DeleteView – Class Based Views Django

Last Updated : 02 Nov, 2022

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 an article at geeksforgeeks. We have already discussed basics of Delete View in Delete View – Function based Views Django. Class-based views provide an alternative way to implement views as Python objects instead of functions. They do not replace function-based views, but have certain differences and advantages when compared to function-based views:

- Organization of code related to specific HTTP methods (GET, POST, etc.) can be addressed by separate methods instead of conditional branching.
- Object oriented techniques such as mixins (multiple inheritance) can be used to factor code into reusable components.

Class based views are simpler and efficient to manage than function-based views. A function based view with tons of lines of code can be converted into a class based views with few lines only. This is where Object Oriented Programming comes into impact.

Django DeleteView – Class Based Views

Illustration of **How to create and use DeleteView** 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
```

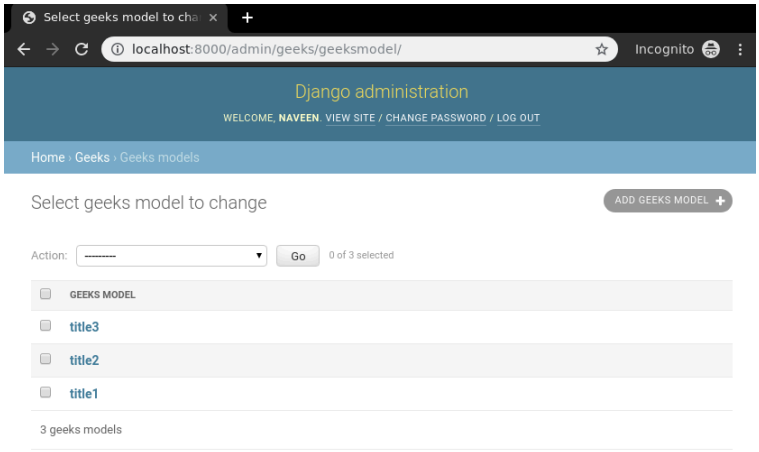
Now let's create some instances of this model using shell, run form bash,

```
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/



Class Based Views automatically setup everything from A to Z. One just needs to specify which model to create DeleteView for, then Class based DeleteView will automatically try to find a template in app_name/modelname_confirm_delete.html. In our case it is geeks/templates/geeks/geeksmodel_confirm_delete.html. Let's create our class based view. In geeks/views.py,

Python3

```
# import generic UpdateView
from django.views.generic.edit import DeleteView

# Relative import of GeeksModel
from .models import GeeksModel

class GeeksDeleteView(DeleteView):
    # specify the model you want to use
    model = GeeksModel

    # can specify success url
    # url to redirect after successfully
    # deleting object
    success_url ="/"

    template_name = "geeks/geeksmodel_confirm_delete.html"
```

Now create a url path to map the view. In geeks/urls.py,

Python3

```
from django.urls import path

# importing views from views..py
from .views import GeeksDeleteView
urlpatterns = [
    # <pk> is identification for id field,
    # slug can also be used
    path('<pk>/delete/', GeeksDeleteView.as_view()),
]
```

Create a template in templates/geeks/geeksmodel_confirm_delete.html,

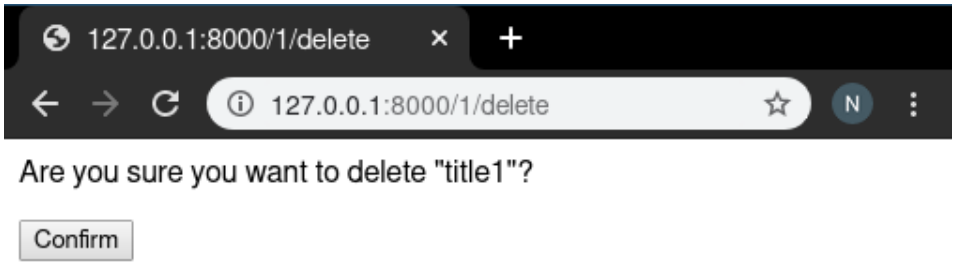
html

```
<form method="post">{% csrf_token %}
```

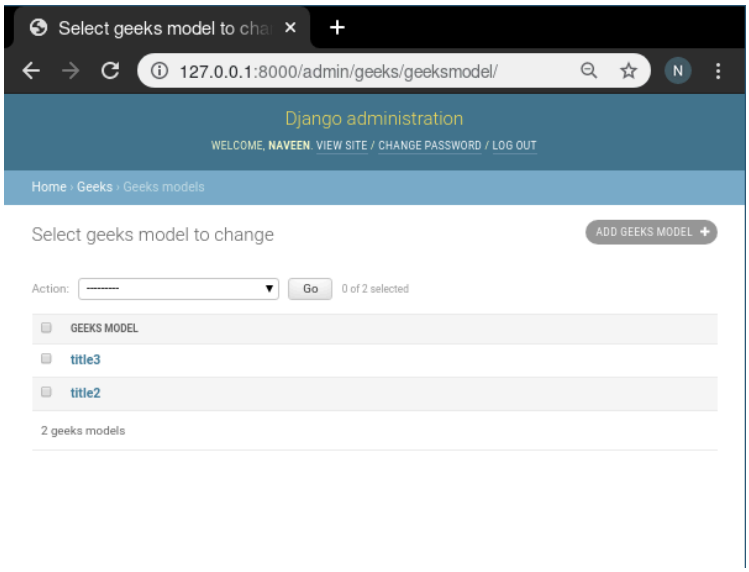
```
<p>Are you sure you want to delete "{{ object }}"?</p>


    <input type="submit" value="Confirm">
</form>
```

Let's check what is there on http://localhost:8000/1/delete



Tap confirm and object will redirect to success_url defined in the view. Let's check if **title1** is deleted from database.



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

Customize Django Admin Interface

Next Article

Similar Reads

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

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

DetailView - Class Based Views Django

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

3 min read

FormView - Class Based Views Django

FormView refers to a view (logic) to display and verify a Django Form. For example, a form to register users at Geeksforgeeks. Class-based views provide ...

3 min read

Class based views - Django Rest Framework

Class-based views help in composing reusable bits of behavior. Django REST Framework provides several pre-built views that allow us to reuse common...

12 min read

Django Class Based Views

Django is a Python-based web framework that allows you to quickly create web applications. It has a built-in admin interface which makes it easy to work with it...

10 min read

How to Use permission_required Decorators with Django Class-Based Views

In Django, permissions are used to control access to views and resources. When working with function-based views (FBVs), decorators like @permission_require...

4 min read

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

Article Tags : Python Django-views Python Django

Practice Tags : python