Writing CBV by using Mixin class:

mixins.py:

from django.http import JsonResponse

class JsonResponseMixin(object):

def render\_to\_json\_response(self,context,\*\*kwargs):

return JsonResponse(context,\*\*kwargs)

CBV:

from testapp.mixins import JsonResponseMixin

class JsonCBV2(JsonResponseMixin,View):

def get(self,request,\*args,\*\*kwargs):

employee\_data={'eno':100,'ename':'Naresh','esal':1000,'eaddr':'Hyderabad'}

return self.render\_to\_json\_response(employee\_data)

Performing database CRUD operations by using web api without

REST Framework:

models.py

from django.db import models

# Create your models here.

class Employee(models.Model):

eno=models.IntegerField()

ename=models.CharField(max\_length=64)

esal=models.FloatField()

eaddr=models.CharField(max\_length=64)

admin.py

from django.contrib import admin

from testapp.models import Employee

# Register your models here.

class EmployeeAdmin(admin.ModelAdmin):

list\_display=['id','eno','ename','esal','eaddr']

admin.site.register(Employee,EmployeeAdmin)

views.py

from django.shortcuts import render

from django.views.generic import View

from testapp.models import Employee

import json

from django.http import HttpResponse

# Create your views here.

class EmployeeCRUDCBV(View):

def get(self,request,\*args,\*\*kwargs):

emp=Employee.objects.get(id=2)

data={

'eno':emp.eno,

'ename':emp.ename,

'esal':emp.esal,

'eaddr':emp.eaddr,

}

json\_data=json.dumps(data)

return HttpResponse(json\_data,content\_type='application/json'

Without Hardcoding id Value

def get(self,request,id,\*args,\*\*kwargs):

emp=Employee.objects.get(id=id)

....

urls.py

from django.contrib import admin

from django.urls import path

from app1 import views

urlpatterns = [

path('admin/', admin.site.urls),

path('get/<int:id>/',views.EmployeeCRUDCBV.as\_view())

]

test.py

import requests

BASE\_URL='http://127.0.0.1:8000/'

ENDPOINT='api/'

n=input('Enter required id:')

r=requests.get(BASE\_URL+ENDPOINT+n+'/')

data=r.json()

print(data)

Serialization:

The process of converting object from one form to another form is called serialization.

Eg: converting python dictionary object to json

json\_data = json.dumps(data)

Serialization by using django.core.serializers Module:

django provides inbuilt module serializers to perform serialization very easily.This module contains serialize() function for this activity.

def get(self,request,id,\*args,\*\*kwargs):

emp=Employee.objects.get(id=id)

json\_data=serialize('json',[emp,],fields=('eno','ename'))

return HttpResponse(json\_data,content\_type='application/json')

If we are not specifying fields attribute, then all fields will be included in json data. For security reasons, if we don't want to provide access to some fields then this fields attribute is very helpful.

Note: Here exclude attribute is not allowed

To get all Records:

class EmployeeListCBV(View):

def get(self,request,\*args,\*\*kwargs):

qs=Employee.objects.all()

json\_data=serialize('json',qs)

return HttpResponse(json\_data,content\_type='application/json')

urls.py

path('list/',views.EmployeeListCBV.as\_view())

test.py

import requests

BASE\_URL='http://127.0.0.1:8000/'

ENDPOINT='list/'

r=requests.get(BASE\_URL+ENDPOINT)

data=r.json()

print(data)

Note: In the output we are getting some extra meta information also.

[{'model': 'testapp.employee', 'pk': 1, 'fields': {'eno': 100, 'ename': 'naresh', 'esal': 1000.0,

'eaddr': 'Mumbai'}}, {'model': 'testapp.employee', 'pk': 2, 'fields': {'eno': 200, 'ename':

'Suresh', 'esal': 2000.0, 'eaddr

': 'Hyderabad'}}, {'model': 'testapp.employee', 'pk': 3, 'fields': {'eno': 300, 'ename':

'Rajesh', 'esal': 3000.0, 'eaddr': 'Hyderabad'}}, {'model': 'testapp.employee', 'pk': 4, 'fields': {'eno': 400, 'ename': 'Ramesh', '

esal': 4000.0, 'eaddr': 'Bangalore'}}

How to get only Original Database Data:

class EmployeeListCBV(View):

def get(self,request,\*args,\*\*kwargs):

qs=Employee.objects.all()

json\_data=serialize('json',qs)

pdict=json.loads(json\_data)

final\_list=[]

for obj in pdict:

final\_list.append(obj['fields'])

json\_data=json.dumps(final\_list)

return HttpResponse(json\_data,content\_type='application/json')

Output:

[{'eno': 100, 'ename': 'Naresh', 'esal': 1000.0, 'eaddr': 'Mumbai'}, {'eno': 200, 'ename':

'Suresh', 'esal': 2000.0, 'eaddr': 'Hyderabad'}, {'eno': 300, 'ename': 'remash', 'esal': 3000.0,

'eaddr': 'Hyderabad'}, {'eno': 400, 'ename': 'rajesh', 'esal': 4000.0, 'eaddr': 'Bangalore'}]