

By Vitor Freitas

I'm a passionate software developer and researcher from Brazil, currently living in Finland. I write about Python, Django and Web Development on a weekly basis. Read more.













TUTORIAL

How to Upload Files With Django

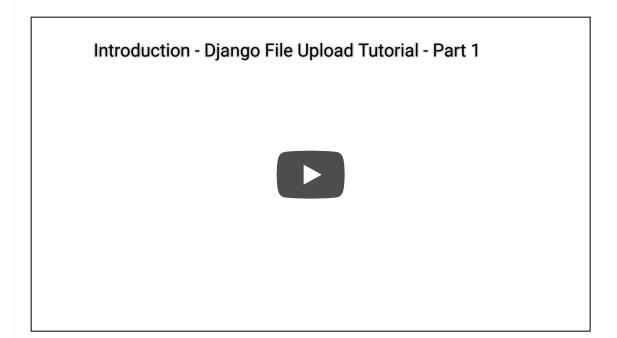


Handling File Upload

1 Updated at Nov 2, 2018: As suggested by @fapolloner, I've removed the manual file handling. Updated the example using FileSystemStorage instead. Thanks!

In this tutorial you will learn the concepts behind Django file upload and how to handle file upload using model forms. In the end of this post you will find the source code of the examples I used so you can try and explore.

This tutorial is also available in video format:



The Basics of File Upload With Django

When files are submitted to the server, the file data ends up placed in request.FILES.

It is mandatory for the HTML form to have the attribute enctype="multipart/form-data" set correctly. Otherwise the request.FILES will be empty.

The form must be submitted using the **POST** method.

Django have proper model fields to handle uploaded files: FileField and ImageField.

The files uploaded to FileField or ImageField are not stored in the database but in the filesystem.

FileField and ImageField are created as a string field in the database (usually VARCHAR), containing the reference to the actual file.

If you delete a model instance containing [FileField] or [ImageField], Django will **not** delete the physical file, but only the reference to the file.

```
The request.FILES is a dictionary-like object. Each key in request.FILES is the name from the <input type="file" name="" />.
```

Each value in request. FILES is an UploadedFile instance.

You will need to set MEDIA_URL and MEDIA_ROOT in your project's settings.py.

```
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

In the development server you may serve the user uploaded files (media) using django.contrib.staticfiles.views.serve() view.

```
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    # Project url patterns...
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_F
```

To access the MEDIA_URL in template you must add django.template.context_processors.media to your context_processeors inside the TEMPLATES config.

Simple File Upload

Following is a minimal file upload example using <code>FileSystemStorage</code>. Use it just to learn about the flow of the process.

simple_upload.html

views.py

```
})
return render(request, 'core/simple_upload.html')
```

File Upload With Model Forms

Now, this is a way more convenient way. Model forms perform validation, automatically builds the absolute path for the upload, treats filename conflicts and other common tasks.

models.py

```
from django.db import models

class Document(models.Model):
    description = models.CharField(max_length=255, blank=True)
    document = models.FileField(upload_to='documents/')
    uploaded_at = models.DateTimeField(auto_now_add=True)
```

forms.py

```
from django import forms
from uploads.core.models import Document

class DocumentForm(forms.ModelForm):
    class Meta:
        model = Document
        fields = ('description', 'document', )
```

views.py

```
def model_form_upload(request):
   if request.method == 'POST':
      form = DocumentForm(request.POST, request.FILES)
      if form.is_valid():
        form.save()
        return redirect('home')
```

```
else:
    form = DocumentForm()
return render(request, 'core/model_form_upload.html', {
    'form': form
})
```

model_form_upload.html

About the FileField upload_to Parameter

See the example below:

```
document = models.FileField(upload_to='documents/')
```

Note the upload_to parameter. The files will be automatically uploaded to MEDIA_ROOT/documents/.

It is also possible to do something like:

```
document = models.FileField(upload_to='documents/%Y/%m/%d/')
```

A file uploaded today would be uploaded to

```
MEDIA_R00T/documents/2016/08/01/.
```

The upload_to can also be a callable that returns a string. This callable accepts two parameters, **instance** and **filename**.

```
def user_directory_path(instance, filename):
    # file will be uploaded to MEDIA_ROOT/user_<id>/<filename>
    return 'user_{0}/{1}'.format(instance.user.id, filename)

class MyModel(models.Model):
    upload = models.FileField(upload_to=user_directory_path)
```

Download the Examples

The code used in this post is available on **Github**.

```
git clone https://github.com/sibtc/simple-file-upload.git

pip install django

python manage.py migrate

python manage.py runserver
```

Related Posts



How to Implement



Advanced Form



How to Use Bootstrap 4
Forms With Django

Grouped Model Choice Rendering with Django Field Crispy Forms

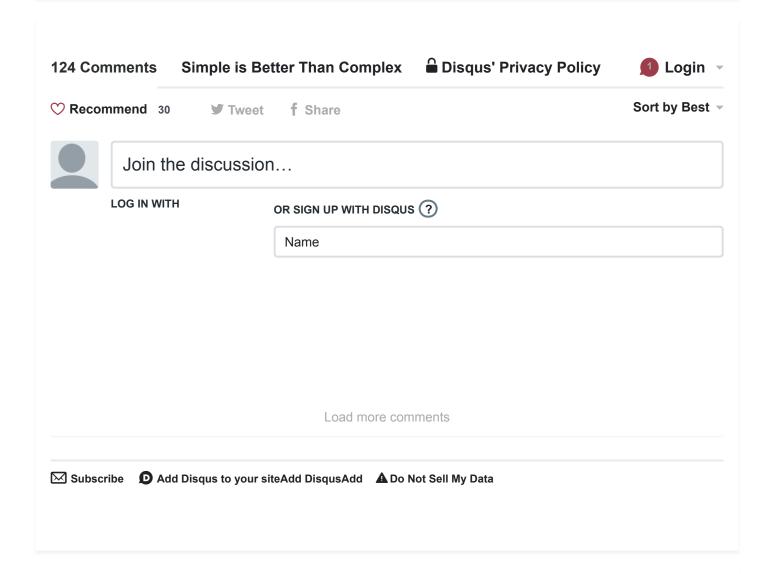
django

Share this post w y f in

media

upload

forms



Subscribe to our Mailing List

Receive updates from the Blog!

Your email address

SUBSCRIBE

Popular Posts



How to Extend Django User Model



How to Setup a SSL Certificate on Nginx for a Django Application



How to Deploy a Django Application to Digital Ocean

© 2015-2019 simple is better than complex cc by-nc-sa 3.0 // about contact faq cookies privacy policy