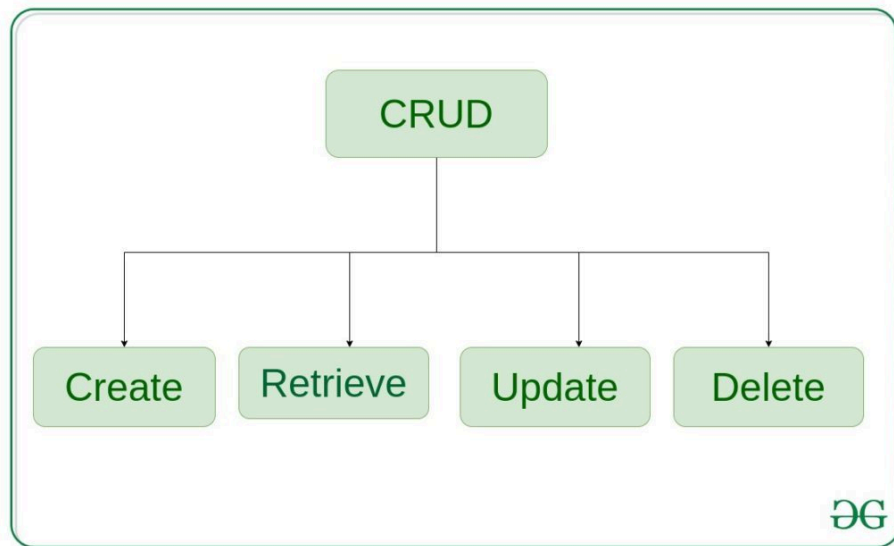




## Django Function Based Views

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



```
1 # import the standard Django Model
2 # from built-in library
3 from django.db import models
4
5 # declare a new model with a name "GeeksModel"
6 class GeeksModel(models.Model):
7
8     # fields of the model
9     title = models.CharField(max_length = 200)
10    description = models.TextField()
11
12    # renames the instances of the model
13    # with their title name
14    def __str__(self):
15        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



```
1 from django import forms
2 from .models import GeeksModel
3
4
5 # creating a form
6 class GeeksForm(forms.ModelForm):
7
8     # create meta class
9     class Meta:
10         # specify model to be used
11         model = GeeksModel
12
13         # specify fields to be used
14         fields = [
15             "title",
16             "description",
17         ]
```

## 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



```
1 from django.shortcuts import render
2
3 # relative import of forms
4 from .models import GeeksModel
5 from .forms import GeeksForm
6
7
8 def create_view(request):
```

```
9      # dictionary for initial data with
10     # field names as keys
11     context = {}
12
13     # add the dictionary during initialization
14     form = GeeksForm(request.POST or None)
15     if form.is_valid():
16         form.save()
17
18     context['form'] = form
19     return render(request, "create_view.html", context)
```

Create a template in templates/create\_view.html,

html



```
1  <form method="POST" enctype="multipart/form-data">
2
3      <!-- Security token -->
4      {% csrf_token %}
5
6      <!-- Using the formset -->
7      {{ form.as_p }}
8
9      <input type="submit" value="Submit">
10 </form>
```

Now visit <http://localhost:8000/>

The screenshot shows a web browser window with the address bar set to [localhost:8000](http://localhost:8000/). The page displays a simple form with two input fields and a submit button. The first field is labeled 'Title:' and is a single-line text input. The second field is labeled 'Description:' and is a multi-line text area. Below the text area is a button labeled 'Submit'.

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

```
1  from django.shortcuts import render
2
3  # relative import of forms
4  from .models import GeeksModel
5
6
7  def list_view(request):
8      # dictionary for initial data with
9      # field names as keys
10     context = {}
11
12     # add the dictionary during initialization
13     context["dataset"] = GeeksModel.objects.all()
14
15     return render(request, "list_view.html", context)
```

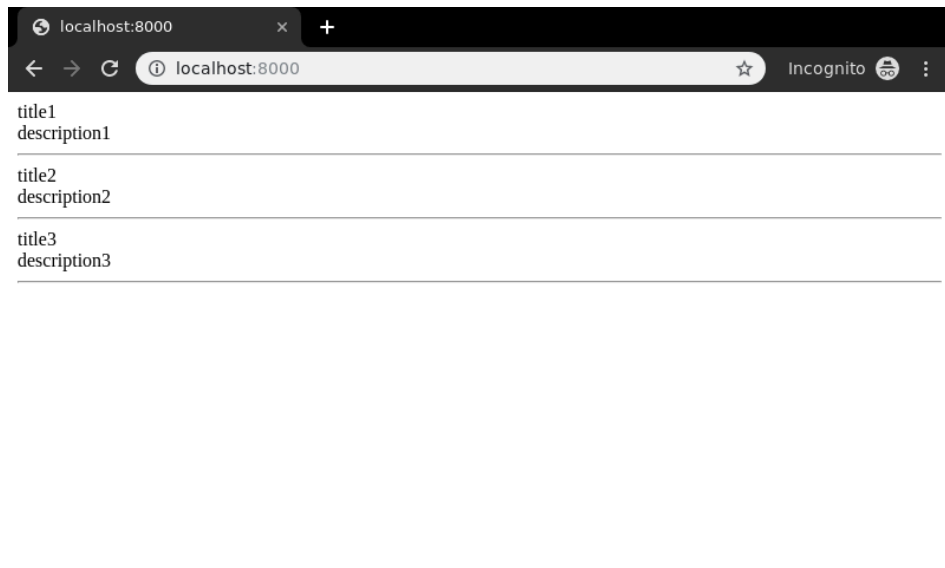
Create a template in `templates/list_view.html`,

#### html

```
1  <div class="main">
2
```

```
3      {% for data in dataset %}.
4
5      {{ data.title }}<br/>
6      {{ data.description }}<br/>
7      <hr/>
8
9      {% endfor %}
10
11 </div>
```

Now visit <http://localhost:8000/>



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

```
1  from django.urls import path
2
3  # importing views from views..py
4  from .views import detail_view
5
6  urlpatterns = [
```

```
7     path('<id>', detail_view ),
8 ]
```

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

### Python3

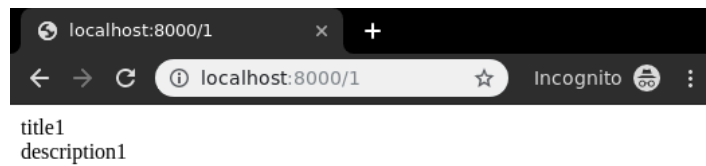
```
1 from django.shortcuts import render
2
3 # relative import of forms
4 from .models import GeeksModel
5
6 # pass id attribute from urls
7 def detail_view(request, id):
8     # dictionary for initial data with
9     # field names as keys
10    context = {}
11
12    # add the dictionary during initialization
13    context["data"] = GeeksModel.objects.get(id = id)
14
15    return render(request, "detail_view.html", context)
```

Create a template in `templates/Detail_view.html`,

### html

```
1 <div class="main">
2
3     <!-- Specify fields to be displayed -->
4     {{ data.title }}<br/>
5     {{ data.description }}<br/>
6
7 </div>
```

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



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

```
1  from django.shortcuts import (get_object_or_404,
2                                  render,
3                                  HttpResponseRedirect)
4
5  # relative import of forms
6  from .models import GeeksModel
7  from .forms import GeeksForm
8
9  # after updating it will redirect to detail_View
10 def detail_view(request, id):
11     # dictionary for initial data with
12     # field names as keys
13     context = {}
14
15     # add the dictionary during initialization
```



```

16     context["data"] = GeeksModel.objects.get(id = id)
17
18     return render(request, "detail_view.html", context)
19
20 # update view for details
21 def update_view(request, id):
22     # dictionary for initial data with
23     # field names as keys
24     context = {}
25
26     # fetch the object related to passed id
27     obj = get_object_or_404(GeeksModel, id = id)
28
29     # pass the object as instance in form
30     form = GeeksForm(request.POST or None, instance = obj)
31
32     # save the data from the form and
33     # redirect to detail_view
34     if form.is_valid():
35         form.save()
36         return HttpResponseRedirect("/") + id
37
38     # add form dictionary to context
39     context["form"] = form
40
41     return render(request, "update_view.html", context)

```

Now create following templates in templates folder,  
In geeks/templates/update\_view.html,

html

```

1 <div class="main">
2     <!-- Create a Form -->
3     <form method="POST">
4         <!-- Security token by Django -->
5         {% csrf_token %}
6
7         <!-- form as paragraph -->
8         {{ form.as_p }}
9
10    <input type="submit" value="Update">

```

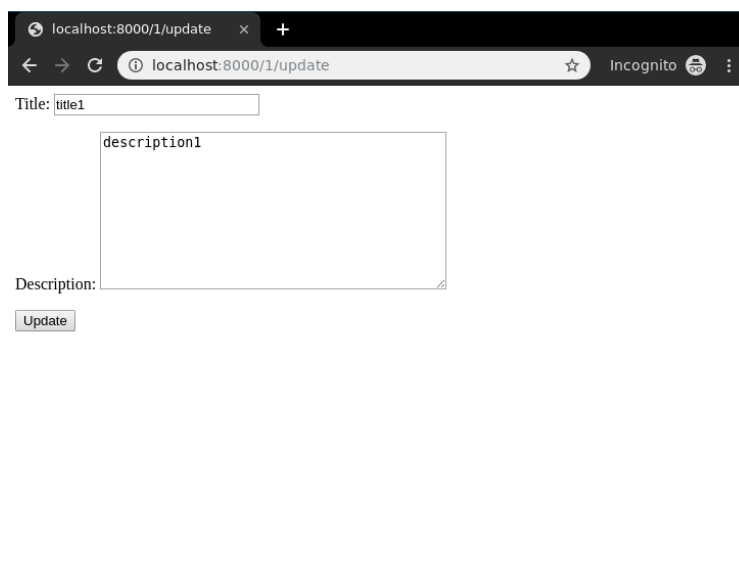
```
11     </form>
12
13 </div>
```

In `geeks/templates/detail_view.html`,

html

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

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



The screenshot shows a web browser window with the address bar displaying `localhost:8000/1/update`. The page content includes a form with the following elements:

- A label "Title:" followed by a text input field containing the value "title1".
- A label "Description:" followed by a text area containing the value "description1".
- An "Update" button located below the description text area.



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



```
1 from django.shortcuts import (get_object_or_404,
2                                render,
3                                HttpResponseRedirect)
4
5 from .models import GeeksModel
6
7
8 # delete view for details
9 def delete_view(request, id):
10     # dictionary for initial data with
11     # field names as keys
12     context = {}
13
14     # fetch the object related to passed id
15     obj = get_object_or_404(GeeksModel, id = id)
16
17
18     if request.method == "POST":
19         # delete object
20         obj.delete()
21         # after deleting redirect to
22         # home page
23         return HttpResponseRedirect("/")
24
25     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



```
1 from django.urls import path
2
3 # importing views from views..py
4 from .views import delete_view
5 urlpatterns = [
6     path('<id>/delete', delete_view ),
```

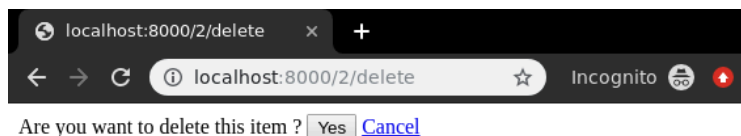
7 ]

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

```
1 <div class="main">
2     <!-- Create a Form -->
3     <form method="POST">
4         <!-- Security token by Django -->
5         {% csrf_token %}
6         Are you want to delete this item ?
7         <input type="submit" value="Yes" />
8         <a href="/">Cancel </a>
9     </form>
10 </div>
```

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



---

To check complete implementation of Function based Delete View, visit [Delete View – Function based Views Django](https://www.geeksforgeeks.org/django-crud-create-retrieve-update-delete-function-based-views/)

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!

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](#)



Corporate & Communications Address:-  
A-143, 9th Floor, Sovereign Corporate  
Tower, Sector- 136, Noida, Uttar Pradesh  
(201305) | Registered Address:- K 061,  
Tower K, Gulshan Vivante Apartment,  
Sector 137, Noida, Gautam Buddh  
Nagar, Uttar Pradesh, 201305



## Company

[About Us](#)  
[Legal](#)  
[In Media](#)  
[Contact Us](#)  
[Advertise with us](#)  
[GFG Corporate Solution](#)  
[Placement Training Program](#)  
[GeeksforGeeks Community](#)

## DSA

[Data Structures](#)  
[Algorithms](#)  
[DSA for Beginners](#)  
[Basic DSA Problems](#)  
[DSA Roadmap](#)  
[Top 100 DSA Interview Problems](#)  
[DSA Roadmap by Sandeep Jain](#)  
[All Cheat Sheets](#)

## Languages

[Python](#)  
[Java](#)  
[C++](#)  
[PHP](#)  
[GoLang](#)  
[SQL](#)  
[R Language](#)  
[Android Tutorial](#)  
[Tutorials Archive](#)

## Data Science & ML

[Data Science With Python](#)  
[Data Science For Beginner](#)  
[Machine Learning](#)  
[ML Maths](#)  
[Data Visualisation](#)  
[Pandas](#)  
[NumPy](#)  
[NLP](#)  
[Deep Learning](#)

## Web Technologies

HTML  
CSS  
JavaScript  
TypeScript  
ReactJS  
NextJS  
Bootstrap  
Web Design

## Computer Science

Operating Systems  
Computer Network  
Database Management System  
Software Engineering  
Digital Logic Design  
Engineering Maths  
Software Development  
Software Testing

## System Design

High Level Design  
Low Level Design  
UML Diagrams  
Interview Guide  
Design Patterns  
OOAD  
System Design Bootcamp  
Interview Questions

## School Subjects

Mathematics  
Physics  
Chemistry  
Biology  
Social Science  
English Grammar  
Commerce  
World GK

## Python Tutorial

Python Programming Examples  
Python Projects  
Python Tkinter  
Web Scraping  
OpenCV Tutorial  
Python Interview Question  
Django

## DevOps

Git  
Linux  
AWS  
Docker  
Kubernetes  
Azure  
GCP  
DevOps Roadmap

## Interview Preparation

Competitive Programming  
Top DS or Algo for CP  
Company-Wise Recruitment Process  
Company-Wise Preparation  
Aptitude Preparation  
Puzzles

## GeeksforGeeks Videos

DSA  
Python  
Java  
C++  
Web Development  
Data Science  
CS Subjects

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved