



[Django](#) [Views](#) [Model](#) [Template](#) [Forms](#) [Jinja](#) [Python SQLite](#) [Flask](#) [Json](#) [Postman](#) [Interview Ques](#)

Createview – Class Based Views Django

Last Updated : 11 May, 2022

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 – 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 Create View – Class Based Views

Illustration of **How to create and use create 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`

Class Based Views automatically setup everything from A to Z. One just needs to specify which model to create View for and the fields. Then Class based `CreateView` will automatically try to find a template in `app_name/modelname_form.html`. In our case it is `geeks/templates/geeks/geeksmodel_form.html`. Let's create our class based view. In `geeks/views.py`,

Python3

```
from django.views.generic.edit import CreateView
from .models import GeeksModel

class GeeksCreate(CreateView):
```

```
# specify the model for create view
model = GeeksModel

# specify the fields to be displayed

fields = ['title', 'description']
```

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 GeeksCreate
urlpatterns = [
    path('', GeeksCreate.as_view() ),
]
```

Create a template in `templates/geeks/geeksmodel_form.html`,

html

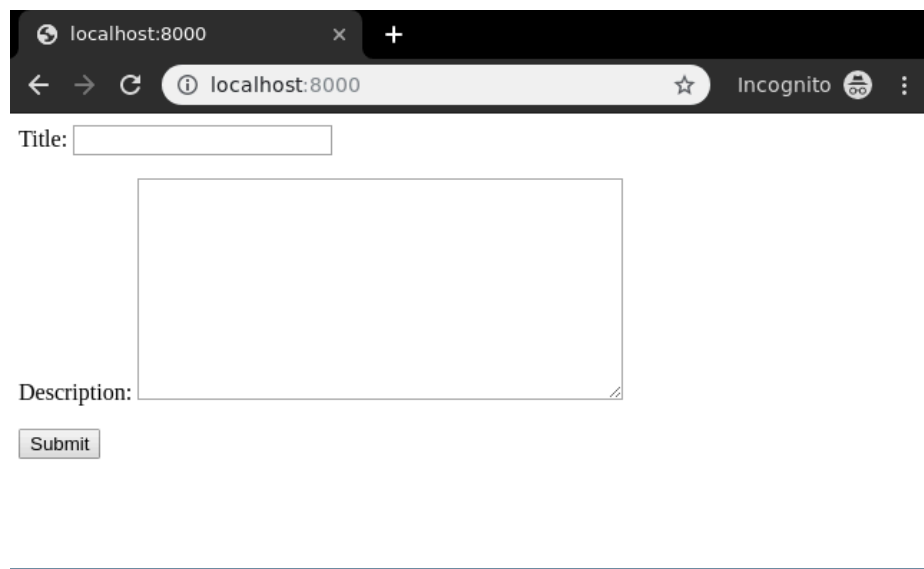
```
<form method="POST" enctype="multipart/form-data">

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

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

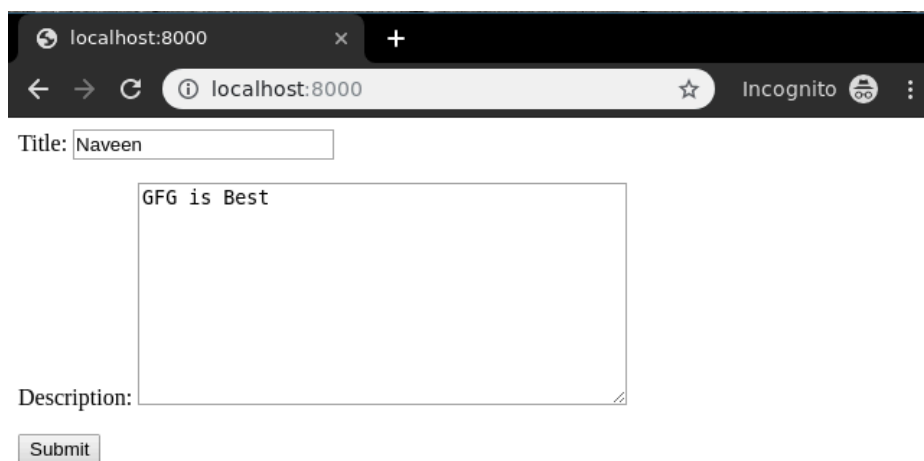
    <input type="submit" value="Submit">
</form>
```

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



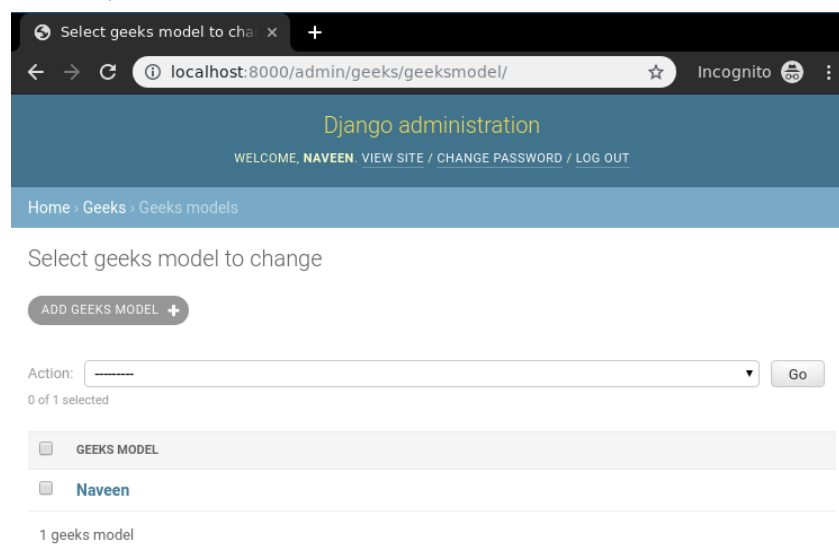
The screenshot shows a web browser window with the address bar set to `localhost:8000`. The page contains a form with a "Title:" label followed by a text input field, a "Description:" label followed by a larger text area, and a "Submit" button at the bottom.

Now let's try to enter data in this form,



The screenshot shows the same web browser window, but now the form is filled with data. The "Title:" input field contains the text "Naveen", and the "Description:" text area contains the text "GFG is Best". The "Submit" button remains at the bottom.

Bingo.! Create view is working and we can verify it using the instance created through the admin panel.



The screenshot shows the Django administration interface. The browser address bar is `localhost:8000/admin/geeks/geeksmodel/`. The page title is "Django administration" and it says "WELCOME, NAVEEN". The breadcrumb trail is "Home > Geeks > Geeks models". The main heading is "Select geeks model to change". There is an "ADD GEEKS MODEL +" button. Below it is an "Action:" dropdown menu and a "Go" button. It says "0 of 1 selected". A table lists the models:

<input type="checkbox"/>	GEEKS MODEL
<input checked="" type="checkbox"/>	Naveen

At the bottom, it says "1 geeks model".

This way one can create view for a model in Django.

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...



19

Previous Article

Django Class Based Views

Next Article

ListView - Class 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

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

DeleteView - Class 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

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...

Article Tags :

PythonDjango-viewsPython 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