List View – Function based Views Django

Last Updated : 27 Aug, 2021

List View refers to a view (logic) to list all or particular instances of a table from the database in a particular order. It is used to display multiple types of data on a single page or view, for example, products on an eCommerce page. Django provides extra-ordinary support for List Views but let's check how it is done manually through a function-based view. This article revolves around list View which involves concepts such as Django Forms, Django Models.
For List View, we need a project with some models and multiple instances which will be displayed.

Django List View – Function Based Views

Illustration of **How to create and use List view** using an Example. Consider a project named geeksforgeeks having an app named geeks.

> *Refer to the following articles to check how to create a project and an app in Django.*

- *How to Create a Basic Project using MVT in Django?*
- *How to Create an App in Django ?*

After you have a project and an app, let's create a model of which we will be creating instances through our view. In geeks/models.py,

Python3

```
# import the standard Django Model
# from built-in library
from django.db import models

# declare a new model with a name "GeeksModel"
class GeeksModel(models.Model):

    # fields of the model
    title = models.CharField(max_length = 200)
    description = models.TextField()

    # renames the instances of the model
    # with their title name
    def __str__(self):
        return self.title
```

After creating this model, we need to run two commands in order to create Database for the same.

```
Python manage.py makemigrations
Python manage.py migrate
```

Now let's create some instances of this model using shell, run form bash,

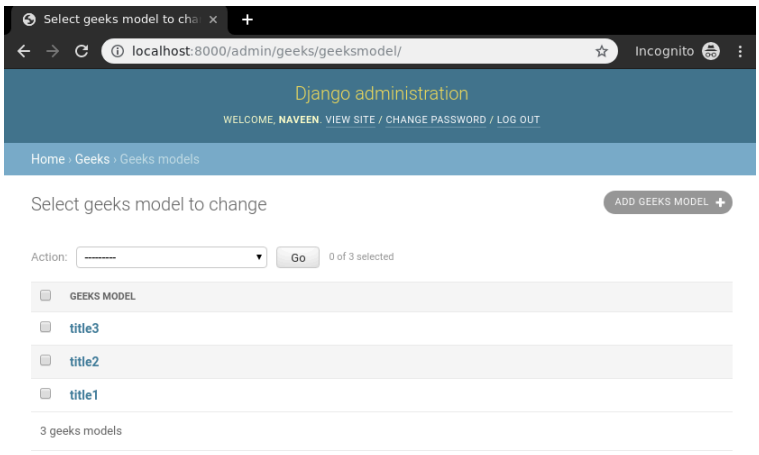
```
Python manage.py shell
```

Enter following commands

```
>>> from geeks.models import GeeksModel
>>> GeeksModel.objects.create(
                      title="title1",
                      description="description1").save()
>>> GeeksModel.objects.create(
                      title="title2",
                      description="description2").save()
>>> GeeksModel.objects.create(
```

```
title="title2",
description="description2").save()
```

Now we have everything ready for back end. Verify that instances have been created from http://localhost:8000/admin/geeks/geeksmodel/



Let's create a view and template for the same. In geeks/views.py,

Python3

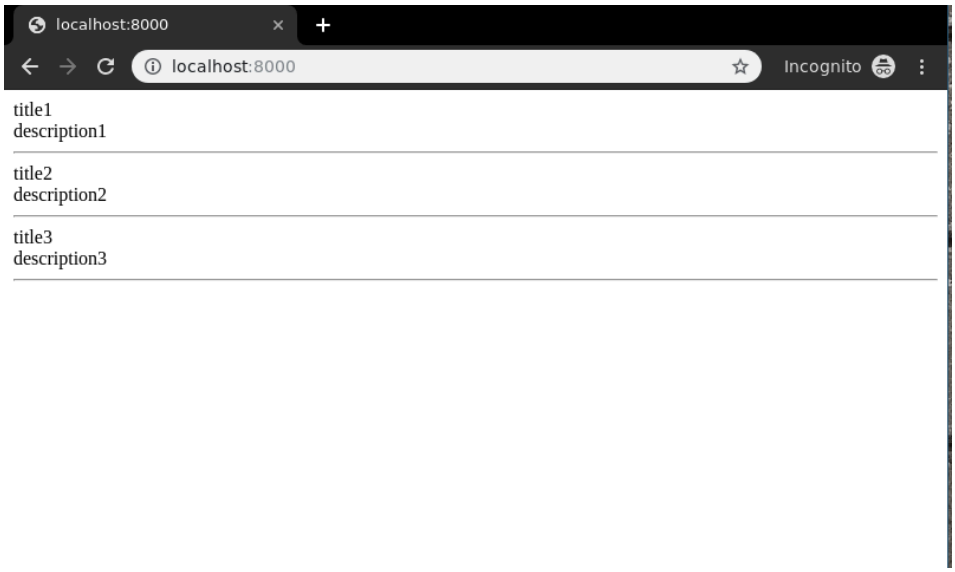
```
from django.shortcuts import render

# relative import of forms
from .models import GeeksModel


def list_view(request):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    context["dataset"] = GeeksModel.objects.all()

    return render(request, "list_view.html", context)
```

Create a template in templates/list_view.html,

html

```
<div class="main">

    {% for data in dataset %}.

    {{ data.title }}<br/>
    {{ data.description }}<br/>
    <hr/>

    {% endfor %}

</div>
```

Let's check what is there on http://localhost:8000/



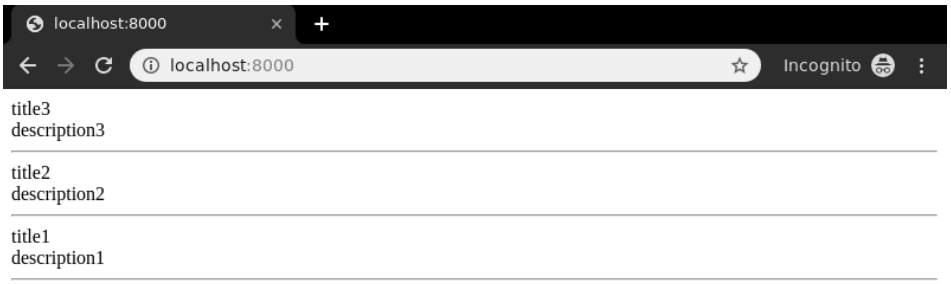
Bingo..!! list view is working fine. One can also display filtered items or order them in different orders based on various features. Let's order the items in reverse manner.
In geeks/views.py,

Python3

```
from django.shortcuts import render

# relative import of models
from .models import GeeksModel


def list_view(request):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    context["dataset"] = GeeksModel.objects.all().order_by("-id")

    return render(request, "list_view.html", context)
```

order_by to arrange instances in different orders

Now visit http://localhost:8000/



filter to show selective instances

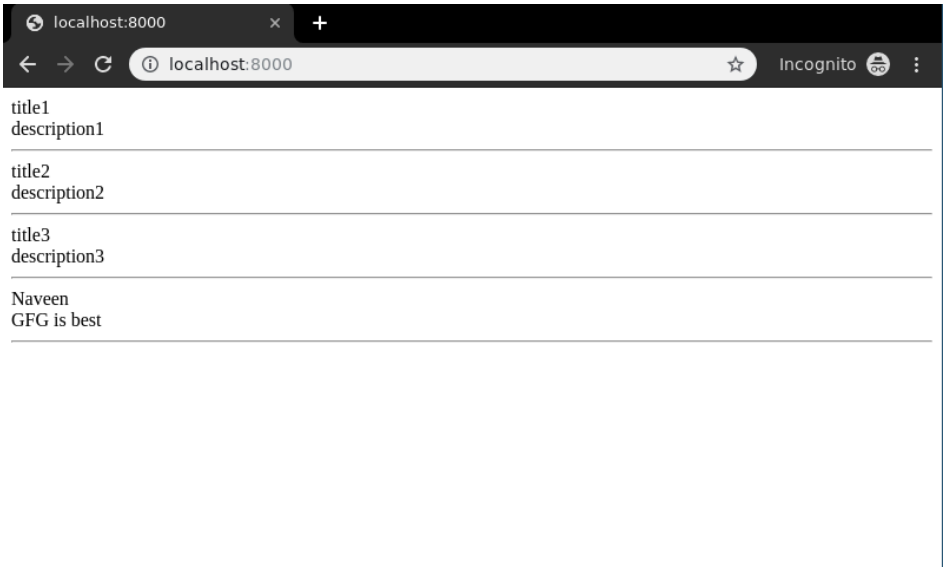
Let's create a different instance to show how filter works. Run

```
Python manage.py shell
```

Now, create another instance,

```
from geeks.models import GeeksModel
GeeksModel.objects.create(title = "Naveen", description = "GFG is
Best").save()
```

Now visit http://localhost:8000/



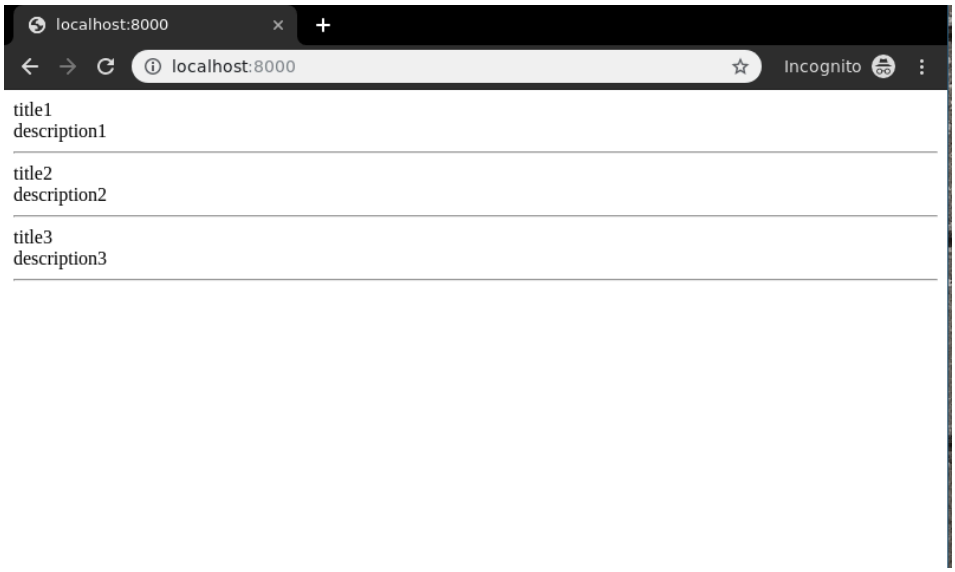
Let's filter this data to those containing word "title" in their title.
In **geeks/views.py**,

Python3

```
from django.shortcuts import render

# relative import of forms
from .models import GeeksModel


def list_view(request):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    context["dataset"] = GeeksModel.objects.all().filter(
        title__icontains = "title"
```

```
    )

    return render(request, "list_view.html", context)
```

Now visit http://localhost:8000/ again,



Are you ready to elevate your web development skills from foundational knowledge to advanced expertise? Explore our Mastering Django Framework - Beginner to Advanced Course on GeeksforGeeks, designed for aspiring developers and experienced programmers. This comprehensive course covers everything you need to know about Django, from the basics to advanced features. Gain practical experience through **hands-on projects** and real-world applications, mastering essential Django principles and techniques. Whether you're just starting or looking to refine your skills, this course will empower you to build sophisticated web applications efficiently. Ready to enhance your web development journey? Enroll now and unlock your potential with Django!

Previous Article Next Article

Create View - Function based Views Django

Detail View - Function based Views Django

Similar Reads

Create View - Function based Views Django

Create View refers to a view (logic) to create an instance of a table in the database. It is just like taking an input from a user and storing it in a specified...

3 min read

Update View - Function based Views Django

Update View refers to a view (logic) to update a particular instance of a table from the database with some extra details. It is used to update entries in the...

4 min read

Detail View - Function based Views Django

Detail View refers to a view (logic) to display a particular instance of a table from the database with all the necessary details. It is used to display multiple types o...

3 min read

Delete View - Function based Views Django

Delete View refers to a view (logic) to delete a particular instance of a table from the database. It is used to delete entries in the database for example, deleting a...

3 min read

Class Based vs Function Based Views - Which One is Better to Use in Django?

Django...We all know the popularity of this python framework all over the world. This framework has made life easier for developers. It has become easier for...

7 min read

Function based Views - Django Rest Framework

Django REST Framework allows us to work with regular Django views. It facilitates processing the HTTP requests and providing appropriate HTTP...

13 min read

Django Function Based Views

Django is a Python-based web framework which allows you to quickly create web application without all of the installation or dependency problems that you...

7 min read

Createview - Class Based Views Django

Create View refers to a view (logic) to create an instance of a table in the database. We have already discussed basics of Create View in Create View –...

3 min read

ListView - Class Based Views Django

List View refers to a view (logic) to display multiple instances of a table in the database. We have already discussed the basics of List View in List View –...

4 min read

UpdateView - Class Based Views Django

UpdateView refers to a view (logic) to update a particular instance of a table from the database with some extra details. It is used to update entries in the databas...

3 min read

Article Tags : Python Django-views Python Django

Practice Tags : python