**Video file : 92**

**Code file : ch82**

**Form View in Django 5**

# 📘 ****FormView in Django 5****

## 🔹 1. What is FormView?

* FormView is a **generic class-based view (GCBV)** used for **handling HTML forms**.
* It automatically:
  + Displays a form on **GET** request.
  + Processes the form on **POST** request.
  + Validates the form data.
  + Redirects to a success\_url if valid.
  + Shows errors if invalid.

## 🔹 2. Class Hierarchy

FormView inherits from many mixins and base classes:

View

 └── ProcessFormView

      └── BaseFormView

           └── FormMixin

                └── TemplateResponseMixin

                     └── FormView

That means FormView supports:

* Template rendering (TemplateResponseMixin)
* Handling forms (FormMixin, BaseFormView)
* Processing form submission (ProcessFormView)
* Normal HTTP GET/POST methods (View)

## 🔹 3. Important Attributes

* **form\_class** → The form class used (e.g., ContactForm).
* **template\_name** → Template used to render the form.
* **success\_url** → Redirect URL after successful submission.
* **initial** → Dictionary of initial values for form fields.
* **prefix** → Used to prefix field names (useful when multiple forms on same page).

## 🔹 4. Important Methods

* **get\_initial(self)**  
  Returns initial data (by default returns a copy of initial dict).  
  You can override it to provide dynamic initial data.
* **form\_valid(self, form)**  
  Called when form is valid.  
  You can access cleaned data and perform actions (e.g., save to DB).  
  Must return super().form\_valid(form) (redirects to success\_url) or a custom response.
* **form\_invalid(self, form)**  
  Called when form validation fails.  
  Useful to add error messages and return super().form\_invalid(form).
* **get\_context\_data(self, \*\*kwargs)**  
  Adds extra context to template.

## 🔹 5. Example 1 – Simple Contact Form

### ✅ forms.py

from django import forms

class ContactForm(forms.Form):

    name = forms.CharField()

    email = forms.EmailField()

    msg = forms.CharField(widget=forms.Textarea)

### ✅ views.py

from django.shortcuts import HttpResponse

from django.views.generic.edit import FormView

from django.contrib import messages

from .forms import ContactForm

class ContactFormView(FormView):

    template\_name = 'myapp/contact.html'

    form\_class = ContactForm

    success\_url = '/thank\_you/'    # Redirect on success

    initial = {'name': 'soman'}    # Pre-fill form field

    # Called when form is valid

    def form\_valid(self, form):

        print(form.cleaned\_data['name'])

        print(form.cleaned\_data['email'])

        print(form.cleaned\_data['msg'])

        return super().form\_valid(form)

    # Called when form is invalid

    def form\_invalid(self, form):

        messages.error(self.request, 'Invalid form field(s)')

        return super().form\_invalid(form)

    # Add extra context for template

    def get\_context\_data(self, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        context["extra"] = True

        return context

### ✅ urls.py

from django.urls import path

from django.views.generic.base import TemplateView

from .views import ContactFormView

urlpatterns = [

    path('contact/', ContactFormView.as\_view(), name='contactForm'),

    path('thank\_you/', TemplateView.as\_view(

        template\_name='myapp/thank\_you.html'

    ), name='thank\_you'),

]

### ✅ thank\_you.html

<!DOCTYPE html>

<html>

<head><title>Thank You</title></head>

<body>

  <h1>THANK YOU – YOUR FORM HAS BEEN SUBMITTED</h1>

</body>

</html>

## 🔹 6. Example 2 – Form with Database (ModelForm)

### ✅ models.py

from django.db import models

class Students(models.Model):

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

    roll = models.IntegerField(unique=True)

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

    def \_\_str\_\_(self):

        return self.name

### ✅ forms.py

from django import forms

from .models import Students

class Student\_Contact\_Form(forms.ModelForm):

    class Meta:

        model = Students

        fields = ['name', 'roll', 'course']

### ✅ views.py

from django.shortcuts import HttpResponse

from django.views.generic.edit import FormView

from django.contrib import messages

from .forms import Student\_Contact\_Form

from .models import Students

class StudentFormView(FormView):

    template\_name = 'myapp/register.html'

    form\_class = Student\_Contact\_Form

    def form\_valid(self, form):

        # Save form data to database

        student = Students(

            name=form.cleaned\_data['name'],

            roll=form.cleaned\_data['roll'],

            course=form.cleaned\_data['course']

        )

        student.save()

        return HttpResponse("Your form has been submitted successfully!")

    def form\_invalid(self, form):

        messages.error(self.request, 'Invalid form field(s)')

        return super().form\_invalid(form)

### ✅ urls.py

from django.urls import path

from .views import StudentFormView

urlpatterns = [

    path('register/', StudentFormView.as\_view(), name='register'),

]

### ✅ register.html

<h2>Student Registration</h2>

<form method="post">

    {% csrf\_token %}

    {{ form.as\_p }}

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

</form>

## 🔹 7. Flow of FormView

1. **GET request** → Renders form.
2. **POST request** → Validates form.
   * If valid → form\_valid() → Redirects to success\_url.
   * If invalid → form\_invalid() → Re-renders form with errors.

# ✅ ****Summary****

* FormView = shortcut for forms (instead of manually writing if request.method == 'POST' in FBVs).
* Use form\_valid() → to process and save data.
* Use form\_invalid() → to handle errors.
* Use success\_url → redirect after success.
* Can work with both Form and ModelForm.