CRUD[Create-Read-Update-Delete]

Delete:

***✅ Step 1: Model (Student)***

***তোমার student model থাকলেই delete কাজ করবে। (তুমি চাইলে এইটা রাখো)***

***# models.py***

from django.db import models

class Student(models.Model):

name = models.CharField(max\_length=50)

def \_\_str\_\_(self):

return self.name

***✅ Step 2: URL***

***urls.py তে একটা route রাখো:***

from django.urls import path

from . import views

urlpatterns = [

path('delete/<int:id>/', views.delete\_student, name='delete\_student'),

]

***✅ Step 3: View (Delete Function)***

***এবার view লিখি। এখানে get\_object\_or\_404 বা extra জিনিস লাগবে না, শুধু basic delete করবো।***

***# views.py***

from django.shortcuts import redirect

from .models import Student

def delete\_student(request, id):

student = Student.objects.get(id=id) *# student বের করলাম*

student.delete() *# মুছে ফেললাম*

return redirect('home') *# মুছে ফেলার পর home পেজে পাঠিয়ে দিবে*

***✅ Step 4: Template থেকে Delete Button***

***ধরে নাও তোমার template এ student list আছে।  
ওখানে শুধু একটা link রাখলেই delete হবে।***

{% for student in students %}

<p>

{{ student.name }}

<a href="{% url 'delete\_student' student.id %}">Delete</a>

</p>

{% endfor %}

***👉 Workflow***

1. ***List পেজে প্রতিটি student এর পাশে delete button থাকবে।***
2. ***Delete এ চাপ দিলে সরাসরি ডাটাবেস থেকে student মুছে যাবে।***
3. ***তারপর home (বা অন্য পেজে) redirect হবে।***

***আচ্ছা 👍 এবার ধাপে ধাপে input name → views.py → models.py এর পুরো ফ্লোটা দেখাই।***

***input name → views.py → models.py***

***1️⃣ models.py (ডাটাবেসে টেবিল)***

***ধরো তুমি ব্লগ পোস্ট রাখতে চাইছো।  
তাহলে একটা model বানাতে হবে:***

from django.db import models

class Post(models.Model):

title = models.CharField(max\_length=200) *# শিরোনাম*

content = models.TextField() *# কনটেন্ট*

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.title

***👉 এখানে title আর content হচ্ছে ডাটাবেস ফিল্ড।***

***2️⃣ HTML form (input name)***

***তুমি একটা form বানালে add\_post.html এ:***

<form method="post">

{% csrf\_token %}

<label>Title</label>

<input type="text" name="title" class="form-control" required>

<label>Content</label>

<textarea name="content" class="form-control" required></textarea>

<button type="submit" class="btn btn-primary mt-2">Save</button>

</form>

***👉 এখানে***

* ***<input name="title">***
* ***<textarea name="content">***

***এই name হচ্ছে key, যেটা দিয়ে views.py থেকে ডাটা পাওয়া যাবে।***

***3️⃣ views.py (ডাটা নেওয়া ও সেভ করা)***

from django.shortcuts import render, redirect

from .models import Post

def add\_post(request):

if request.method == "POST":

*# এখানে input name="title" থেকে ডাটা নিচ্ছি*

title = request.POST.get("title")

content = request.POST.get("content")

*# models.py তে ডাটা সেভ করলাম*

Post.objects.create(title=title, content=content)

*# সেভ করার পর লিস্ট পেজে পাঠাচ্ছি*

return redirect("post\_list")

return render(request, "add\_post.html")

def post\_list(request):

posts = Post.objects.order\_by('-created\_at')

return render(request, "post\_list.html", {"posts": posts})

***👉 এখানে***

* ***request.POST.get("title") মানে HTML এ name="title" এর ডাটা।***
* ***request.POST.get("content") মানে HTML এ name="content" এর ডাটা।***
* ***এগুলো নিয়ে Post.objects.create(...) দিয়ে ডাটাবেসে সেভ হলো।***

***4️⃣ urls.py***

from django.urls import path

from . import views

urlpatterns = [

path('', views.post\_list, name="post\_list"),

path('add/', views.add\_post, name="add\_post"),

]

***5️⃣ Flow (সহজ ভাষায়)***

1. ***ইউজার form এ লিখলো → input এর name="title" আর name="content"***
2. ***সাবমিট করলে Django request.POST এর ভেতরে এই নামগুলো key হিসেবে রাখে***
   * ***যেমন request.POST['title'] = "My First Blog"***
   * ***request.POST['content'] = "This is my blog content"***
3. ***views.py এ .get("title") আর .get("content") দিয়ে সেই ডাটা পড়া হয়***
4. ***সেই ডাটা models.py এর মাধ্যমে ডাটাবেসে সেভ করা হয়***

***📌 তাই input এর name attribute → views.py তে ব্যবহার হয় (ডাটা নেওয়ার জন্য)।  
📌 models.py তে name attribute এর সাথে direct সম্পর্ক নাই।***

***Upload Images***

**১) Model এ ImageField ব্যবহার করা**

তোমার model এ image ফাইল রাখার জন্য ImageField ব্যবহার করতে হবে।

**models.py**

from django.db import models

class Student(models.Model):

name = models.CharField(max\_length=100)

photo = models.ImageField(upload\_to='photos/') *# image upload হবে "media/photos/" ফোল্ডারে*

def \_\_str\_\_(self):

return self.name

**২) settings.py এ MEDIA সেট করা**

import os

MEDIA\_URL = '/media/'

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

**৩) urls.py তে Media serve করা (Development এ)**

from django.conf import settings

from django.conf.urls.static import static

from django.urls import path

from . import views

urlpatterns = [

path('', views.home, name='home'),

]

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**৪) Form তৈরি করা**

**forms.py**

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

class Meta:

model = Student

fields = ['name', 'photo']

**৫) View লেখা**

**views.py**

from django.shortcuts import render, redirect

from .forms import StudentForm

from .models import Student

def home(request):

if request.method == "POST":

form = StudentForm(request.POST, request.FILES) *# FILES লাগবেই!*

if form.is\_valid():

form.save()

return redirect('home')

else:

form = StudentForm()

students = Student.objects.all()

return render(request, 'home.html', {'form': form, 'students': students})

**৬) Template এ form ব্যবহার করা**

**home.html**

<!DOCTYPE html>

<html>

<head>

<title>Photo Upload Example</title>

</head>

<body>

<h2>Student Photo Upload</h2>

<form method="post" enctype="multipart/form-data">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Upload</button>

</form>

<h2>All Students</h2>

<ul>

{% for student in students %}

<li>

{{ student.name }} <br>

<img src="{{ student.photo.url }}" width="150">

</li>

{% endfor %}

</ul>

</body>

</html>

**৭) Pillow Install করতে হবে**

Image upload এর জন্য Pillow লাগবে। Install করো:

pip install Pillow

👉 এখন তুমি form থেকে ছবি upload করলে সেটা media/photos/ ফোল্ডারে সেভ হবে এবং template এ show করবে।

**Nijer Nam e Media save korte chaile**

def student\_directory\_path(instance, filename):

return os.path.join('student/media', instance.name, filename)

***👉 এই ফাংশনটা upload\_to এর জন্য ব্যবহার করা হচ্ছে।  
মানে, যখন তুমি কোনো Student মডেলের photo (ImageField) তে একটা ছবি আপলোড করবে, Django কে বলে দেবে ফাইলটা কোন path এ save হবে।***

***ব্যাখ্যা করি:***

* ***instance  
  মানে হচ্ছে যেই Student object টা save হবে। যেমন, যদি তুমি একটা student বানাও যার name = "Farhan" তাহলে এখানে instance.name = "Farhan" হবে।***
* ***filename  
  মানে হচ্ছে আপলোড করা ফাইলের আসল নাম (যেটা তুমি select করেছো)।  
  যেমন, যদি তুমি photo.jpg আপলোড করো, তাহলে filename হবে "photo.jpg"।***
* ***os.path.join('student/media', instance.name, filename)  
  এই লাইনটা ফাইনাল ফাইল path বানায়।  
  উদাহরণস্বরূপ, যদি student এর নাম হয় Farhan আর ফাইল হয় photo.jpg, তাহলে ফাইল save হবে এই path এ:***

student/media/Farhan/photo.jpg

***তাই student\_directory\_path এর কাজ:***

***প্রতিটা student এর জন্য আলাদা ফোল্ডার তৈরি করে তার নাম অনুযায়ী, আর তার ভেতরে ছবি save করে।***

***উদাহরণ:***

student = Student.objects.create(

name="Farhan",

email="farhan@gmail.com",

phone="12345",

password="1234"

)

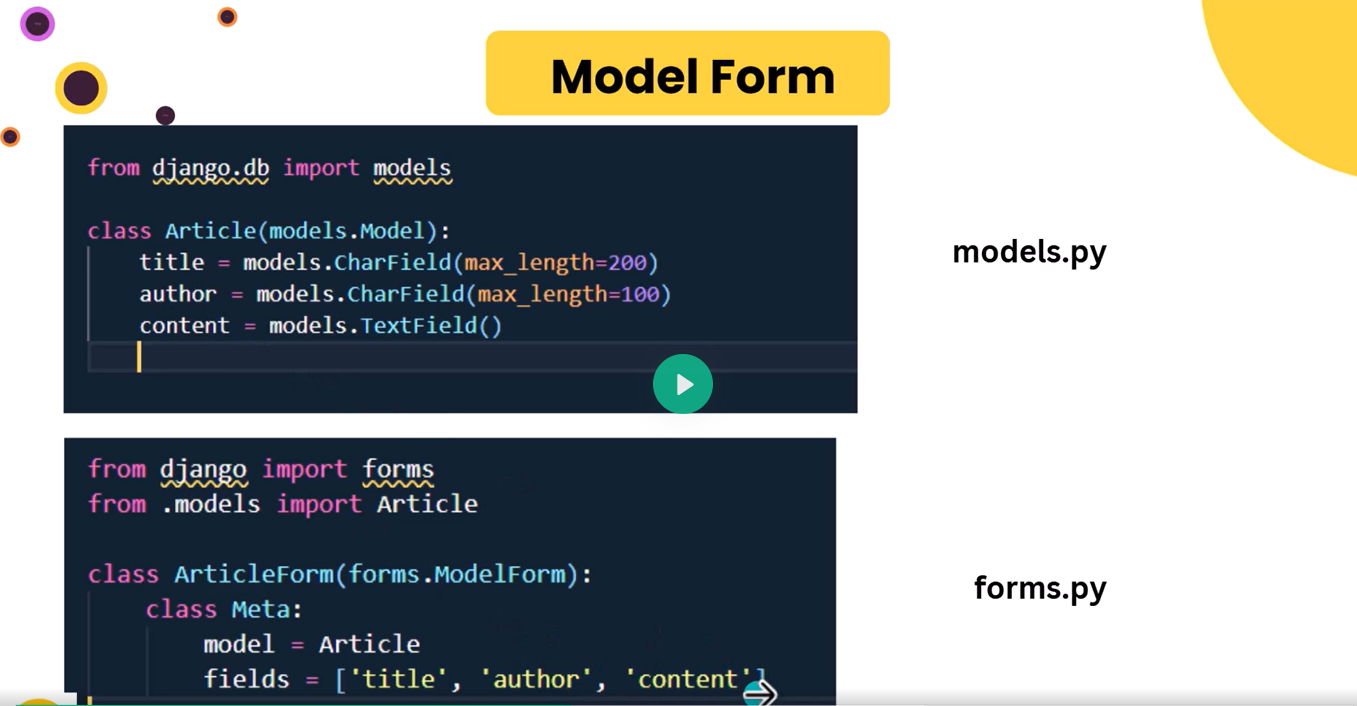
student.photo = "photo.jpg"

student.save()

***এবার ছবিটা চলে যাবে 👉***

media/student/media/Farhan/photo.jpg

***👉 প্রশ্ন: তুমি চাইছো ছবিগুলো media/ ফোল্ডারের ভেতরে save হোক, নাকি আলাদা student/media/ ফোল্ডারেই রাখতে চাইছো? আমি তোমাকে পুরো settings.py MEDIA settings সহ করে দেখাই?***ModelForm



Meta Class

a metaclass is a class that defines how other classes should behave. Provides additional information about the model form. Here are some common options that

can be defined in the Meta class:

model: The model that the form is based on.

fields: A list of fields to include in the form. If option is not specified, all fields in the model will be included in the form.

exclude: A list of fields to exclude from the form.

widgets: A dictionary of field names and their associated widgets.

labels: A dictionary of field names and their associated labels.

help\_texts: A dictionary of field names and their associated help text.

error\_messages: A dictionary of field names and their associated error messages.

**Meta class কী?**

ModelForm এ Meta class হলো **configuration** যেখানে আমরা বলি:

* কোন **Model** এর সাথে Form যুক্ত হবে
* Form এ কোন **Field** গুলো থাকবে
* কোন **Labels, Widgets, Help text** ব্যবহার হবে

প্রকৃতপক্ষে এটা **Form এর behaviour ঠিক করার জায়গা**।

**Meta class এ সাধারণly ব্যবহৃত attribute**

| **Attribute** | **কাজ** | **উদাহরণ** |
| --- | --- | --- |
| model | কোন মডেলের ফিল্ড থেকে Form তৈরি হবে | model = Student |
| fields | কোন ফিল্ডগুলো Form এ থাকবে | fields = ['name', 'email', 'phone'] |
| exclude | কোন ফিল্ডগুলো Form থেকে বাদ থাকবে | exclude = ['password'] |
| labels | ফিল্ডের label পরিবর্তন করা | labels = {'name': 'Student Name'} |
| help\_texts | ফিল্ডের পাশে সাহায্যের টেক্সট | help\_texts = {'email': 'Enter a valid email'} |
| widgets | ফিল্ডের input টাইপ/সাজানো | widgets = {'name': forms.TextInput(attrs={'class':'form-control'})} |

**Example:**

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

class Meta:

model = Student

fields = ['name', 'email', 'phone', 'photo'] *# Form এ কোন ফিল্ড দেখাবে*

labels = {

'name': 'Student Name',

'phone': 'Phone Number'

}

help\_texts = {

'email': 'Enter a valid email address'

}

widgets = {

'name': forms.TextInput(attrs={'class': 'form-control'}),

'email': forms.EmailInput(attrs={'class': 'form-control'}),

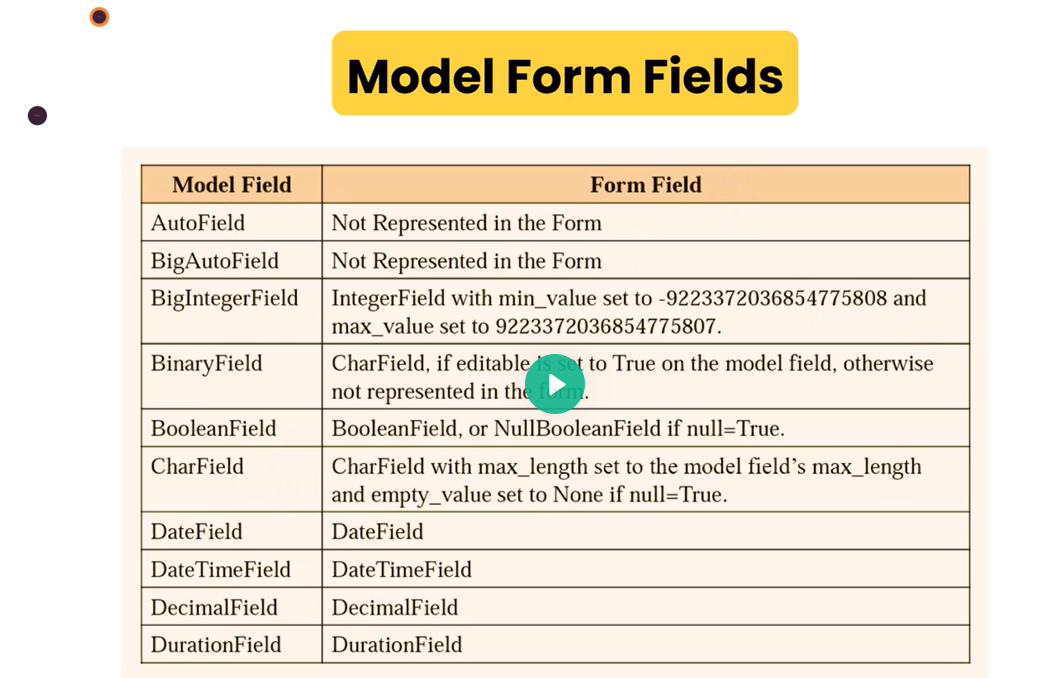
}

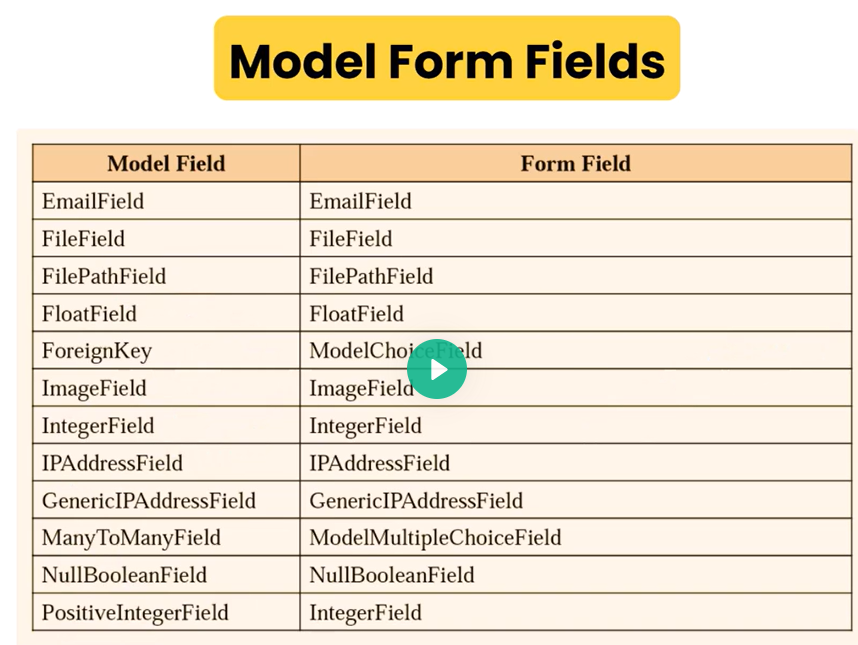
**সহজভাবে বললে:**

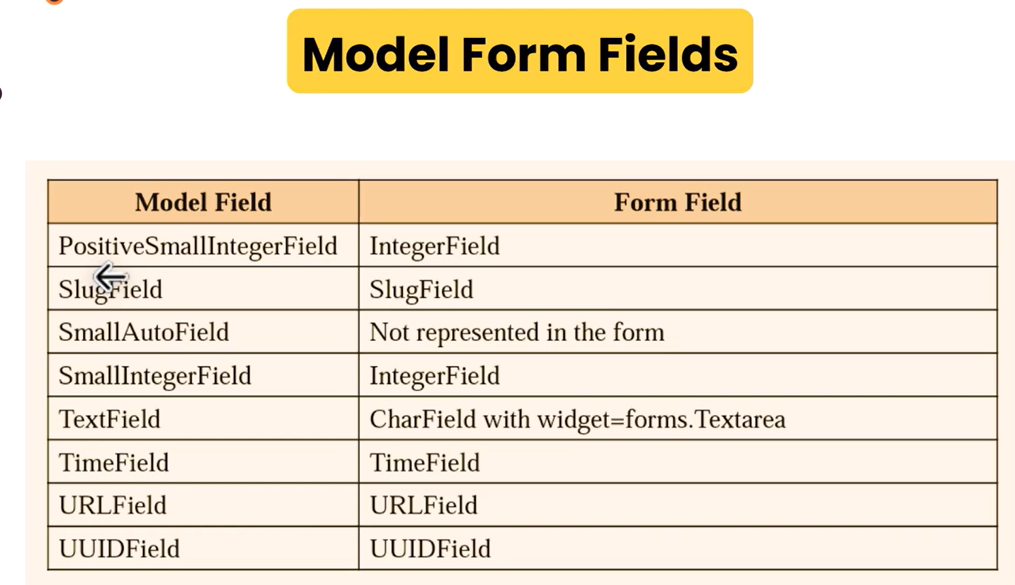
Meta class = **Form কে কীভাবে behave করবে তা define করার জায়গা।**

* কোন Model থেকে আসবে → model
* কোন ফিল্ড দেখাবে → fields বা exclude
* লেবেল, হেল্প, ফিল্ড সাজানো → labels, help\_texts, widgets

**Models Form Fields**

****

****

****

Form

**🔹 Form কি?**

**Form** মানে হলো একটা **Box / Field** যেটার মাধ্যমে **User (ব্যবহারকারী)** থেকে Data নেওয়া হয়।  
যেমন – নাম, ইমেইল, পাসওয়ার্ড, ছবি ইত্যাদি।

ওয়েবসাইটে তুমি যেসব Form দেখ –

* Login Form
* Signup Form
* Contact Form
* File Upload Form

এসবই হলো **Form**।

**🔹 Django Form এর ধরন**

Django তে দু’ধরনের Form আছে –

1. **forms.Form** → Manual (নিজে field লিখতে হয়)
2. **forms.ModelForm** → Automatic (Model থেকে field নিয়ে আসে)

👉 **ModelForm** সবচেয়ে সহজ, কারণ Model এর সাথে সরাসরি কানেক্টেড।

**🔹 Form কিভাবে কাজ করে? (Step by Step)**

**1️⃣ User Form fill করে**

* User নাম, ইমেইল, বয়স লিখে Submit করে।

**2️⃣ Form Data Request আকারে server এ যায়**

* Form এর Data → Django View এ যায় (request.POST)
* যদি File/Photo থাকে → (request.FILES)

**3️⃣ Form Validation হয়**

* Django চেক করে Data ঠিক আছে কি না
  + যেমন Email ফিল্ডে সত্যি Email দেওয়া হলো তো?
  + বয়স সংখ্যা দিলো তো?

**4️⃣ Valid হলে Save হয় (Database এ)**

* Form → Model → Database এ ডাটা ঢুকে যায়।

**5️⃣ Not Valid হলে Error দেখায়**

* যদি ভুল Data দাও (যেমন Email এর জায়গায় সংখ্যা দাও), তাহলে Error Message দেখাবে।

**🔹 ছোট Example**

👉 Model (database structure):

from django.db import models

class Student(models.Model):

name = models.CharField(max\_length=100)

email = models.EmailField()

age = models.IntegerField()

👉 Form (auto create from model):

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

class Meta:

model = Student

fields = ['name', 'email', 'age']

👉 View (Form process):

from django.shortcuts import render, redirect

from .forms import StudentForm

def add\_student(request):

if request.method == "POST": *# submit করলে*

form = StudentForm(request.POST)

if form.is\_valid(): *# চেক করবে valid কিনা*

form.save() *# Database এ Save হবে*

return redirect('success')

else:

form = StudentForm() *# ফাঁকা ফর্ম show করবে*

return render(request, 'add\_student.html', {'form': form})

👉 Template (Form show করার জন্য):

<form method="POST">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Save</button>

</form>

**🔹 Summary (সহজ ভাষায়)**

* Form = User → Data নেওয়ার সিস্টেম।
* Django Form → Data Validation করে।
* ModelForm → Model এর সাথে কানেক্টেড থাকে, তাই ডাটা সহজে Database এ Save হয়।
* Step: **User fill → Server → Validation → Database Save**