django-bft - A Big File Transfer App Written in Django Documentation

Release 0.1

Jay McEntire

Mar 23, 2017

Contents

CHAPTER 1

Introduction

django-bft is a Django application that handles large file transfers. The application's frontend has both a flash upload as well as a javascript upload version.

django-bft was origianlly created for Utah State University to fulfill a specific need of sending large files through email.

The site can be viewed at https://bft.usu.edu

CHAPTER 2

Requirements

1. **Web server**

   Apache using **mod-wsgi** is the prefered web server of choice and, currently, is the only web server that has been tested with django-bft. Web servers such as Lighttp and Nginx may work, but have not been tested.

   Note: django-bft uses ajax to display a progress bar during upload. As such, the Django development web server will not be able to upload files since it is only single threaded.

2. **Database**

   All databases supported by Django should work with django-bft. (Postgres, MySQL, and Oracle)

3. **Django**

   django-bft has been created using the latest version of Django (1.3). Django 1.1x or later will probably work, but with the change in how **static** files are handled, you are advised to install django-staticfiles for static file serving.

   Refer to: https://github.com/jezdez/django-staticfiles

CHAPTER 3

Installation

To get this application up and running, please follow the steps below:

1. Create a Django production environment using the setup of your choice. Refer to: http://docs.djangoproject.com/en/dev/intro/install/
2. Create a new Django Project:

```
$ django-admin.py startproject <projectname>
```

3. Install django-bft to either your PYTHON_PATH or in a folder inside your project:
   - Install from pip:

```
$ pip install django-bft
```

   - Download and install from source:

```
$ python setup.py install
```

   - Install source to local directory:

```
$ python setup.py build
$ cp build/lib/bft /<PROJECT_ROOT>/
```

4. Add the following to your **settings.py** file:
   - Add 'bft', 'staticfiles' and 'admin' to INSTALLED_APPS:

```
INSTALLED_APPS = (
        ...
        'django.contrib.admin',
        'django.contrib.staticfiles',
        'bft',
)
```

   - Make sure your static file finders are configured:

```
STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
)
```

- Add 'bft.context_processors.bft' to the context processors:

```
from django.conf import global_settings
...

TEMPLATE_CONTEXT_PROCESSORS = (
    'bft.context_processors.bft',
) + global_settings.TEMPLATE_CONTEXT_PROCESSORS
```

- Add 'bft.utils.upload_handlers.UploadProgressCachedHandler' to the file upload handlers:

```
from django.conf import global_settings
...

FILE_UPLOAD_HANDLERS = (
    'bft.utils.upload_handlers.UploadProgressCachedHandler',
) + global_settings.FILE_UPLOAD_HANDLERS
```

- (Optional) - You might want to change the location of the FILE_UPLOAD_TEMP_DIR. This setting is the location of where files will be temporarily be stored while uploading. The default for this is the operating systems temp directory. If space is limited at this directory, it is advised to change this setting.

5. Modify **app_settings.py** as needed.
6. If reCaptcha is to be used, a public and private key will be needed.

   Refer to: http://www.google.com/recaptcha

   The python reCaptcha library is also **required**. You can install it onto your PYTHON_PATH by using pip:

```
$ pip install recaptcha-client
```

7. Because this app is designed to upload very large files, enforcing size restrictions using Django is not recomended and rarly works. It is better to add this enforcement on the web server layer. You can do this on Apache by adding the following directives to your Apache config:

```
LimitRequestBody 1073741824 #The same size as specified in your app_settings.py␣
↪file
ErrorDocument 413 http://<SERVER_NAME>/413error/
```

8. If you care about your files being secured, you will probably want to protect the directory where files are stored using apache:

```
<Location /media/files>
        Order allow,deny
        Deny from all
</Location>
```

   You may also want to consider adding the necessary directives to redirect non-ssl traffic over to ssl:

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

9. Map your urls.py to django-bft's urls. An example of this would be:

```
urlpatterns = patterns('',
        ...
        (r'', include('bft.urls')),
)
```

10. Setup a cron job to handle file archiving and deletion. An example of this could be:

```
#!/bin/sh

./manage.py deleteuploads
./manage.py deletetempfiles
```

11. Don't forget to collect your static files and sync your database:

```
$ python manage.py syncdb
$ python manage.py collectstatic
```