**Views are two types**

1. **Function Based Views (FBV)**
2. **Class Based Views (CBV)**

Class-based views provide an alternative way to implement views as Python objects instead of functions. They do not replace function-based views, but have certain differences and advantages when compared to function-based views:

* Organization of code related to specific HTTP methods (**GET**, **POST**, etc.) can be addressed by separate methods instead of conditional branching.
* Object oriented techniques such as mixins (multiple inheritance) can be used to factor code into reusable components.

**Generic View**

**Generic View** does not have any predefined functionality.

A generic class based view inherits **View class** from django.views module.

class <class-name>(View): 🡪 Inheritance

def get(self,request):

pass

def post(self,request):

Class based view override two methods of View class to handle request.

If request method is GET, override get() method of View

If request method is POST,override post() method of View

|  |  |
| --- | --- |
| FBV | CBV |
| def fun1(request):  if request.method==”GET”:  ….  If request.method==”POST”:  …. | class CBV(View):  def get(self,request):  ……  def post(self,request):  ….. |

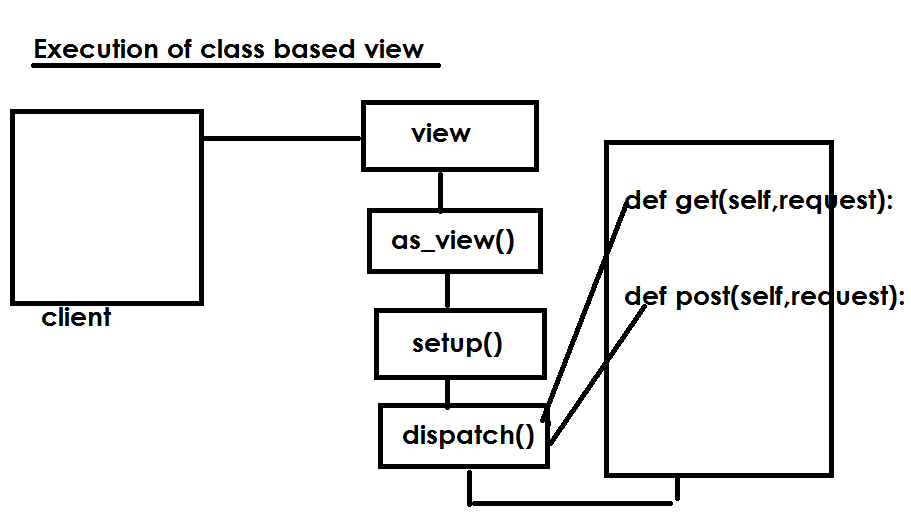
Urls.py

url-pattern

path(‘x/’,views.fun1)

path(‘y/’,views.CBV.as\_view())

Django’s URL resolver expects to send the request and associated arguments to a callable function, not a class, class-based views have an **[as\_view()](https://docs.djangoproject.com/en/4.0/ref/class-based-views/base/" \l "django.views.generic.base.View.as_view" \o "django.views.generic.base.View.as_view)** class method which returns a function that can be called when a request arrives for a URL matching the associated pattern. The function creates an instance of the class, calls [**setup()**](https://docs.djangoproject.com/en/4.0/ref/class-based-views/base/#django.views.generic.base.View.setup) to initialize its attributes, and then calls its [**dispatch()**](https://docs.djangoproject.com/en/4.0/ref/class-based-views/base/#django.views.generic.base.View.dispatch) method. **dispatch** looks at the request to determine whether it is a **GET**, **POST**, etc, and relays the request to a matching method if one is defined, or raises **[HttpResponseNotAllowed](https://docs.djangoproject.com/en/4.0/ref/request-response/" \l "django.http.HttpResponseNotAllowed" \o "django.http.HttpResponseNotAllowed)** if not:



**Views.py**

from django.shortcuts import render

from django.views import View

from django.http import HttpResponse

# Create your views here.

class CBV1(View):

    def get(self,request):

        msg="<h1> This is Get Method of Class Based View </h1>"

        resp=HttpResponse(msg)

        return resp

**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',

]

Class based view which read data from model

Models.py

class Employee(models.Model):

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

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

python manage.py makemigrations

python manage.py migrate

**Views.py**

from django.shortcuts import render

from django.views import View

from django.http import HttpResponse

# Create your views here.

from cbvapp.models import Employee

class CBV1(View):

    def get(self,request):

        qs=Employee.objects.all()

        output=""

        for emp in qs:

            empno=emp.empno

            ename=emp.ename

            output=output+str(empno)+str(ename)+"<br>"

        resp=HttpResponse(output)

        return resp

**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