Basic File Manager for Django Documentation

Release 0.6

Simonas Kazlauskas

April 17, 2015

Contents

BFM is an Apache Licenced server file manager made with ease of use and new web technologies in mind.

Because of that BFM looks like Django Admin page. Therefore BFM shouldn't cause any unfamiliarity for anyone who used Django before.

It has minimal dependencies and most of those are optional as they enable optional features.

CHAPTER 1

Core Features

- Select multiple files at once,
- Realtime upload status reporting,
- File browsing,
- File actions (Delete, Rename, Touch),
- Directory browsing,
- Admin applet (You can make Uploader appear in any admin page).

Note: Core features requires no optional dependencies, however *Core dependencies* are required.

CHAPTER 2

Next steps

2.1 Introduction

2.1.1 Why another? There's adminfiles and filebrowser already.

Why software gets created? Because someone were unhappy with already existing software.

That was case for me too - I found it hard to make both those to work the way I want, so I made my own filebrowser.

2.1.2 Why use BFM?

1. You hate Flash, love HTML5, CSS3, JavaScript and have modern browser.
2. You want simple and fast interface for managing your files online.
3. You couldn't even make another two work. BFM is simple to setup.

2.1.3 Why you would not want to use BFM yet?

1. If you want integrate it with TinyMCE or something of the like, you shouldn't think about BFM. **YET**.
2. You want mature project.
3. You need to support IE, which is not even a browser.

2.2 Support tables, dependencies

2.2.1 Browser

Version, required to have full functionality of BFM. BFM itelf will still work with older browser, but some things, like file uploading won't work.

Browser	Version
Chrome/Chromium	7 +
Firefox	5 +
Opera	Not supported *
IE	10 + *
Safari	5.0 +

* Opera and IE 9 and below doesn't support XHR2, which is used to upload files and report upload status.

2.2.2 Core dependencies

BFM was only tested on Python 2.6 and Python 2.7, so it is safe to assume, that dependencies are as follows:

- Python: 2.6 or 2.7
- Django: 1.3 or 1.4

2.3 Installation

2.3.1 pip and easy_install

Easiest way to install BFM is to use `pip`:

```
$ pip install django_bfm
```

And second to easiest is with `easy_install`:

```
$ easy_install django_bfm
```

Note: Using easy_install is discouraged. Why? Read here.

2.3.2 GIT repository

Alternatively, you can clone and install from Github repository, where project is developed.

```
$ git clone git://github.com/simukis/django-bfm.git
$ cd django-bfm
$ python2 setup.py install
```

or with pip:

```
pip install -e git+git@github.com:simukis/django-bfm.git#egg=Package
```

2.4 Configuration

2.4.1 Adding to Django

After downloading and installing BFM, you need to configure your Django project to work with it.

1. Add `'django_bfm',` to your `INSTALLED_APPS` in **settings.py**,
2. Add `url(r'^files/', include('django_bfm.urls')),` to your `urlpatterns` in **urls.py**,
3. Make sure you have staticfiles enabled (with context processor) and run *python manage.py collectstatic*,
4. Make sure, that static files are served correctly by your production server.

Note: You don't need to run collectstatic if you are working in develovepment server. It serves files automatically.

2.4.2 Next steps

Settings variables

Variables, that users should define in their `settings.py` file, to customize BFM installation.

Directories and URLs

If these values are not set, then BFM takes values from `MEDIA_ROOT` and `MEDIA_URL`.

BFM_MEDIA_DIRECTORY

The absoulute path to the directory, where files accessible by BFM resides. Please note, that BFM will not be able to access directories, that are parents to specified one.

Example: `/home/example.com/media/uploads/`

Default: `MEDIA_ROOT`

BFM_MEDIA_URL

URL to use when referring to media located in `BFM_MEDIA_DIRECTORY`.

Example: `/media/uploads/` or `http://media.example.com/uploads/`

It **must** end in slash.

Default: `MEDIA_URL` if empty and `BFM_MEDIA_DIRECTORY` is not equal to `MEDIA_ROOT` (either by setting by hand or defaulting to that value).

BFM appearance

You can control ammount of files displayed in one page and things like that.

BFM_FILES_IN_PAGE

BFM uses Pagination to ensure, that reasonable ammount of files is displayed at once. You may specify, how much that "reasonable" is for you by giving this variable an integer value.

Example: `50`

Default: `20`

LOGIN_URL

To make sure, that only authorized users fiddle with, BFM requires user to be logged in. If user is not logged in, it will be redirected to login.

Note: This setting variable is same as Django LOGIN_URL.

Uploader behaviour

You can control several aspects of uploader like files uploaded at once.

BFM_ADMIN_UPLOAD_DIR

Directory path relative to `BFM_MEDIA_DIRECTORY`. Tells BFM where to save files uploaded with *Admin Applet*.

Example: `admin_files/` and files will be uploaded to `{{BFM_MEDIA_DIRECTORY}}/admin_files/`.

Default: empty string.

BFM_UPLOADS_AT_ONCE

Tells BFM how much files to upload at once. Requires integer value.

Example: `3`

Default: `2`

Note: If you experiencing issues like dropped connections with uploader then set this option to `1`.

2.5 Settings variables

Variables, that users should define in their `settings.py` file, to customize BFM installation.

2.5.1 Directories and URLs

If these values are not set, then BFM takes values from `MEDIA_ROOT` and `MEDIA_URL`.

BFM_MEDIA_DIRECTORY

The absoulute path to the directory, where files accessible by BFM resides. Please note, that BFM will not be able to access directories, that are parents to specified one.

Example: `/home/example.com/media/uploads/`

Default: `MEDIA_ROOT`

BFM_MEDIA_URL

URL to use when referring to media located in `BFM_MEDIA_DIRECTORY`.

Example: `/media/uploads/` or `http://media.example.com/uploads/`

It **must** end in slash.

Default: `MEDIA_URL` if empty and `BFM_MEDIA_DIRECTORY` is not equal to `MEDIA_ROOT` (either by setting by hand or defaulting to that value).

2.5.2 BFM appearance

You can control ammount of files displayed in one page and things like that.

BFM_FILES_IN_PAGE

BFM uses Pagination to ensure, that reasonable ammount of files is displayed at once. You may specify, how much that "reasonable" is for you by giving this variable an integer value.

Example: `50`

Default: `20`

LOGIN_URL

To make sure, that only authorized users fiddle with, BFM requires user to be logged in. If user is not logged in, it will be redirected to login.

Note: This setting variable is same as Django LOGIN_URL.

2.5.3 Uploader behaviour

You can control several aspects of uploader like files uploaded at once.

BFM_ADMIN_UPLOAD_DIR

Directory path relative to `BFM_MEDIA_DIRECTORY`. Tells BFM where to save files uploaded with *Admin Applet*.

Example: `admin_files/` and files will be uploaded to `{{BFM_MEDIA_DIRECTORY}}/admin_files/`.

Default: empty string.

BFM_UPLOADS_AT_ONCE

Tells BFM how much files to upload at once. Requires integer value.

Example: `3`

Default: `2`

Note: If you experiencing issues like dropped connections with uploader then set this option to `1`.

2.6 Admin Applet

> **Warning:** BFM javascript file downloads all files it requires including newest version of jQuery, so all `$` and `jQuery` variables defined before will be rewritten. `django.jQuery` variable won't change.

Making admin applet to work requires more work because application specific files should be edited.

Firstly, open your application, in which you want applet to appear, `admin.py` file and these lines just after imports

```
from django.core.urlresolvers import reverse
from django.utils.functional import lazy
reverse_lazy = lazy(reverse, str)
```

Note: If you are using Django 1.4, you can just import it: `from django.core.urlresolvers import reverse_lazy`.

Then in your `SomethingAdmin` class add another class named Media, if it doesn't exist yet.

Finally, add `reverse_lazy('bfm_opt')` into `js` tuple of `Media` class.

In the end `Media` class should look something like this:

```
class EntryAdmin(admin.ModelAdmin):
    class Media:
        # ... Your other css and js ...
        js += (reverse_lazy('bfm_opt'),)
    #Your admin continues here...
```

That's all. You're ready to see your applet in http://example.com/admin/application/model.

You may want to change some options, that changes behavior of Admin applet. You can see them in *Uploader behaviour*.

2.7 Additional features

These are features, that does require some additional dependencies, like `PIL`.

2.7.1 Image resizing

Image resizing is file action (button appears near delete, rename, etc buttons) which helps you to thumbnailize your uploaded images.

Dialog box supports aspect ratio locking, selecting resizing mode (like `bicubic`).

Dependency: `PIL`

CHAPTER 3

Indices and tables

- *genindex*
- *search*