**Generic Display Views**

The two following generic class-based views are designed to display data. On many projects they are typically the most commonly used views. This views are used for displaying data from model.

1. DetailView
2. ListView

**ListView**

A page representing a list of objects.

While this view is executing, **self.object\_list** will contain the list of objects (usually, but not necessarily a queryset) that the view is operating upon.

Method Flowchart

1. [setup()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/base/#django.views.generic.base.View.setup)
2. [dispatch()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/base/#django.views.generic.base.View.dispatch)
3. [http\_method\_not\_allowed()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/base/#django.views.generic.base.View.http_method_not_allowed)
4. [get\_template\_names()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/mixins-simple/#django.views.generic.base.TemplateResponseMixin.get_template_names)
5. [get\_queryset()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/mixins-multiple-object/#django.views.generic.list.MultipleObjectMixin.get_queryset)
6. [get\_context\_object\_name()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/mixins-multiple-object/#django.views.generic.list.MultipleObjectMixin.get_context_object_name)
7. [get\_context\_data()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/mixins-multiple-object/#django.views.generic.list.MultipleObjectMixin.get_context_data)
8. [get()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/generic-display/#django.views.generic.list.BaseListView.get)
9. [render\_to\_response()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/mixins-simple/#django.views.generic.base.TemplateResponseMixin.render_to_response)

models.py

from django.db import models

# Create your models here.

class Employee(models.Model):

    empno=models.IntegerField(primary\_key=True)

    ename=models.CharField(max\_length=20)

**insert data inside model**

**views.py**

from django.shortcuts import render

from django.views.generic import ListView

from django.http import HttpResponse

# Create your views here.

from cbvapp.models import Employee

class CBV1(ListView):

    model=Employee

**template**

**employee\_list.html (predefined name model-name\_list.html)**

**template is search inside templates folder. Inside template folder there is a folder with application name**

**templates**

**cbvapp**

**employee\_list.html**

<html>

    <body>

        {% for emp in object\_list %}

        <h2>{{emp.empno}},{{emp.ename}} </h2>

        {% endfor %}

    </body>

</html

**Urls.py**

from django.contrib import admin

from django.urls import path

from cbvapp import views

urlpatterns = [

    path('admin/', admin.site.urls),

    path('cbv/',views.CBV1.as\_view())

]

**Settings.py**

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'cbvapp',

]

TEMPLATE\_PATH=os.path.join(BASE\_DIR,"templates")

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [TEMPLATE\_PATH],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

**ListView is having the following class level attributes**

1. **context\_object\_name**
2. **template\_name**

context\_object\_name attribute is used to define custom query set name or object list name. this object name must be string. If context object name is not defined inside view. Django runtime system create default context object name with “object\_list”

template\_name attribute is used to define custom/user defined template name. if template name is not defined it search for template name with model\_name\_list.html.

**Example:**

class StudentList(ListView):

model=Student

context\_object\_name=”studlist”

template\_name=”studrepo.html”

fields=[“rollno”,”name”]

template 🡪 studrepo.html

{% for stud in studlist %}

{{stud.rollno}} {{stud.name}}

{% endfor %}

**CreateView**

A view that displays a form for creating an object, redisplaying the form with validation errors (if there are any) and saving the object.

**Syntax:**

class view\_class\_name(CreateView):

model=model-name

fields=[list of fields]

success\_url=url-pattern

create one template with modelname\_form.html

CreateView generate form with input fields. Form input fields are created using number fields/columns exists model.

This form is displayed within template as paragraph

Default model form object is name “form”

**models.py**

from django.db import models

# Create your models here.

class Student(models.Model):

    rollno=models.IntegerField(primary\_key=True)

    name=models.CharField(max\_length=20)

    course=models.CharField(max\_length=20)

**views.py**

from django.http import HttpResponse

from django.shortcuts import render

from django.views.generic import CreateView

from cbvapp.models import Student

# Create your views here.

class StudentView(CreateView):

    model=Student

    fields="\_\_all\_\_"

    success\_url="/page1"

def page1\_view(request):

    msg="Student Added"

    return HttpResponse(msg)

student\_form.html

<form method="post">

    {% csrf\_token %}

    {{form.as\_p}}

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

</form>

Urls.py

urlpatterns = [

    path('admin/', admin.site.urls),

    path('',views.StudentView.as\_view()),

    path('page1/',views.page1\_view),

]

Settings.py

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'cbvapp',

]

TEMPLATE\_PATH=os.path.join(BASE\_DIR,"templates")

TEMPLATES = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [TEMPLATE\_PATH],

        'APP\_DIRS': True,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]