**Video file: 93**

**Code file : ch 83**

**Create View in Django 5**

In last we have our own rough notes :

Gpt\_notes:

# 📘 ****CreateView in Django 5****

## 🔹 1. What is CreateView?

* CreateView is a **generic class-based view** that provides a form for creating a new object and saving it directly into the database.
* It **extends FormView** but with automatic model integration.
* Unlike FormView, you don’t need to manually save objects – CreateView handles that for you.

## 🔹 2. Class Hierarchy

View

 └── ProcessFormView

      └── BaseFormView

           └── ModelFormMixin

                └── SingleObjectMixin

                     └── TemplateResponseMixin

                          └── CreateView

👉 This means CreateView:

* Displays a form (from model).
* Saves data to the database when valid.
* Redirects on success.

## 🔹 3. Important Attributes

* **model** → The model class linked to the form.
* **fields** → List of model fields to show in form.
* **form\_class** → Instead of fields, you can pass a custom ModelForm.
* **template\_name** → The template file for rendering form. Default is <modelname>\_form.html.
* **success\_url** → URL to redirect after success (or use get\_absolute\_url).

## 🔹 4. Example 1 – Simple CreateView

### ✅ models.py

from django.db import models

class Student(models.Model):

    name = models.CharField(max\_length=70)

    email = models.EmailField()

    password = models.CharField(max\_length=70)

    def \_\_str\_\_(self):

        return self.name

### ✅ views.py

from django.views.generic.edit import CreateView

from .models import Student

class StudentCreateView(CreateView):

    model = Student

    fields = ['name', 'email', 'password']  # Auto-generate form from model fields

    success\_url = '/thanks/'  # Redirect after successful save

### ✅ urls.py

from django.urls import path

from myapp.views import StudentCreateView

from django.views.generic.base import TemplateView

urlpatterns = [

    path('create/', StudentCreateView.as\_view(), name='studentcreate'),

    path('thanks/', TemplateView.as\_view(template\_name='myapp/thanks.html'), name='thanks'),

]

### ✅ templates/student\_form.html

<!DOCTYPE html>

<html>

<head><title>Create Student</title></head>

<body>

  <form method="POST">

    {% csrf\_token %}

    {{ form.as\_p }}

    <input type="submit" value="Submit">

  </form>

</body>

</html>

👉 That’s all! CreateView will **save data automatically**.

## 🔹 5. Example 2 – Using get\_absolute\_url Instead of success\_url

If you don’t want to hardcode success\_url, use get\_absolute\_url inside the model.

### ✅ models.py

from django.db import models

from django.urls import reverse

class Student(models.Model):

    name = models.CharField(max\_length=70)

    email = models.EmailField()

    password = models.CharField(max\_length=70)

    def get\_absolute\_url(self):

        return reverse("thanks")  # Redirects to thanks page after save

### ✅ views.py

from django.views.generic.edit import CreateView

from .models import Student

class StudentCreateView(CreateView):

    model = Student

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

👉 Now, success\_url is **automatically handled** by get\_absolute\_url.

## 🔹 6. Example 3 – Custom Template Name

By default, Django expects student\_form.html.  
You can override with template\_name:

class StudentCreateView(CreateView):

    model = Student

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

    template\_name = 'myapp/register.html'

## 🔹 7. Example 4 – Customizing Form Without forms.py

Sometimes, you may want to add CSS classes, placeholders, or change widgets dynamically.  
You can override get\_form():

from django import forms

class StudentCreateView(CreateView):

    model = Student

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

    template\_name = 'myapp/register.html'

    def get\_form(self):

        form = super().get\_form()

        form.fields['name'].widget = forms.TextInput(attrs={'class': 'myclass'})

        form.fields['password'].widget = forms.PasswordInput(attrs={'class': 'mypass'})

        return form

## Mostly people use this instead of with out form

## 🔹 8. Example 5 – Customizing Form with forms.py (Recommended ✅)

Better approach: define a ModelForm and use it.

### ✅ forms.py

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

    class Meta:

        model = Student

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

        widgets = {

            'name': forms.TextInput(attrs={'class': 'myclass'}),

            'password': forms.PasswordInput(attrs={'class': 'mypass'}),

        }

### ✅ views.py

from django.views.generic.edit import CreateView

from .models import Student

from .forms import StudentForm

class StudentCreateView(CreateView):

    model = Student

    form\_class = StudentForm   # Use custom form

    template\_name = 'myapp/register.html'  # Must specify since default is student\_form.html

# ✅ ****Summary****

* FormView → Manages form display & validation, but you must **manually save** data.
* CreateView → Extends FormView and automatically **saves objects** into the database.
* Requires: model + fields (or form\_class).
* Success handling:
  + Either success\_url in view.
  + Or get\_absolute\_url() in model.
* Templates:
  + Default → <modelname>\_form.html.
  + Can override with template\_name.
* Form customization:
  + Override get\_form() (not recommended).
  + OR use a forms.py with ModelForm (recommended).

👉 So, **FormView vs CreateView**:

* **FormView** → flexible, generic form handling (no auto-save).
* **CreateView** → model-aware, automatically saves data.

Our own notes

In this we also save the object , that in from view cannot give this

Now see this :

In this this is enough to save the data into dataase :

Models.py:

from django.db import models

# Create your models here.

class Student(models.Model):

    name = models.CharField(max\_length=70)

    email = models.EmailField()

    password = models.CharField(max\_length=70)

Views.py:

from django.shortcuts import render

from django.views.generic.edit import CreateView

from .models import Student

class StudentCreateView(CreateView): # it demands the template with specific name student form

    model = Student # this is our model

    fields = ['name','email','password'] # fields thos who want to show

in this we cannot need other things to save data into the database

urls.py:

from django.urls import path

from myapp import views

urlpatterns = [

    path('create/',views.StudentCreateView.as\_view(),name='studentcreate' ),

]

Templates/student\_form.html:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <form action="" method="POST">

        {{form.as\_p}}

        <input type="text" value="Submit">

    </form>

</body>

</html>

now this is enogh for saving the data but it demonds successor message url so we write that :

views.py:

from django.shortcuts import render

from django.views.generic.edit import CreateView

from .models import Student

class StudentCreateView(CreateView): # it demands the template with specific name student form

    model = Student

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

    success\_url = '/thanks/'   # we set the successurl if data submitted

urls.py:

from django.urls import path

from myapp import views

from django.views.generic.base import TemplateView

urlpatterns = [

    path('create/',views.StudentCreateView.as\_view(),name='studentcreate'),

    path('thanks/',TemplateView.as\_view(template\_name = 'myapp/thanks.html'),name='thanks'),

]

Instead of upper using we use this :

If we don’t go to send or set the success\_url

So we use this :-------------

Models.py:

from django.db import models

from django.urls import reverse

# Create your models here.

class Student(models.Model):

    name = models.CharField(max\_length=70)

    email = models.EmailField()

    password = models.CharField(max\_length=70)

    def get\_absolute\_url(self):

        return reverse("thanks")

views.py:

from django.shortcuts import render

from django.views.generic.edit import CreateView

from .models import Student

class StudentCreateView(CreateView): # it demands the template with specific name student form

    model = Student

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

urls.py:

from django.urls import path

from myapp import views

from django.views.generic.base import TemplateView

urlpatterns = [

    path('create/',views.StudentCreateView.as\_view(),name='studentcreate'),

    path('thanks/',TemplateView.as\_view(template\_name = 'myapp/thanks.html'),name='thanks'),

]

If we want to change the complete template so we use this

Views.py:

class StudentCreateView(CreateView): # it demands the template with specific name student form

    model = Student

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

    template\_name = 'myapp/register.html'

now in this you noticed how we don customize the form because we don’t have forms.py file or others so how we handle that we can handle this llke :

we have one attribute

def get\_form(self)

views.py:

class StudentCreateView(CreateView): # it demands the template with specific name student form

    model = Student

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

    template\_name = 'myapp/register.html'

    def get\_form(self):

        form = super().get\_form()

        form.fields['name'].widget = forms.TextInput(attrs={'class':'myclass'})

        form.fields['password'].widget = forms.TextInput(attrs={'class':'mypass'})

        return form

but mostly people don’t use this they use form and paste it and use it see below :

forms.py:

make this file and use it we can give the widget and all other things that basic forms needs for customize accourding to our need

from django import forms

from .models import Student

class StudentForm(forms.ModelForm):

    class Meta:

        model = Student

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

        widgets = {

            'name' : forms.TextInput(attrs={'class':'myclass'}),

            'password':forms.PasswordInput(attrs={'class':'mypass'}),

        }

and in views.py:

class StudentCreateView(CreateView): # it demands the template with specific name student form

    form\_class = StudentForm

    model = Student

    template\_name = 'myapp/register.html'  # now we must use that because we use form our createview can not give me our default tempalte