



# Python | Uploading images in Django

Last Updated : 03 Jun, 2024

In most websites, we often deal with media data such as images, files, etc. In Django, we can deal with the images with the help of the model field which is ImageField. In this article, we have created the app image\_app in a sample

Trending Now DSA Web Tech Foundational Courses Data Science Practice Problem Python Machine Learning

Prerequisite – [Introduction to Django](#)

## Python



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

**MEDIA\_ROOT** is for server path to store files in the computer. **MEDIA\_URL** is the reference URL for the browser to access the files over Http. In the urls.py we should edit the configuration like this

```
from django.conf import settings
from django.conf.urls.static import static
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,
                           document_root=settings.MEDIA_ROOT)
```

A sample models.py should be like this, in that we have created a **Hotel model** which consists of hotel name and its image. In this project we are taking the hotel name and its image from the user for hotel booking website.

## Python



```
1 # models.py
2 class Hotel(models.Model):
3     name = models.CharField(max_length=50)
4     hotel_Main_Img = models.ImageField(upload_to='images/')
```

Here upload\_to will specify, to which directory the images should reside, by default django creates the directory under **media** directory which will be automatically created when we upload an image. No need of explicit creation of **media** directory. We have to create a forms.py file under image\_app, here we are dealing with **model form** to make content easier to understand.

### Python



```
1 # forms.py
2 from django import forms
3 from .models import Hotel
4
5
6 class HotelForm(forms.ModelForm):
7
8     class Meta:
9         model = Hotel
10        fields = ['name', 'hotel_Main_Img']
```

Django will implicitly handle the form verification's with out declaring explicitly in the script, and it will create the analogous form fields in the page according to model fields we specified in the models.py file. This is the advantage of model form. Now create a **templates** directory under image\_app in that we have to create a html file for uploading the images. HTML file should look like this.

### html



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
```

```
        <title>Hotel_image</title>
6    </head>
7    <body>
8        <form method = "post" enctype="multipart/form-data">
9            {% csrf_token %}
10           {{ form.as_p }}
11           <button type="submit">Upload</button>
12       </form>
13 </body>
14 </html>
```

When making a POST request, we have to encode the data that forms the body of the request in some way. So, we have to specify the encoding format in the form tag. multipart/form-data is significantly more complicated but it allows entire files to be included in the data. The csrf\_token is for protection against Cross-Site Request Forgeries. form.as\_p simply wraps all the elements in HTML paragraph tags. The advantage is not having to write a loop in the template to explicitly add HTML to surround each title and field. In the views.py under image\_app in that we have to write a view for taking requests from user and gives back some html page.

### Python



```
1  from django.http import HttpResponseRedirect
2  from django.shortcuts import render, redirect
3  from .forms import HotelForm
4
5  # Create your views here.
6
7
8  def hotel_image_view(request):
9
10     if request.method == 'POST':
11         form = HotelForm(request.POST, request.FILES)
12
13         if form.is_valid():
14             form.save()
15             return redirect('success')
16     else:
17         form = HotelForm()
```

```
        return render(request, 'hotel_image_form.html', {'form':
form})
19
20
21 def success(request):
22     return HttpResponse('successfully uploaded')
```

whenever the hotel\_image\_view hits and that request is POST, we are creating an instance of model form form = HotelForm(request.POST, request.FILES) image will be stored under request.FILES one. If it is valid save into the database and redirects to success url which indicates successful uploading of the image. If the method is not POST we are rendering with html template created. urls.py will look like this –

### Python



```
1 from django.contrib import admin
2 from django.urls import path
3 from django.conf import settings
4 from django.conf.urls.static import static
5 from .views import hotel_image_view
6
7 urlpatterns = [
8     path('image_upload', hotel_image_view,
9         name='image_upload'),
10    path('success', success, name='success'),
11 ]
12
13 if settings.DEBUG:
14     urlpatterns += static(settings.MEDIA_URL,
15                             document_root=settings.MEDIA_ROOT)
```

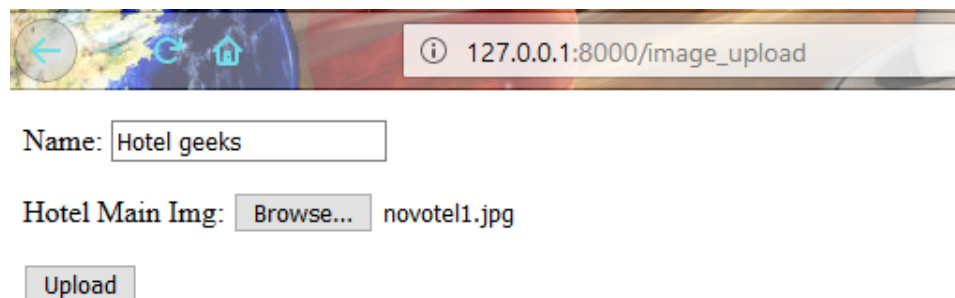
Output:

```
C:\WINDOWS\system32\cmd.exe - python manage.py runserver

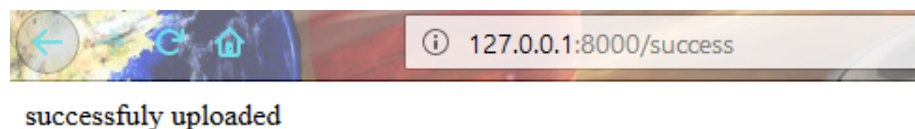
F:\Summer\image_upload>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
October 22, 2018 - 07:46:38
Django version 2.0.5, using settings 'image_upload.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

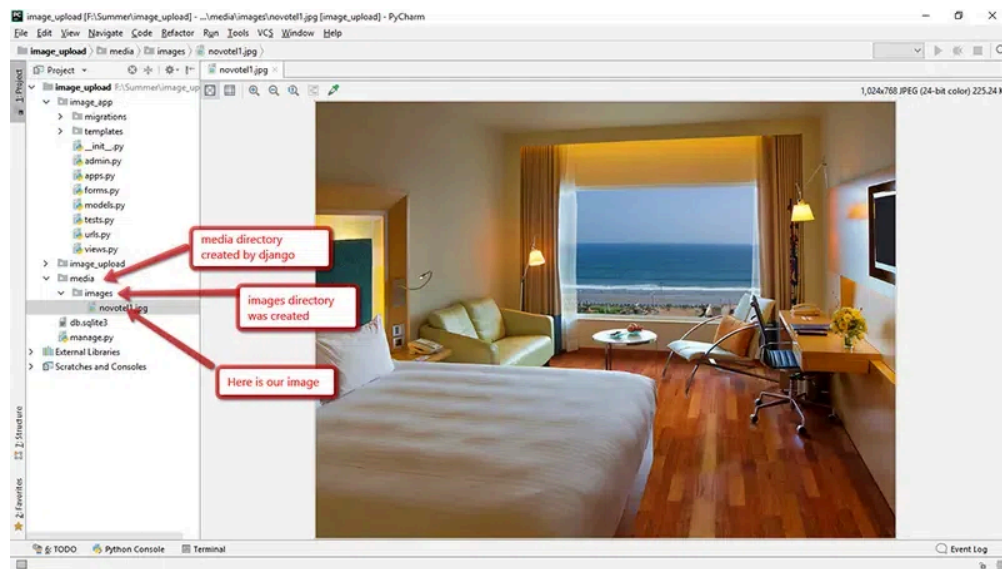
Now make the migrations and run the server. When we hit the URL in the browser, in this way it looks.



After uploading the image it will show success.



Now in the project directory media directory will be created, an images directory will be created and the image will be stored under it. Here is the final result.



Now we can write a view for accessing those images, for simplicity let's take example with one image and it is also applicable for many images.

### Python



```

1  # Python program to view
2  # for displaying images
3
4
5  def display_hotel_images(request):
6
7      if request.method == 'GET':
8
9          # getting all the objects of hotel.
10         Hotels = Hotel.objects.all()
11         return render((request, 'display_hotel_images.html',
12                        {'hotel_images': Hotels}))

```

A sample html file template for displaying images.

### html



```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Hotel Images</title>

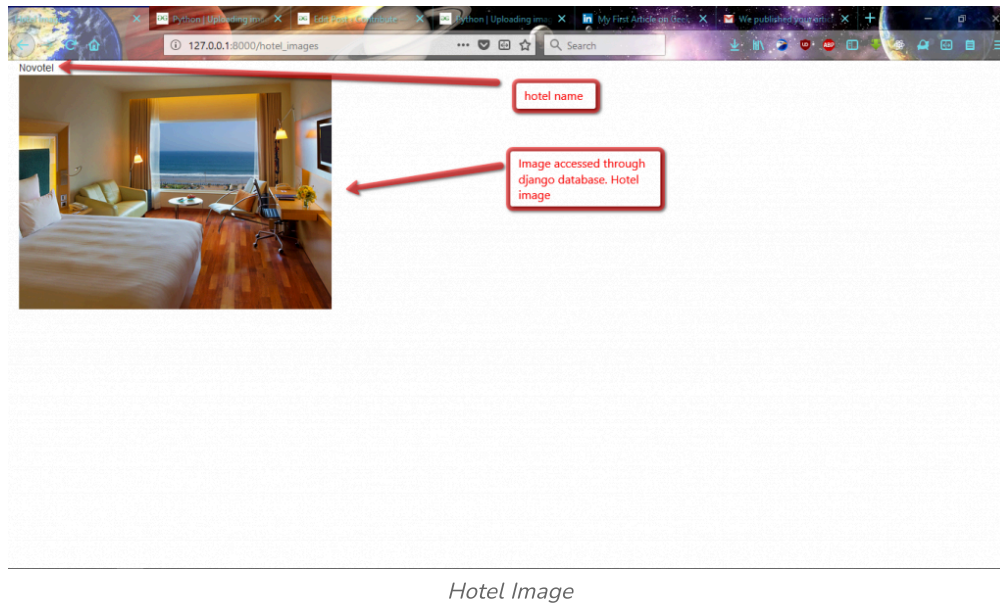
```

```
7     <meta name="viewport" content="width=device-width,
initial-scale=1">
8     <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bo
otstrap.min.css">
9     <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jque
ry.min.js">
10    </script>
11    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/boot
strap.min.js">
12    </script>
13 </head>
14 <body>
15
16     {% for hotel in hotel_images %}
17         <div class="col-md-4">
18             {{ hotel.name }}
19             
20         </div>
21     {% endfor %}
22
23 </body>
24 </html>
```

Insert the url path in the urls.py file

```
# urls.py
path('hotel_images', display_hotel_images, name = 'hotel_images'),
```

Here is the final view on the browser when we try to access the image.



Hotel Image

Learn in a distraction-free environment with refined, **high-quality content** and **35+ expert-led tech courses** to help you crack any interview. From programming languages and DSA to web development and data science, [GeeksforGeeks Premium](#) has you covered!

Choose [GeeksforGeeks Premium](#) today and also get access to **Unlimited Article Summarization**, **100% Ad free environment**, **A.I. Bot support** in all coding problems, and much more. [Go Premium!](#)

S Saiva...



### Previous Article

Intermediate fields in Django | Python

### Next Article

Change Object Display Name using  
\_\_str\_\_ function - Django Models | Python

## Similar Reads



## Uploading Image Using Django with Firebase

Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a mor...

4 min read

## Uploading files on Google Drive using Python

In the modern era of the internet, a large number of mundane or work-related tasks are carried out over the internet. Compromise of data stored on a device...

3 min read

## Google Cloud Platform - Ways of Uploading Data to GCS

Before you can harness the power of the cloud, to serve your content, you have to get your data into it. In this article, we will look into all the different ways you ca...

3 min read

## Downloading and Uploading Files from Internet

The terms downloading and uploading are generally used while browsing the Internet. Receiving data or a file from the Internet on your computer is referred t...

7 min read

## Servlet - Uploading File

Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the...

2 min read

## Why would \$\_FILES be empty when uploading files to PHP ?

While uploading the files or writing the code, developers do many mistakes which we cannot figure out most of the time. Some spelling mistakes, some...

3 min read

## Uploading a Reverse Shell to a Web Server in Kali Linux

We basically hack the webserver for gaining access to the system. We look into what is inside the web server and we want to have full control of the web serve...

3 min read

## Uploading and Downloading Files in Flask

This article will go over how to upload and download files using a Flask database using Python. Basically, we have a section for uploading files where we can...

7 min read

## Uploading and Reading a CSV File in Flask

Flask is a flexible, lightweight web-development framework built using python. A Flask application is a Python script that runs on a web server, which listens to...

3 min read

## How to Compress Image in Android Before Uploading it to Firebase Storage?

Usually, in an app we make users upload images like profile images or others. But there is a chance that the uploaded image is only for display purposes, that too i...

4 min read

Article Tags :

[Technical Scripter](#)

[Technical Scripter 2018](#)



Corporate & Communications Address:-  
A-143, 9th Floor, Sovereign Corporate  
Tower, Sector- 136, Noida, Uttar Pradesh  
(201305) | Registered Address:- K 061,  
Tower K, Gulshan Vivante Apartment,  
Sector 137, Noida, Gautam Buddh  
Nagar, Uttar Pradesh, 201305



### Company

[About Us](#)

[Legal](#)

### Languages

[Python](#)

[Java](#)

In Media  
Contact Us  
Advertise with us  
GFG Corporate Solution  
Placement Training Program  
GeeksforGeeks Community

C++  
PHP  
GoLang  
SQL  
R Language  
Android Tutorial  
Tutorials Archive

## DSA

Data Structures  
Algorithms  
DSA for Beginners  
Basic DSA Problems  
DSA Roadmap  
Top 100 DSA Interview Problems  
DSA Roadmap by Sandeep Jain  
All Cheat Sheets

## Web Technologies

HTML  
CSS  
JavaScript  
TypeScript  
ReactJS  
NextJS  
Bootstrap  
Web Design

## Computer Science

Operating Systems  
Computer Network  
Database Management System  
Software Engineering  
Digital Logic Design  
Engineering Maths  
Software Development  
Software Testing

## System Design

High Level Design  
Low Level Design  
UML Diagrams  
Interview Guide  
Design Patterns  
OOAD  
System Design Bootcamp  
Interview Questions

## Data Science & ML

Data Science With Python  
Data Science For Beginner  
Machine Learning  
ML Maths  
Data Visualisation  
Pandas  
NumPy  
NLP  
Deep Learning

## Python Tutorial

Python Programming Examples  
Python Projects  
Python Tkinter  
Web Scraping  
OpenCV Tutorial  
Python Interview Question  
Django

## DevOps

Git  
Linux  
AWS  
Docker  
Kubernetes  
Azure  
GCP  
DevOps Roadmap

## Interview Preparation

Competitive Programming  
Top DS or Algo for CP  
Company-Wise Recruitment Process  
Company-Wise Preparation  
Aptitude Preparation  
Puzzles

## School Subjects

Mathematics  
Physics  
Chemistry  
Biology  
Social Science  
English Grammar  
Commerce  
World GK

## GeeksforGeeks Videos

DSA  
Python  
Java  
C++  
Web Development  
Data Science  
CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved