django-sendfile2

Release 0.7.0

Matt Molyneaux

Aug 08, 2022

CONTENTS:

This is a wrapper around web-server specific methods for sending files to web clients. This is useful when Django needs to check permissions associated files, but does not want to serve the actual bytes of the file itself. i.e. as serving large files is not what Django is made for.

Note this should not be used for regular file serving (e.g. CSS etc), only for cases where you need Django to do some work before serving the actual file.

- Download: https://pypi.org/project/django-sendfile2/
- Source: https://github.com/moggers87/django-sendfile2
- Documentation: https://django-sendfile2.readthedocs.io/

CHAPTER

ONE

SUPPORTED PYTHON VERSIONS

Python 3.7, 3.8, 3.9, and 3.10 are currently supported by this library.

CHAPTER

TWO

SUPPORTED DJANGO VERSIONS

Django 2.2, 3.2, and 4.0 are currently supported by this library.

CHAPTER

THREE

FORK

This project is a fork of django-sendfile. The original project appears mostly dead and has a number of outstanding bugs (especially with Python 3).

CHAPTER

FOUR

FUNDING

If you have found django-sendfile2 to be useful and would like to see its continued development, please consider buying me a coffee.

4.1 Getting Started

4.1.1 Installation

Install via pip:

```
pip install django-sendfile2
```

And then add `django_sendfile` to `INSTALLED_APPS` in your settings module.

Note: It is not strictly nessessary to have django_sendfile in `INSTALLED_APPS`, but this may change in future.

You will need to have the following set in your settings module:

- `SENDFILE_BACKEND` - the dotted module notation of the backend you wish to use
- `SENDFILE_ROOT` - the directory you wish to serve files from

Additionally, you may need to set `SENDFILE_URL` . See the *Backends* documentation for more details.

4.1.2 Use In Views

Use the `sendfile()` function instead of the usual `HttpResponse` function:

```
from django_sendfile import sendfile

@login_required
def my_secret_view(request):
    return sendfile(request, "/opt/my_secret.txt", mimetype="text/plain")
```

Alternatively, if you prefer class based views something like this would be required:

```
from django_sendfile import sendfile

class MySecretView(LoginRequiredMixin, View):
```

(continues on next page)

(continued from previous page)

```
    def render_to_response(self, context):
        return sendfile
```

4.2 Backends

Backends are specified by setting `SENDFILE_BACKEND` to the dotted path of the backend you wish to use. E.g.:

Listing 1: settings.py

```
SENDFILE_BACKEND = "django_sendfile.backends.simple"
```

4.2.1 Development backend

django_sendfile.backends.development

The Development backend is only meant for use while writing code. It uses Django's static file serving code to do the job, which is only meant for development. It reads the whole file into memory and the sends it down the wire - not good for big files, but OK when you are just testing things out.

It will work with the Django dev server and anywhere else you can run Django.

4.2.2 Simple backend

django_sendfile.backends.simple

This backend is one step up from the development backend. It uses Django's `django.core.files.base.File` class to try and stream files from disk. However some middleware (e.g. GzipMiddleware) that rewrites content will causes the entire file to be loaded into memory. So only use this backend if you are not using middleware that rewrites content or you only have very small files.

4.2.3 mod_wsgi backend

django_sendfile.backends.mod_wsgi

The mod_wsgi backend will only work when using mod_wsgi in daemon mode, not in embedded mode. It requires a bit more work to get it to do the same job as xsendfile though. However some may find it easier to setup, as they don't need to compile and install mod_xsendfile.

Firstly there one more Django setting that needs to be given:

- `SENDFILE_URL` - internal URL prefix for all files served via sendfile

These settings are needed as this backend makes mod_wsgi send an internal redirect, so we have to convert a file path into a URL. This means that the files are visible via Apache by default too. So we need to get Apache to hide those files from anything that's not an internal redirect. To so this we can use some mod_rewrite magic along these lines:

```
RewriteEngine On
# see if we're on an internal redirect or not
RewriteCond %{THE_REQUEST} ^[\S]+\ /private/
RewriteRule ^/private/ - [F]
```

(continues on next page)

(continued from previous page)

```
Alias /private/ /home/john/Development/myapp/private/
<Directory /home/john/Development/myapp/private/>
    Order deny,allow
    Allow from all
</Directory>
```

In this case I have also set:

Listing 2: settings.py

```
SENDFILE_ROOT = '/home/john/Development/myapp/private/'
SENDFILE_URL = '/private'
```

All files are stored in a folder called 'private'. We forbid access to this folder (`RewriteRule ^/private/ - [F]`) if someone tries to access it directly (`RewriteCond %{THE_REQUEST} ^[\S]+\ /private/`) by checking the original request (`THE_REQUEST`).

Allegedly `IS_SUBREQ` can be used to perform the same job, but I was unable to get this working.

4.2.4 Nginx backend

`django_sendfile.backends.nginx`

As with the mod_wsgi backend you need to set an extra settings:

- `SENDFILE_URL` - internal URL prefix for all files served via sendfile

You then need to configure Nginx to only allow internal access to the files you wish to serve. More details on this are here.

For example though, if I use the Django settings:

Listing 3: settings.py

```
SENDFILE_ROOT = '/home/john/Development/django-sendfile/examples/protected_downloads/
↪protected'
SENDFILE_URL = '/protected'
```

Then the matching location block in nginx.conf would be:

```
location /protected/ {
  internal;
  root   /home/john/Development/django-sendfile/examples/protected_downloads;
}
```

You need to pay attention to whether you have trailing slashes or not on the `SENDFILE_URL` and `SENDFILE_ROOT` values, otherwise you may not get the right URL being sent to Nginx and you may get 404s. You should be able to see what file Nginx is trying to load in the error.log if this happens. From there it should be fairly easy to work out what the right settings are.

4.2.5 xsendfile backend

django_sendfile.backends.xsendfile

Install either mod_xsendfile in Apache or use Lighthttpd. You may need to configure mod_xsendfile, but that should be as simple as:

```
XSendFile On
```

In your virtualhost file/conf file.

4.3 Custom Backend

A django-sendfile2 backend is simply a Python module that contains a callable named `sendfile`, for example:

Listing 4: myModule.py

```
def sendfile(request, filename, **kwargs):
    response = HttpResponse()
    response["X-My-Custom-Header"] = filename
    return response
```

Assuming the module is in your Python path and named `myModule`, you'd set `SENDILE_BACKEND` like so:

Listing 5: settings.py

```
SENDFILE_BACKEND = "myModule"
```

. . . and use django-sendfile2 in your views as you would normally.

Warning: Don't get confused between this `sendfile` callable and *sendfile()* used in your views. The latter accepts slightly different arguments and takes care of various `Content-*` headers.

4.4 django_sendfile

4.4.1 django_sendfile package

Subpackages

django_sendfile.backends package

Submodules

django_sendfile.backends.development module

`django_sendfile.backends.development.`**`sendfile`**(*request*, *filename*, ***kwargs*)

Send file using Django dev static file server.

> **Warning:** Do not use in production. This is only to be used when developing and is provided for convenience only

django_sendfile.backends.mod_wsgi module

`django_sendfile.backends.mod_wsgi.`**`sendfile`**(*request*, *filename*, ***kwargs*)

django_sendfile.backends.nginx module

`django_sendfile.backends.nginx.`**`sendfile`**(*request*, *filename*, ***kwargs*)

django_sendfile.backends.simple module

`django_sendfile.backends.simple.`**`sendfile`**(*request*, *filepath*, ***kwargs*)

Use the SENDFILE_ROOT value composed with the path arrived as argument to build an absolute path with which resolve and return the file contents.

If the path points to a file out of the root directory (should cover both situations with '..' and symlinks) then a 404 is raised.

`django_sendfile.backends.simple.`**`was_modified_since`**(*header=None*, *mtime=0*, *size=0*)

Was something modified since the user last downloaded it?

header
This is the value of the If-Modified-Since header. If this is None, I'll just return True.

mtime
This is the modification time of the item we're talking about.

size
This is the size of the item we're talking about.

django_sendfile.backends.xsendfile module

`django_sendfile.backends.xsendfile.`**`sendfile`**(*request*, *filename*, ***kwargs*)

Module contents

Submodules

django_sendfile.utils module

`django_sendfile.utils.`**`sendfile`**(*request*, *filename*, *attachment=False*, *attachment_filename=None*, *mimetype=None*, *encoding=None*)

Create a response to send file using backend configured in `SENDFILE_BACKEND`

`filename` is the absolute path to the file to send.

If `attachment` is `True` the `Content-Disposition` header will be set accordingly. This will typically prompt the user to download the file, rather than view it. But even if `False`, the user may still be prompted, depending on the browser capabilities and configuration.

The `Content-Disposition` filename depends on the value of `attachment_filename`:

`None` (default): Same as `filename` `False`: No `Content-Disposition` filename `String`: Value used as filename

If neither `mimetype` or `encoding` are specified, then they will be guessed via the filename (using the standard Python mimetypes module)

Module contents

4.5 Security

If you find a security issue with django-sendfile2, email security@moggers87.co.uk. If you want to send an encrypted report, then please use key id 0x878B5A2A1D47C084.

django-sendfile2 follows the same security reporting model that has worked for other open source projects: If you report a security vulnerability, it will be acted on immediately and a fix with complete full disclosure will go out to everyone at the same time.

4.6 Changelog

4.6.1 0.7.0

release-date
2022-08-08

- Fix reflected file download vulnerability
- Add support for spaces in filenames

4.6.2 0.6.1

release-date
2021-01-18

- Fixed Django 4.0 compatibility
- Add support for Python 3.10
- Remove support for Python 3.5 and 3.6
- Remove support for Django 3.1

4.6.3 0.6.0

release-date
2020-06-17

- Fixed issue where django-sendfile could serve *any* file, even if it was outside `SENDFILE_ROOT`. `SENDFILE_ROOT` is now required for all backends.

4.6.4 0.5.1

release-date
2019-12-30

- **Fix issue with versioneer not being updated about the package name change**
  - tox now does a proper sdist and install to avoid this in future

4.6.5 0.5.0

release-date
2019-12-30

- Rename Python package from `sendfile` to `django_sendfile`
  - This will require changing `SENDFILE_BACKEND`, `INSTALLED_APPS`, and any imports
- Remove code used to support Python 2.7
- Add support for the latest versions of Django and Python

4.6.6 Earlier Releases

Sorry, we didn't keep a changelog prior to version 0.5.0!

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX