**Code file: ch 90**

**Video file:100**

**Method Decorator in Class Based View Django 5**

Iin this we use the ch87 file

Simple authentication

In this we use decorator instead of mixings

If we want to use the decorator there are certain methods

1. We use in urls

Remove mixins from views.py

Now we add decorators

from django.contrib.auth.decorators import login\_required

urlpatterns = [

    path('dashboard/', login\_required(CustomerDashboardView.as\_view()), name='customer\_dashboard'),

1. we see second method in seller we also use in views.py

from django.utils.decorators import method\_decorator

from django.contrib.auth.decorators import login\_required

class SellerDashboardView(View):

    @method\_decorator(login\_required)

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

        return super().dispatch(request, \*args, \*\*kwargs)

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

        return render(request, 'seller/dashboard.html')

1. another method :

@method\_decorator(login\_required)

class SellerDashboardView(View):

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

        return render(request, 'seller/dashboard.html')

another method if we use permission decorators :

views.py

@method\_decorator(permission\_required('product.can\_add'), name='dispatch')

class SellerDashboardView( View):

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

        return render(request, 'seller/dashboard.html')

in urls.py:

1. If we want to use multiple decorator at a time :

@method\_decorator([login\_required, permission\_required('product.can\_add')], name='dispatch')

class SellerDashboardView( View):

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

        return render(request, 'seller/dashboard.html')