**Video file: 91**

**Code file : ch81**

**Detail View in Django 5**

# Generic Class-Based View: ****DetailView**** in Django 5

## ✅ 1. What is DetailView?

* DetailView is a **Generic Class-Based View (GCBV)** used to display the details of a **single object** from a model.
* It saves us from writing repetitive queries like Student.objects.get(pk=pk).
* By default:
  + **Template name** → <app>/<model>\_detail.html
  + **Context variable** → <model> (e.g., student)

## ✅ 2. Inheritance Chain of DetailView

DetailView → BaseDetailView → SingleObjectMixin → View

           → TemplateResponseMixin

* **View** → handles HTTP methods (get, post).
* **SingleObjectMixin** → fetches a single object (get\_object()).
* **BaseDetailView** → combines View + SingleObjectMixin.
* **TemplateResponseMixin** → handles rendering templates.
* **DetailView** → final class combining both.

👉 So DetailView = single object fetch + template rendering.

## ✅ 3. Manual Implementation (Without DetailView)

# views.py

from django.shortcuts import render

from django.views import View

from myapp.models import Student

class SingleStudentView(View):

    def get(self, request, pk):

        single\_student\_data = Student.objects.get(pk=pk)

        return render(request, 'myapp/single\_student\_data.html', {'single\_student': single\_student\_data})

# urls.py

from django.urls import path

from myapp import views

urlpatterns = [

    path('<int:pk>/', views.SingleStudentView.as\_view(), name='single\_students'),

]

<!-- single\_student\_data.html -->

<h3>

  {{ single\_student.name }}

  {{ single\_student.roll }}

  {{ single\_student.course }}

</h3>

d

## ✅ 4. Using DetailView

### (a) Basic Usage

# views.py

from django.views.generic.detail import DetailView

from myapp.models import Student

class StudentDetailView(DetailView):

    model = Student

# urls.py

path('student/<int:pk>', views.StudentDetailView.as\_view(), name='student\_detail'),

<!-- student\_detail.html (default template name) -->

{{ student.name }}

{{ student.roll }}

{{ student.course }}

⚠️ Note: DetailView requires pk (or slug) in URL. Passing anything else causes error.

### (b) Custom Primary Key URL Parameter (pk\_url\_kwarg)

# views.py

class StudentDetailView1(DetailView):

    model = Student

    pk\_url\_kwarg = 'geek'

# urls.py

path('student/<int:geek>', views.StudentDetailView1.as\_view(), name='student\_detail1'),

👉 Here, URL still expects an integer (PK), but instead of pk, we use geek.

### (c) Changing Template Name with template\_name\_suffix

# views.py

class StudentDetailView2(DetailView):

    model = Student

    pk\_url\_kwarg = 'geek'

    template\_name\_suffix = '\_jankari'  # default student\_detail.html → student\_jankari.html

# urls.py

path('student2/<int:geek>', views.StudentDetailView2.as\_view(), name='student\_detail2'),

<!-- student\_jankari.html -->

<h3>

  {{ student.name }}

  {{ student.roll }}

  {{ student.course }}

</h3>

### (d) Completely Changing Template with template\_name

# views.py

class StudentDetailView3(DetailView):

    model = Student

    pk\_url\_kwarg = 'geek'

    template\_name = 'myapp/student.html'  # fully custom name

# urls.py

path('student3/<int:geek>', views.StudentDetailView3.as\_view(), name='student\_detail3'),

<!-- student.html -->

<h3>

  {{ student.name }}

  {{ student.roll }}

  {{ student.course }}

</h3>

### (e) Changing Context Variable with context\_object\_name

# views.py

class StudentDetailView4(DetailView):

    model = Student

    template\_name = 'myapp/newstudent.html'

    context\_object\_name = 'stu'

# urls.py

path('student4/<int:pk>', views.StudentDetailView4.as\_view(), name='student\_detail4'),

<!-- newstudent.html -->

{{ stu.name }}

{{ stu.roll }}

{{ stu.course }}

### (f) Adding Multiple Contexts with get\_context\_data()

# views.py

class StudentDetailView5(DetailView):

    model = Student

    template\_name = 'myapp/students1.html'

    context\_object\_name = 'student'

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

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

        context["students"] = Student.objects.all().order\_by('name')  # extra context

        return context

# urls.py

path('student5/<int:pk>', views.StudentDetailView5.as\_view(), name='student\_detail5'),

<!-- students1.html -->

<h3>

  {{ student.name }} {{ student.roll }} {{ student.course }}

  <hr />

  {% for stu in students %}

    {{ stu.name }} {{ stu.roll }} {{ stu.course }} <br />

  {% endfor %}

</h3>

## ✅ 5. Key Attributes of DetailView

* model → Model class to fetch object from.
* pk\_url\_kwarg → Custom keyword for primary key in URL (default: pk).
* slug\_field & slug\_url\_kwarg → For lookups by slug instead of pk.
* template\_name\_suffix → Appends suffix to default template name.
* template\_name → Fully custom template.
* context\_object\_name → Custom context variable name.
* get\_context\_data() → Add extra context to template.
* get\_object() → Override for custom object retrieval logic.

# 🎯 ****Final Summary****

* DetailView → GCBV for displaying a single object.
* **Inheritance chain**: DetailView → BaseDetailView → SingleObjectMixin + TemplateResponseMixin + View.
* **Defaults**:
  + Template → <app>/<model>\_detail.html
  + Context → <model> (e.g., student)
* **Customization Options**:
  + Change URL param name with pk\_url\_kwarg.
  + Change template suffix with template\_name\_suffix.
  + Use template\_name for a fully custom template.
  + Change context with context\_object\_name.
  + Add multiple contexts with get\_context\_data().

👉 In short: DetailView = **Single Object Display + Template Rendering + Easy Customization**.