**Apr 27**

Types of Views:

1.FBV's

2.CBV's

**Class Based View(CBV):**

**-->**FBVs are old where as CBVs are new.

**-->**CBVs are very very easy to use when compared with FBVs. The most commonly used type of views in realtime is CBVs.

**-->**FBVs are more powerful when compared with CBVs. If we are unable to handle with CBVs then only we have to go for FBVs.

CBVs meant for common requirement.

**Ex:**

Read data from Employee table--->CBVs

Complex operations over Employee and Customer tables simultaneously--->FBVs

bootstrap(CBV)

css(FBV)

**Ex:**

D:\Django\_20MAR\_7PM>django-admin startproject cbvproject

D:\Django\_20MAR\_7PM>cd cbvproject

D:\Django\_20MAR\_7PM\cbvproject>py manage.py startapp testapp

-->Add app in settings.py

* **views.py**

from django.views.generic import View

from django.http import HttpResponse

class HelloWorldView(View):

def get(self,request):

return HttpResponse('<h1>This response is from class based view</h1>')

* **urls.py**

path('hello/',views.HelloWorldView.as\_view())

**Note:**

**1.**While defining class based view we have to extend View class.

**2.**To provide response to GET request Djnago will always call get() method. Hence we have to override this mrthod in our view class. Similarly other http methods like post(), put(), delete().......

**3.**While defining url pattern we have to use as\_view() method.

**Template based application by using CBV:**

from django.views.generic import TemplateView

class TemplateCBV(TemplateView):

template\_name = 'testapp/results.html'

* **results.html**

<body>

<h1>Hello this response from template based CBV</h1>

</body>

* **urls.py**

path('tt/',views.TemplateCBV.as\_view()),

**How to send context parameter:**

* **views.py**

class TemplateCBV2(TemplateView):

template\_name = 'testapp/results2.html'

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

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

context['name'] = 'Radhika'

context['marks'] = 98

context['subject'] = 'Python'

return context

* **results2.html**

<body>

<h1>Student Information</h1>

<h2>Student Name:{{name}}</h2>

<h2>Student Marks:{{marks}}</h2>

<h2>Student Subject:{{subject}}</h2>

</body>

* **urls.py**

path('tt2/',views.TemplateCBV2.as\_view())

**Model Related View classes to perform CRUD operations**

View

TemplateView

To perform CRUD operations, predefined View classes are:

ListView -->To select all records(R)

DetailView -->To get details of a particular record(R)

CreateView -->To insert a record(C)

DeleteView -->To delete a record(D)

UpdateView -->To update record(U)

**1).ListView:**

We can use ListView class to list out all records from the database(Model).

It is alternative way to:ModelClassname.objects.all()

Default template file name:modelname\_list.html

Default context object name:modelname\_list

**Ex:ListView class by using CBV's:**

D:\Django\_20MAR\_7PM>django-admin startproject cbvproject2

D:\Django\_20MAR\_7PM>cd cbvproject2

D:\Django\_20MAR\_7PM\cbvproject2>py manage.py startapp testapp

**-->**Add app in settings.py

* **models.py**

class Book(models.Model):

title = models.CharField(max\_length=30)

author = models.CharField(max\_length=30)

pages = models.IntegerField()

price = models.FloatField()

**-->**makemigrations and migrate

* **admin.py**

from testapp.models import Book

class BookAdmin(admin.ModelAdmin):

list\_display = ['title','author','pages','price']

admin.site.register(Book,BookAdmin)

* **views.py**

from django.views.generic import ListView

from testapp.models import Book

class BookListView(ListView):

model = Book

#default template file: book\_list.html

#default context object name: book\_list

* **urls.py**

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

* **book\_list.html**

<body>

<div class="container">

<h1>All Books Information</h1>

{% for book in book\_list %}

<ul>

<li>Title:<strong>{{book.title}}</strong></li>

<li>Author:<strong>{{book.author}}</strong></li>

<li>Pages:<strong>{{book.pages}}</strong></li>

<li>Price:<strong>{{book.price}}</strong></li>

</ul>

<hr>

{% endfor %}

</div>

</body>