django-downloadview Documentation

Release 1.0

Benoît Bryon

October 28, 2013

Contents

Django-DownloadView provides (class-based) generic download views for Django.

Example, in some urls.py:

```
from django.conf.urls import url, url_patterns
from django_downloadview import ObjectDownloadView
from demoproject.download.models import Document  # A model with a FileField.


# ObjectDownloadView inherits from django.views.generic.BaseDetailView.
download = ObjectDownloadView.as_view(model=Document, file_field='file')

url_patterns = ('',
    url('^download/(?P<slug>[A-Za-z0-9_-]+)/$', download, name='download'),
)
```

CHAPTER 1

Ressources

- online documentation: http://django-downloadview.readthedocs.org
- PyPI page: http://pypi.python.org/pypi/django-downloadview
- code repository: https://github.com/benoitbryon/django-downloadview
- bugtracker: https://github.com/benoitbryon/django-downloadview/issues

CHAPTER 2

Contents

2.1 Demo project

The `demo/` folder holds a demo project to illustrate (and test) django-downloadview usage.

2.1.1 Browse demo code online

See demo folder in project's repository [1].

2.1.2 Deploy the demo

System requirements:

- Python [2] version 2.6 or 2.7, available as `python` command.

  Note: You may use Virtualenv [3] to make sure the active `python` is the right one.

- `make` and `wget` to use the provided `Makefile`.

Execute:

```
git clone git@github.com:benoitbryon/django-downloadview.git
cd django-downloadview/
make demo
```

It installs and runs the demo server on localhost, port 8000. So have a look at http://localhost:8000/

Note: If you cannot execute the Makefile, read it and adapt the few commands it contains to your needs.

Browse and use `demo/demoproject/` as a sandbox.

[1] https://github.com/benoitbryon/django-downloadview/tree/master/demo/demoproject/
[2] http://python.org
[3] http://virtualenv.org

2.1.3 References

2.2 Installation

This project is open-source, published under BSD license. See *License* for details.

If you want to install a development environment, you should go to *Contributing to the project* documentation.

Install the package with your favorite Python installer. As an example, with pip:

```
pip install django-downloadview
```

There is no need to register this application in your Django's `INSTALLED_APPS` setting.

Next, you'll have to setup some download view(s). See *demo project* for examples, and *API documentation*.

Optionally, you may setup additional *server optimizations*.

2.3 Optimizations

Some reverse proxies allow applications to delegate actual download to the proxy:

- with Django, manage permissions, generate files...
- let the reverse proxy serve the file.

As a result, you get increased performance: reverse proxies are more efficient than Django at serving static files.

2.3.1 Nginx

If you serve Django behind Nginx, then you can delegate the file download service to Nginx and get increased performance:

- lower resources used by Python/Django workers ;
- faster download.

See Nginx X-accel documentation [4] for details.

Configure some download view

As an example, let's consider an application called "myapp".

`settings.py`:

```
INSTALLED_APPS = (
    # ...
    'myapp',
    # ...
)
MYAPP_STORAGE_LOCATION = '/var/www/files/'  # Could be MEDIA_ROOT for public
                                            # files.
```

This application holds a `Document` model.

`myapp/models.py`:

[4] http://wiki.nginx.org/X-accel

```
from django.conf import settings
from django.core.files.storage import FileSystemStorage
from django.db import models


storage = FileSystemStorage(location=settings.MYAPP_STORAGE_LOCATION)


class Document(models.Model):
    file = models.FileField(storage=storage, upload_to='document')
```

Notice the `storage` and `upload_to` parameters: files for `Document` model live in `/var/www/files/document/` folder.

Then we configured a download view for this model, restricted to authenticated users:

`myapp/urls.py`:

```
from django.conf.urls import url, url_patterns
from django.contrib.auth.decorators import login_required

from django_downloadview import ObjectDownloadView

from myapp.models import Document


download = login_required(ObjectDownloadView.as_view(model=Document))

url_patterns = ('',
    url('^document/(?P<pk>[0-9]+/download/$', download, name='download'),
)
```

As is, Django is to serve the files, i.e. load chunks into memory and stream them.

Nginx is much more efficient for the actual streaming... Let's use it!

Configure Nginx

See Nginx X-accel documentation [1] for details.

Here is what you could have in `/etc/nginx/sites-available/default`:

```
charset utf-8;

# Django-powered service.
upstream frontend {
    server 127.0.0.1:8000 fail_timeout=0;
}

server {
    listen 80 default;

    # File-download proxy.
    #
    # Will serve /var/www/files/myfile.tar.gz when passed URI
    # like /optimized-download/myfile.tar.gz
    #
    # See http://wiki.nginx.org/X-accel
    # and https://github.com/benoitbryon/django-downloadview
```

```
    location /optimized-download {
        internal;
        # Location to files on disk.
        # See Django's settings.NGINX_DOWNLOAD_MIDDLEWARE_MEDIA_ROOT
        alias /var/www/files/;
    }

    # Proxy to Django-powered frontend.
    location / {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;
        proxy_pass http://frontend;
    }
}
```

... where specific configuration is the `location /optimized-download` section.

Note: `/optimized-download` is not available for the client, i.e. users won't be able to download files via `/optimized-download/<filename>`.

> **Warning:** Make sure Nginx can read the files to download! Check permissions.

Global delegation, with XAccelRedirectMiddleware

If you want to delegate all file downloads to Nginx, then use `django_downloadview.nginx.XAccelRedirectMiddleware`.

Register it in your settings:

```
MIDDLEWARE_CLASSES = (
    # ...
    'django_downloadview.nginx.XAccelRedirectMiddleware',
    # ...
)
```

Setup the middleware:

```
NGINX_DOWNLOAD_MIDDLEWARE_MEDIA_ROOT = MYAPP_STORAGE_LOCATION
NGINX_DOWNLOAD_MIDDLEWARE_MEDIA_URL = '/optimized-download'
```

Optionally fine-tune the middleware. Default values are `None`, which means "use Nginx's defaults".

```
NGINX_DOWNLOAD_MIDDLEWARE_EXPIRES = False  # Force no expiration.
NGINX_DOWNLOAD_MIDDLEWARE_WITH_BUFFERING = False  # Force buffering off.
NGINX_DOWNLOAD_MIDDLEWARE_LIMIT_RATE = False  # Force limit rate off.
```

Local delegation, with x_accel_redirect decorator

If you want to delegate file downloads to Nginx on a per-view basis, then use `django_downloadview.nginx.x_accel_redirect()` decorator.

Adapt `myapp/urls.py`:

```
from django.conf.urls import url, url_patterns
from django.contrib.auth.decorators import login_required

from django_downloadview import ObjectDownloadView
+ from django_downloadview.nginx import x_accel_redirect

from myapp.models import Document


download = login_required(ObjectDownloadView.as_view(model=Document))
+ download = x_accel_redirect(download,
+                             media_root=settings.MY_APP_STORAGE_LOCATION,
+                             media_url='/optimized-download')

url_patterns = ('',
    url('^document/(?P<pk>[0-9]+/download/$', download, name='download'),
)
```

Common issues

`Unknown charset "utf-8" to override`

Add `charset utf-8;` in your nginx configuration file.

`open() "path/to/something" failed (2:  No such file or directory)`

Check your `settings.NGINX_DOWNLOAD_MIDDLEWARE_MEDIA_ROOT` in Django configuration VS `alias` in nginx configuration: in a standard configuration, they should be equal.

References

Currently, only nginx's X-Accel [5] is supported, but contributions are welcome [6]!

2.3.2 How does it work?

The feature is inspired by Django's TemplateResponse [7]: the download views return some `django_downloadview.response.DownloadResponse` instance. Such a response doesn't contain file data.

By default, at the end of Django's request/response handling, Django is to iterate over the `content` attribute of the response. In a `DownloadResponse`, this `content` attribute is a file wrapper.

It means that decorators and middlewares are given an opportunity to capture the `DownloadResponse` before the content of the file is loaded into memory As an example, `django_downloadview.nginx.XAccelRedirectMiddleware` replaces `DownloadResponse` instance by some `django_downloadview.nginx.XAccelRedirectResponse`.

[5] http://wiki.nginx.org/X-accel
[6] https://github.com/benoitbryon/django-downloadview/issues?labels=optimizations
[7] https://docs.djangoproject.com/en/1.4/ref/template-response/

2.3.3 References

2.4 API

Here is API documentation, generated from code.

2.4.1 django_downloadview

django_downloadview Package

`django_downloadview` Package

`decorators` Module

`middlewares` Module

`nginx` Module

`response` Module

`utils` Module

`views` Module

2.5 About django-downloadview

2.5.1 Alternatives and related projects

This document presents other projects that provide similar or complementary functionalities. It focuses on differences with django-downloadview.

Django's static file view

Django has a builtin static file view [8]. It can stream files. As explained in Django documentation, it is designed for development purposes. For production, static files'd better be served by some optimized server.

Django-downloadview can replace Django's builtin static file view:

- perform actions with Django when receiving download requests: check permissions, generate files, gzip, logging, signals...
- delegate actual download to a reverse proxy for increased performance.
- disable optimization middlewares or decorators in development, if you want to serve files with Django.

[8] https://docs.djangoproject.com/en/1.4/ref/contrib/staticfiles/#static-file-development-view

References

2.5.2 License

2.5.3 Authors & contributors

Original code by Novapost team:

- Nicolas Tobo <https://github.com/nicolastobo>
- Lauréline Guérin <https://github.com/zebuline>
- Gregory Tappero <https://github.com/coulix>
- Benoît Bryon <benoit@marmelune.net>

2.5.4 Changelog

1.0 (2012-12-04)

- Introduced optimizations for Nginx X-Accel: a middleware and a decorator
- Introduced generic views: DownloadView and ObjectDownloadView
- Initialized project

2.6 Contributing to the project

This document provides guidelines for people who want to contribute to the project.

2.6.1 Create tickets

Please use the bugtracker [9] **before** starting some work:

- check if the bug or feature request has already been filed. It may have been answered too!
- else create a new ticket.
- if you plan to contribute, tell us, so that we are given an opportunity to give feedback as soon as possible.
- Then, in your commit messages, reference the ticket with some `refs #TICKET-ID` syntax.

2.6.2 Fork and branch

- Work in forks and branches.
- Prefix your branch with the ticket ID corresponding to the issue. As an example, if you are working on ticket #23 which is about contribute documentation, name your branch like `23-contribute-doc`.
- If you work in a development branch and want to refresh it with changes from master, please rebase [10] or merge-based rebase [11], i.e. don't merge master.

2.6.3 Setup a development environment

System requirements:

- Python [12] version 2.6 or 2.7, available as `python` command.

  Note: You may use Virtualenv [13] to make sure the active `python` is the right one.

- make and wget to use the provided `Makefile`.

Execute:

```
git clone git@github.com:benoitbryon/django-downloadview.git
cd django-downloadview/
make develop
```

If you cannot execute the Makefile, read it and adapt the few commands it contains to your needs.

2.6.4 The Makefile

A `Makefile` is provided to ease development. Use it to:

- setup the development environment: `make develop`
- update it, as an example, after a pull: `make update`
- run tests: `make test`
- build documentation: `make documentation`

The `Makefile` is intended to be a live reference for the development environment.

[9] https://github.com/benoitbryon/django-downloadview/issues
[10] http://git-scm.com/book/en/Git-Branching-Rebasing
[11] http://tech.novapost.fr/psycho-rebasing-en.html
[12] http://python.org
[13] http://virtualenv.org

2.6.5 Documentation

Follow style guide for Sphinx-based documentations [14] when editing the documentation.

2.6.6 Test and build

Use the Makefile.

2.6.7 References

[14] http://documentation-style-guide-sphinx.readthedocs.org/

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*