# Django detail view

## Django detail view¶

- Django detail view shows the single object information to the user.
- It fetches the object from the database using primary key or an equivalent.
- Let's write a view for contact detail view page.

### contact model¶

my_app/models.py

```
from django.db import models

class Contact(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.EmailField(max_length=255)
    phone = models.CharField(max_length=10, null=True)

    class Meta:
        db_table = "contact"
```

### function based detail view¶

- It's useful when we have a functionality with simple requirements.
- Look at the code below.
-

```
from django.http import Http404
from django.shortcuts import render
from my_app.models import Contact

def fbv_detail_view(request, pk):
    try:
        obj = Contact.objects.get(id=pk)
    except Contact.DoesNotExist:
        raise Http404
    context = {"object": obj}
    return render(request, 'detail_template.html', context)
```

- Detail view function takes two parameters `request` and `pk`
- `request` is an instance of `HttpRequest` class, which contains the request info
- `pk` is a url keyword argument used query the database to get the object.

## class based detail view¶

- Django provides generic class `DetailView`
- We can inherit the `DetailView` and provide the required attibutes and/or methods to make it work.
- Only use if it fits your use case.
- Let's look at the code below.

```
from django.views.generic.detail import DetailView
from my_app.models import Contact

class CBVListView(DetailView):
    template_name = 'detail_template.html'
    pk_url_kwarg = 'pk'
    queryset = Contact.objects.all()
```

- In the above code `pk_url_kwarg` is used as a primary key in the query.
- Either we need to provider `queryset` attribute or `get_queryset()` or `get_object()` methods.
- There are other attributes and methods available to override based on the functionality.
- We can find more info at ccbv
- It sends the context `{"object": <Model object>}` to the template.

## detail_template.html¶

- Let's look at the code for the template

```
<div>
    <p>Name: {{ object.first_name }}</p>
    <p>Email: {{ object.email }}</p>
</div>
```

- Template receives the context and reders the template.

## configure urls¶

- open `my_app/urls.py` and add below code to it.

```
from django.urls import path
from my_app import views

urlpatterns = [
    # ...
```

```
    path('fbv_detail_view/<int:pk>', views.fbv_detail_view,
name='fbv_detail_view'),
    path('cbv_detail_view/<int:pk>', views.CBVDetailView.as_view(),
name='cbv_detail_view'),
    # ...
]
```

- Open url http://127.0.0.1:8000/my_app/fbv_detail_view/1 to see the response for function based detail view
- Open url http://127.0.0.1:8000/my_app/cbv_detail_view/1 to see the response for class based detail view
- Replace the `1` with any `id` in the `contact` table.
- It throws 404 status code if id does not exist in the table.

## References¶

- https://ccbv.co.uk/projects/Django/4.1/django.views.generic.detail/DetailView/
- https://docs.djangoproject.com/en/dev/ref/class-based-views/generic-display/#detailview