrivateMediaMixin, admin.ModelAdmin):
    private_media_fields = ('pdf',)
```

By default, this mixin requires you to have the <applabel>.can_change_<model> permission.

Attributes:

- private_media_fields: list or tuple of field names that should be treated as private media fields.
- private_media_permission_required: (custom) permission to check instead of the default <applabel>.can_change_<model>
- private_media_view_options: optional arguments to pass down to the sendfile.sendfile function.

Views

Django Privates ships with a built in permission-check view, used by the admin integration. It's suitable to be used standalone.

It's built on top of `django.contrib.auth.mixins.PermissionRequiredMixin` and `django.view.generic.DetailView`, so the methods/attributes of these base classes are available.

```
from privates.views import PrivateMediaView


class InvoicePDFView(PrivateMediaView):
    queryset = Invoice.objects.all()
    file_field = 'pdf'
    permission_required = 'applabel.can_change_invoice'
```

Tests

To isolate tests, you should clean up any uploaded files generated during tests. There is a wrapper around `django.test.override_settings` available to facilitate this:

```
from privates.test import temp_private_root


@temp_private_root()
class MyTests(TestCase):
    pass
```

The usage is the same as `override_settings`, so you can use it as a class decorator, test method decorator or context manager.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search