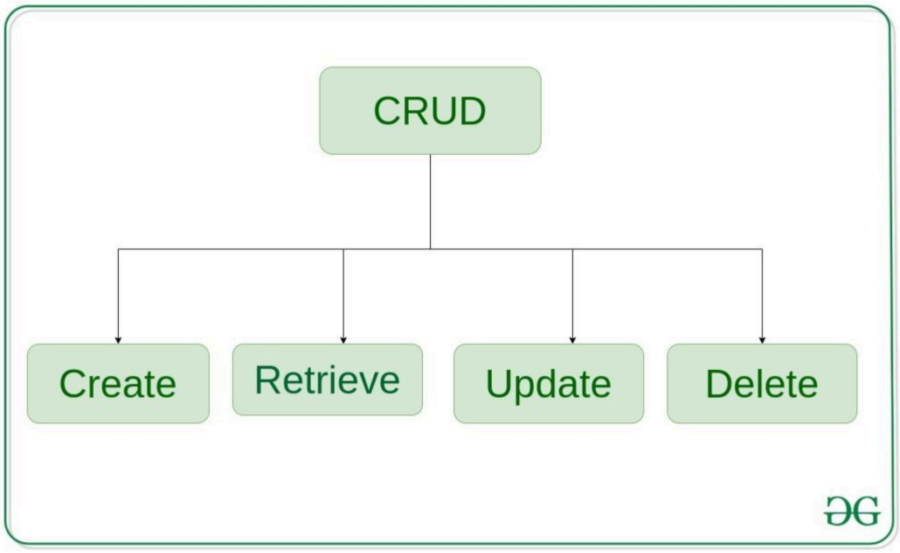
Django Function Based Views

Last Updated : 23 Sep, 2024

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 normally will find with other frameworks. Django is based on MVT (Model View Template) architecture and revolves around CRUD (Create, Retrieve, Update, Delete) operations. CRUD can be best explained as an approach to building a Django web application. In general CRUD means performing Create, Retrieve, Update and Delete operations on a table in a database. Let's discuss what actually CRUD means,



Create – create or add new entries in a table in the database.
Retrieve – read, retrieve, search, or view existing entries as a list(List View) or retrieve a particular entry in detail (Detail View)
Update – update or edit existing entries in a table in the database
Delete – delete, deactivate, or remove existing entries in a table in the database

Django Function Based Views – CRUD Operations

Illustration of **How to create and use CRUD 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

```python
# 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 we will create a Django ModelForm for this model. Refer this article for more on modelform – Django ModelForm – Create form from Models. create a file forms.py in geeks folder,

Python

```python
from django import forms
from .models import GeeksModel


# creating a form
class GeeksForm(forms.ModelForm):

    # create meta class
    class Meta:
        # specify model to be used
        model = GeeksModel

        # specify fields to be used
        fields = [
            "title",
            "description",
        ]
```

Create View

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 table.
In geeks/views.py,

Python

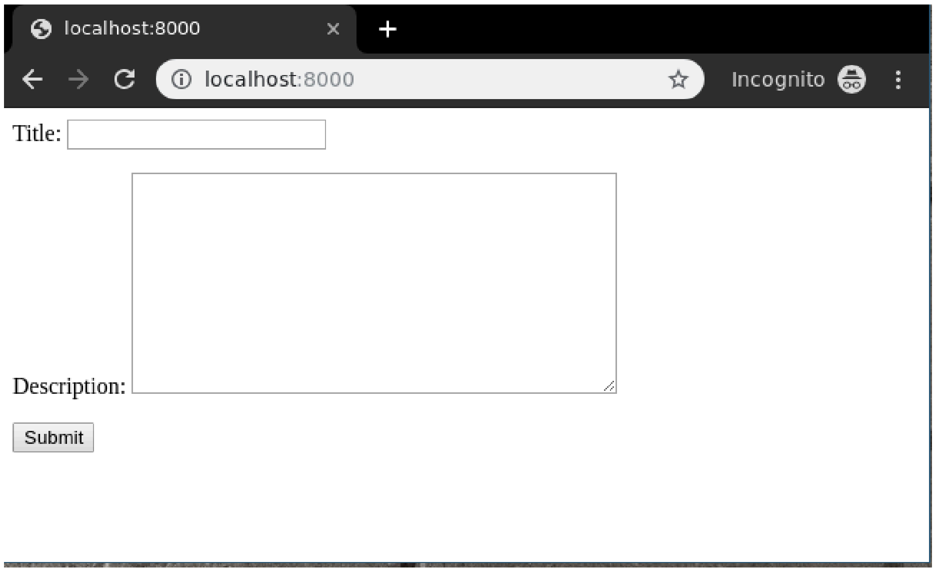
```python
from django.shortcuts import render

# relative import of forms
from .models import GeeksModel
from .forms import GeeksForm


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

    # add the dictionary during initialization
    form = GeeksForm(request.POST or None)
    if form.is_valid():
        form.save()

    context['form'] = form
    return render(request, "create_view.html", context)
```

Create a template in templates/create_view.html,

html

```html
<form method="POST" enctype="multipart/form-data">

    <!-- Security token -->
    {% csrf_token %}

    <!-- Using the formset -->
    {{ form.as_p }}

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

Now visit http://localhost:8000/



To check complete implementation of Function based Create View, visit Create View – Function based Views Django.

Retrieve View

Retrieve view is basically divided into two types of views Detail View and List View.

List View

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. In geeks/views.py.

Python3

```python
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)
```

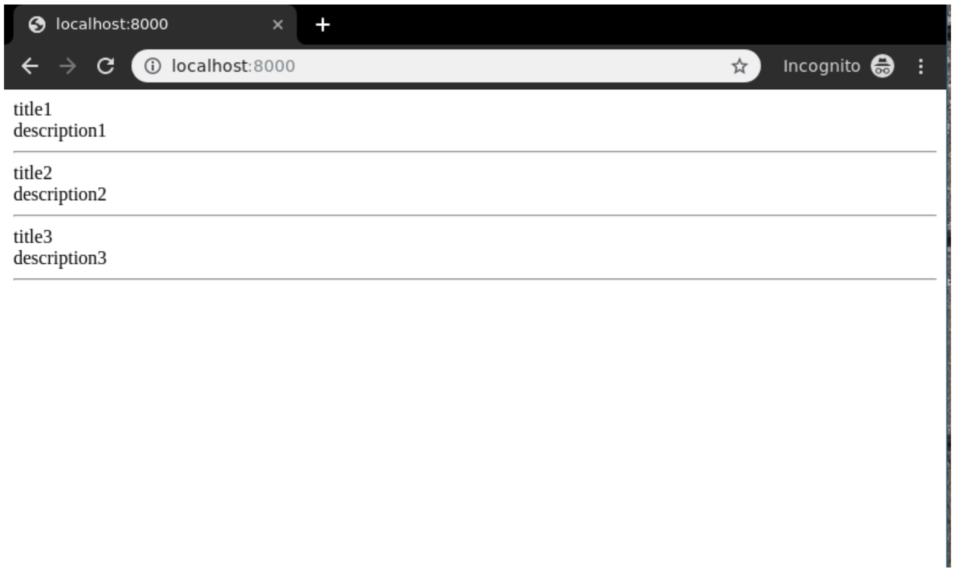
Create a template in templates/list_view.html,

html

```html
<div class="main">

```

```
    {% for data in dataset %}.

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

    {% endfor %}

</div>
```

Now visit http://localhost:8000/



```
title1
description1
title2
description2
title3
description3
```

To check complete implementation of Function based List View, visit List View – Function based Views Django

Detail View

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 of data on a single page or view, for example, profile of a user.

In geeks/views.py,

Python3

```
from django.urls import path

# importing views from views..py
from .views import detail_view

urlpatterns = [
    path('<id>', detail_view ),
]
```

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

Python3

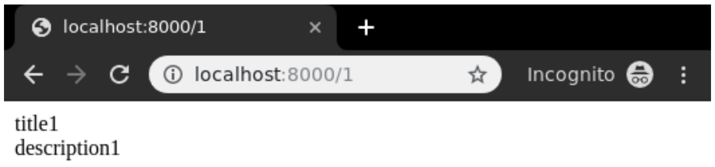
```python
from django.shortcuts import render

# relative import of forms
from .models import GeeksModel

# pass id attribute from urls
def detail_view(request, id):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    context["data"] = GeeksModel.objects.get(id = id)

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

Create a template in templates/Detail_view.html,

html

```html
<div class="main">

    <!-- Specify fields to be displayed -->
    {{ data.title }}<br/>
    {{ data.description }}<br/>

</div>
```

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



title1
description1

To check complete implementation of Function based Detail View, visit [Detail View – Function based Views Django]

Update View

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 database for example, updating an article at geeksforgeeks.
In geeks/views.py,

Python3

```python
from django.shortcuts import (get_object_or_404,
                              render,
                              HttpResponseRedirect)

# relative import of forms
from .models import GeeksModel
from .forms import GeeksForm

# after updating it will redirect to detail_View
def detail_view(request, id):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    context["data"] = GeeksModel.objects.get(id = id)

    return render(request, "detail_view.html", context)

# update view for details
def update_view(request, id):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # fetch the object related to passed id
    obj = get_object_or_404(GeeksModel, id = id)

    # pass the object as instance in form
    form = GeeksForm(request.POST or None, instance = obj)

    # save the data from the form and
    # redirect to detail_view
    if form.is_valid():
        form.save()
        return HttpResponseRedirect("/"+id)

    # add form dictionary to context
    context["form"] = form

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

Now create following templates in templates folder,

In geeks/templates/update_view.html,

html

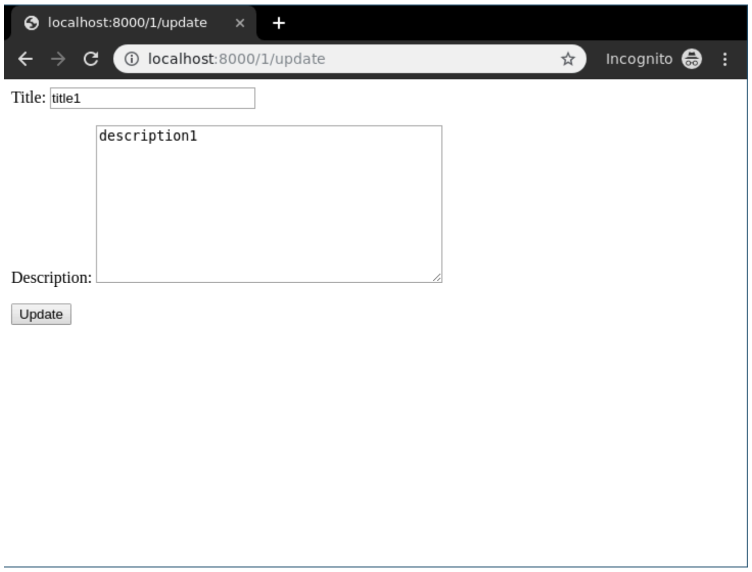
```html
<div class="main">
    <!-- Create a Form -->
    <form method="POST">
        <!-- Security token by Django -->
        {% csrf_token %}

        <!-- form as paragraph -->
        {{ form.as_p }}

        <input type="submit" value="Update">
    </form>

</div>
```

In geeks/templates/detail_view.html,

html

```
<div class="main">
    <!-- Display attributes of instance -->
    {{ data.title }} <br/>
    {{ data.description }}
</div>
```

Let’s check if everything is working, visit: http://localhost:8000/1/update.



Title: title1

Description: description1

To check complete implementation of Function based update View, visit [Update View – Function based Views Django]

Delete View

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.
In geeks/views.py

Python3

```python
from django.shortcuts import (get_object_or_404,
                              render,
                              HttpResponseRedirect)

from .models import GeeksModel


# delete view for details
def delete_view(request, id):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # fetch the object related to passed id
    obj = get_object_or_404(GeeksModel, id = id)


    if request.method =="POST":
        # delete object
        obj.delete()
        # after deleting redirect to
        # home page
        return HttpResponseRedirect("/")

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

Now a url mapping to this view with a regular expression of id,

In geeks/urls.py

Python3

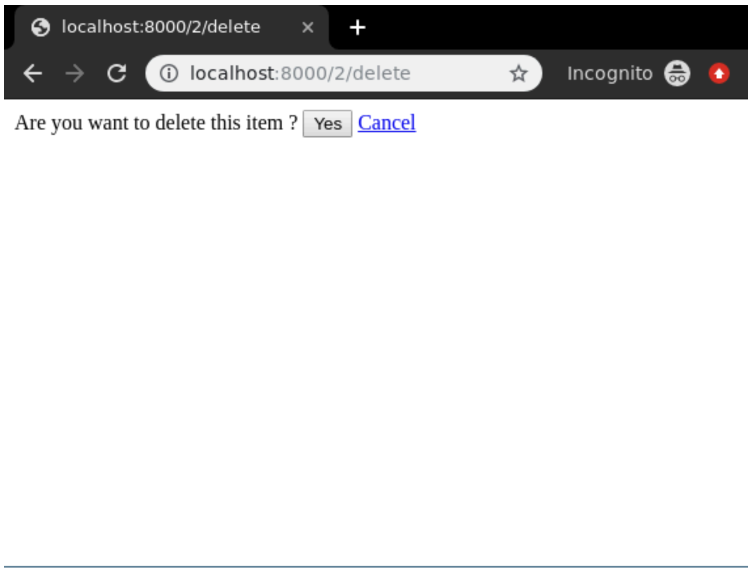
```python
from django.urls import path

# importing views from views..py
from .views import delete_view
urlpatterns = [
    path('<id>/delete', delete_view ),
]
```

Template for delete view includes a simple form confirming whether user wants to delete the instance or not. In geeks/templates/delete_view.html,

html

```html
<div class="main">
    <!-- Create a Form -->
    <form method="POST">
        <!-- Security token by Django -->
        {% csrf_token %}
        Are you want to delete this item ?
        <input type="submit" value="Yes" />
        <a href="/">Cancel </a>
    </form>
</div>
```

Everything ready, now let's check if it is working or not, visit http://localhost:8000/2/delete



Are you want to delete this item ? Yes Cancel

To check complete implementation of Function based Delete View, visit [Delete View – Function based Views Django]

N Nave... 45

Next Article

Create View - Function based Views Django

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

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

List View - Function based Views Django

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 ...

3 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

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 : Django Python Django-views Python Django

Practice Tags : python