Django File Uploads: How to Upload Images and Files

Jul 14, 2021

Django · *7 min read*

Today I'm going to cover how to add user document and image file uploads to your Django project as a simple HTML form and Django ModelForm.

I'm using Bootstrap 5 for some basic styling.

Django Media Files

First configure your media upload settings before creating and uploading the form.

Add Media URL to Django settings

mysite > mysite > settings.py

```
MEDIA_URL = '/media/'

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

Head over to the *settings.py* file and specify the `MEDIA_URL` and `MEDIA_ROOT`. They are not specified by default.

Install Pillow

Windows Command Prompt

```
(env) C:\Users\Owner\Desktop\Code\env\mysite> pip install Pi
```

Also install Pillow for media upload support.

Add Media URL to Django URLs

mysite > mysite > urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings #add this
from django.conf.urls.static import static #add this

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include ('main.urls')),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_
```

Next, go to the *mysite > urls.py* and add the helper function above.

Keep in mind it only works when `DEBUG` is set to `True` and the URL specified in the settings is local (i.e. `/media/` not `https://media.site.com/`).

Now, when an image is uploaded in development, it is added to a media directory located in *mysite* > *media*.

For production, you'll want to serve your media files for something like Amazon CloudFront for faster rendering.

Django File/Image Upload: Simple Form

It's time to create a simple form that uploads files using the Django file system.

This example makes it easier to understand the file upload process. However, I recommend using a Django ModelForm for user file uploads to easily reference the data.

HTML template for simple file uploads

mysite > *main* > *templates* > *main* > *(New File) upload.html*

```
            <h2 class="my-4">Add a new movie script</h2>
            <form method="post" enctype="multipart/form-data">
              {% csrf_token %}
                  <input type="file" name="upload" accept=".do
                  <br>
                  <button class="btn btn-dark my-4" type="subm
            </form>

            <!--See File Uploaded-->
            <h2 class="my-4">Movie Script</h2>
            <div class="row">
              {% if uploaded_file_url %}
              <div class="col-lg-4 col-md-6 col-sm-12 pb-4">
                <p>Uploaded to:<a href="{{ uploaded_file_url }
                <a class="btn btn-dark my-4" href="/">Return t
              {% else %}
              <p>No movies scripts added.</p>
              {% endif %}
            </div>

          </div>


{% endblock %}
```

Go to your HTML template and add a `<form>` element. Add the attributes `method="post"` and `enctype="multipart/form-data"`.

`POST` is used to send the data to the server while `enctype` specifies how the form-data should be encoded when sent to the server.

Both of these attributes are required to post file uploads.

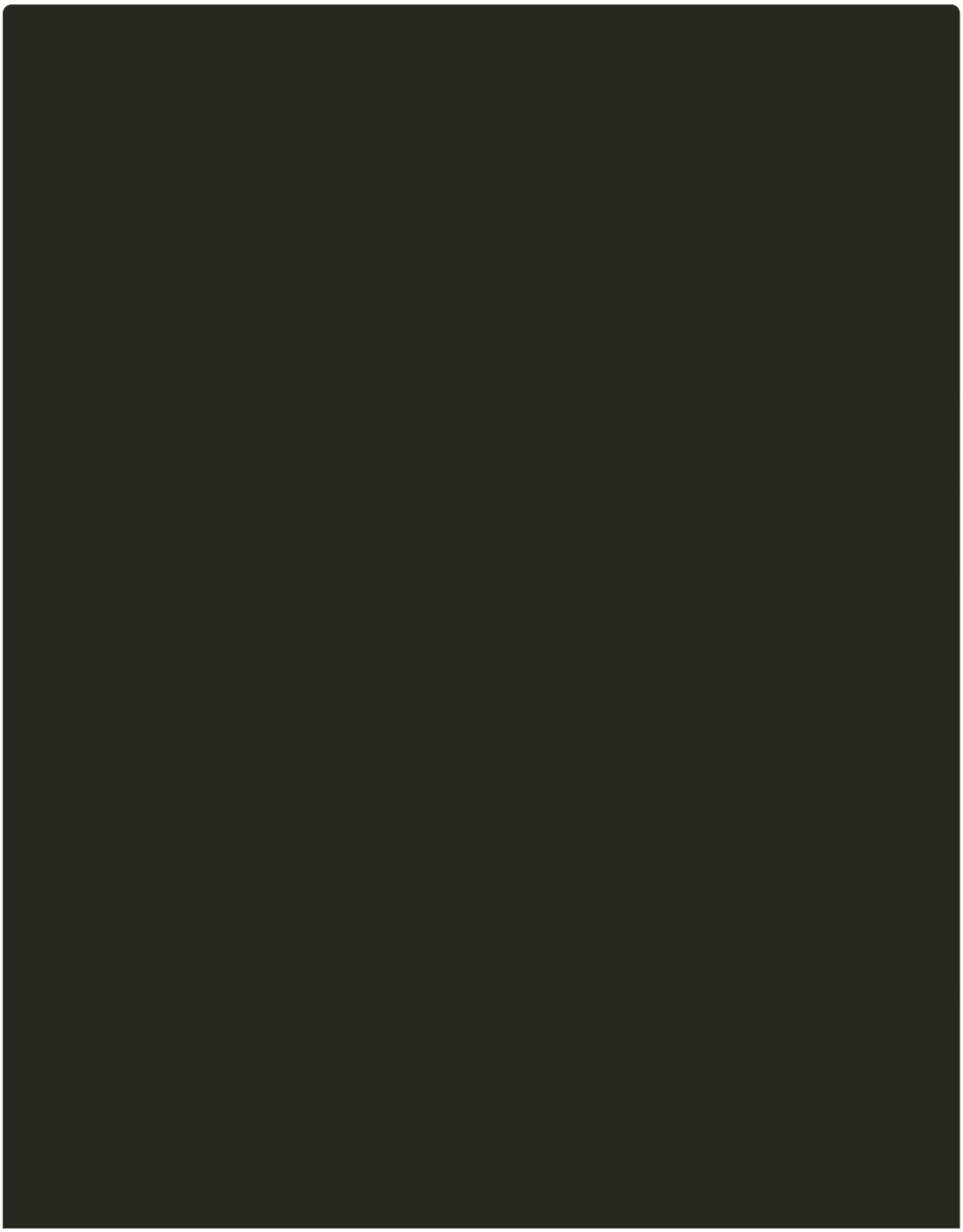
For this simple form, where just using a basic input field specified as a file upload.

Add the `accept` attribute value to define the accepted file types. We'll accept any pdf, text file, or Word Document.

Once the file is uploaded, the URL is called using context pass in the views.

HTML template for simple image uploads

mysite > main > templates > main > upload.html



```
        </div>


{% endblock %}
```

Go to your HTML template and add a `<form>` element with attributes `method="post"` and `enctype="multipart/form-data"`.

Were going to accept any image file for this input field.

Once the image file is uploaded, we'll display the image using context form the views.

Update urls.py

mysite > main > urls.py

```
from django.urls import path
from . import views

app_name = "main"


urlpatterns = [
    path("", views.homepage, name="homepage"),
    path("upload", views.upload, name="upload")
]
```

Update the *main > urls.py* if you added a new HTML template.

Update views.py

mysite > main > views.py

```
from django.shortcuts import  render
from django.core.files.storage import FileSystemStorage

def upload(request):
    if request.method == 'POST' and request.FILES['upload']:
        upload = request.FILES['upload']
        fss = FileSystemStorage()
        file = fss.save(upload.name, upload)
        file_url = fss.url(file)
        return render(request, 'main/upload.html', {'file_ur
    return render(request, 'main/upload.html')
```

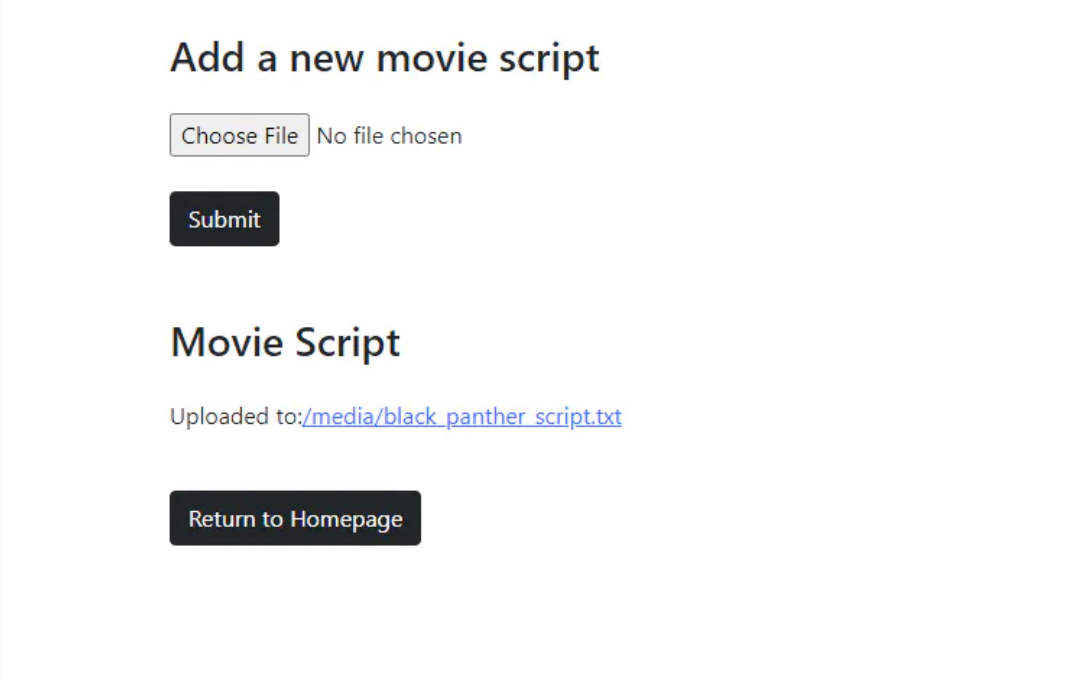
The views are the same for both the file and image upload. You need to import the Django `FileSystemStorage`, a class that implements local file system storage.

File uploads are received and bound to the form in `request.FILES`. However, this only works if the `enctype` is specified in the HTML form.

The uploaded file is then accessible via `request.FILES['upload']`, saved in the `FileSystemStorage`, and returned as `uploaded_file_url` for the context in our template.

The default upload location is the `MEDIA_ROOT` in the settings unless overridden.

The `FileSystemStorge` class inherits from the Django `Storage` class, which contains the `save` and `url` methods used above.


Add a new movie script
Choose File
No file chosen
Submit
Movie Script
Uploaded to:/media/black_panther_script.txt
Return to Homepage

Django File Upload: Django ModelForm

Now let's move on to ModelForms. Use a Django ModelForm if you're looking to save the information uploaded by users in a model object that can be referenced and loaded into any template.

Create a file model field

mysite > main > models.py

```
from django.db import models

# Create your models here.
class Movies(models.Model):
    file = models.FileField(upload_to='documents/', None=True
    image = models.ImageField(upload_to='images/', None=True)
```

Go to *models.py* and add a `FieldField()` with a specified media file upload path.

This is also an example of an `ImageField()` upload as a model field.

`upload_to` automatically uploads to the media ROOT specified in the settings. You do **not** need to specify media in `upload_to`.

The upload location isn't important, but it should be named appropriately (i.e. documents uploaded to the */media/documents/* directory, images to the */media/images/* directory).

Create Django form fields from Django model fields

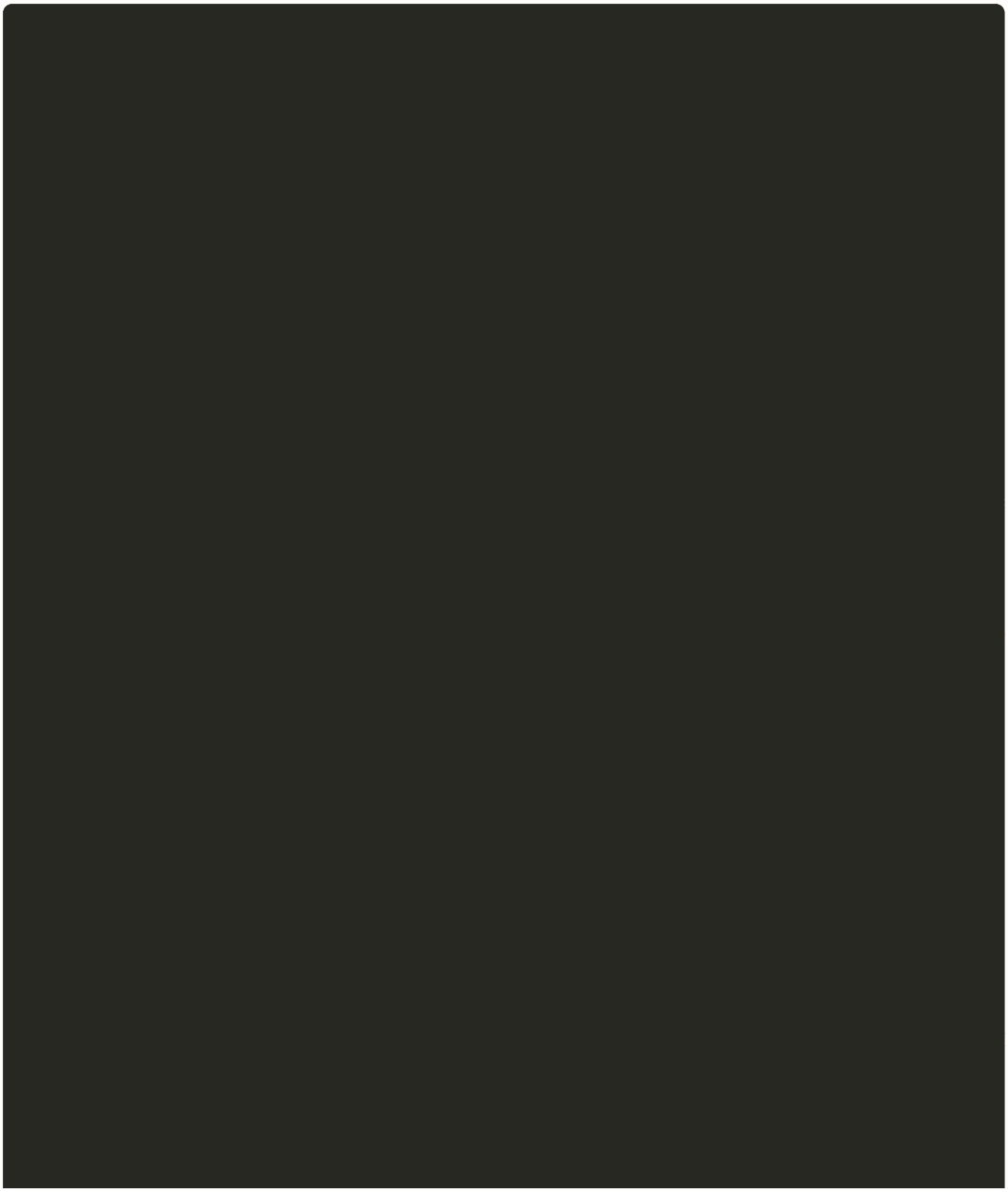
mysite > main > forms.py

```
from django import forms
from .models import Movie


# Create your forms here.
```

```
class MovieForm(forms.ModelForm):

    class Meta:
        model = Movie
        fields = ('file', 'image')
```

Now head over to the *forms.py* file, import the `Movie` model and override the form behavior by adding the model field as the form field.

Add the ModelForm to the HTML template



```
            </div>
            {% empty %}
            <p>No movies added.</p>
            {% endfor %}
          </div>

        </div>


{% endblock %}
```

Now you can call `{{form}}` which holds all of the form's fields and attributes set in *forms.py*.

This is the equivalent of the inputs added earlier in the Simple form.

Open your developer tools to see:

Configure django-crispy-forms and add the `crispy` filter to the form if you want to style your Django form.

You can also add a for loop that iterates all of the existing model objects saved under the Movie model so you can call the file being uploaded in the template.

Create a view function

mysite > main > views.py

```
from django.shortcuts import render, redirect
from .forms import MovieForm
from .models import Movie


# Create your views here.
def homepage(request):
    return render(request, 'main/home.html')

def upload(request):
    if request.method == "POST":
        form = MovieForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
        return redirect("main:upload")
    form = MovieForm()
    movies = Movie.objects.all()
    return render(request=request, template_name="main/uploa
```

Now on to the views, which as you can see are now simplified by using a Django ModelForm.

All we need to do is pass in `request.POST` and `request.FILES` into the form and save the form.

Working with a ModelForm means this saved form will automatically be saved as a Model object under the Movie model.

There is not additional code necessary to create the model objects.

You can then call the uploads as context by passing in the model objects to the template.

If you're looking to learn more about Django ModelForms, checkout this guide.

Django

Follow us @ordinarycoders

POST A COMMENT

Join the community

Comment

Tif_8ce May 12, 2022

i can't upload anything and it doesn't work it. there's no error to. anyone can help me?

Reply

Luke_l8l Mar 23, 2022

I can't view my uploads on the website. For context, I can upload files just fine. However, every time I do so, they go to a media folder in the parent directory just outside my ~mysite~ directory. Any idea as to why this is happening?

Reply

8

WRITTEN BY
JAYSHA

Hello! I enjoy learning about new CSS frameworks, animation libraries, and SEO.

Dec 05, 2021

What are RSS feeds in Django?: How to Generate your own Django RSS Feed

Django · *5 min read*

Nov 23, 2021

How to write tests in Django: Best practices for testing in Django

Django · *9 min read*

A Django Rest Framework Guide

Django · *7 min read*

Jul 05, 2021

A Guide to the Django ModelForm - Creating Forms from Models

Django · *5 min read*

BLOG

Django

Bootstrap

React

Beginners

Tailwind

JavaScript

Python

Beginners

CONTACT

info@ordinarymedia.com