Render HTML Forms (GET & POST) in Django

Last Updated : 16 Aug, 2021

Django is often called “Batteries Included Framework” because it has a default setting for everything and has features that can help anyone develop a website rapidly. Talking about forms, In HTML, a form is a collection of elements inside <form>...</form> that allow a visitor to do things like entering text, select options, manipulate objects or controls, and so on, and then send that information back to the server. Basically, it is a collection of data for processing it for any purpose including saving it in the database or fetching data from the database. Django supports all types of HTML forms and rendering data from them to a view for processing using various logical operations.

> *To know more about HTML forms, visit HTML | form Tag.*

Django Views Model Template Forms Jinja Python SQLite Flask Json Postman Interview Ques

Models. One can create forms in Django and use them to fetch data from the user in a convenient manner.

To begin with forms, one needs to be familiar with GET and POST requests in forms.

- **GET:** GET, by contrast, bundles the submitted data into a string, and uses this to compose a URL. The URL contains the address where the data must be sent, as well as the data keys and values. You can see this in action if you do a search in the Django documentation, which will produce a URL of the form https://docs.djangoproject.com/search/?q=forms&release=1.
- **POST:** Any request that could be used to change the state of the system – for example, a request that makes changes in the database – should use POST.

Render HTML Forms in Django Explanation

Illustration of **Django Forms** 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 ?*

Let's create a simple HTML form to show how can you input the data from a user and use it in your view. Enter following code in geeks > templates > home.html

HTML

```html
<form action = "" method = "get">
    <label for="your_name">Your name: </label>
    <input id="your_name" type="text" name="your_name">
    <input type="submit" value="OK">
</form>
```

Now to render it in our view we need to modify urls.py for geeks app. Enter the following code in geeksforgeeks > urls.py

Python3

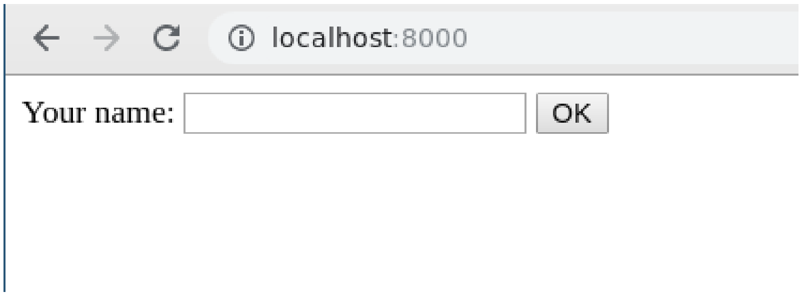
```python
from django.urls import path

# importing views from views..py
from .views import geeks_view

urlpatterns = [
    path('', home_view ),
]
```

Now, let’s move to our home_view and start checking how are we going to get the data. Entire data from an HTML form in Django is transferred as a JSON object called a request. Let’s create a view first and then we will try all methods to fetch data from the form.

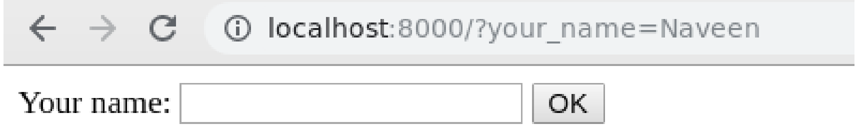
Python3

```python
from django.shortcuts import render

# Create your views here.
def home_view(request):

    # logic of view will be implemented here
    return render(request, "home.html")
```

As we have everything set up let us run Python manage.py run server and check if the form is there on the home page.



By default every form ever written in HTML makes a GET request to the back end of an application, a GET request normally works using queries in the URL. Let’s demonstrate it using the above form, Fill up the form using your name, and let’s check what happens.


localhost:8000/?your_name=Naveen

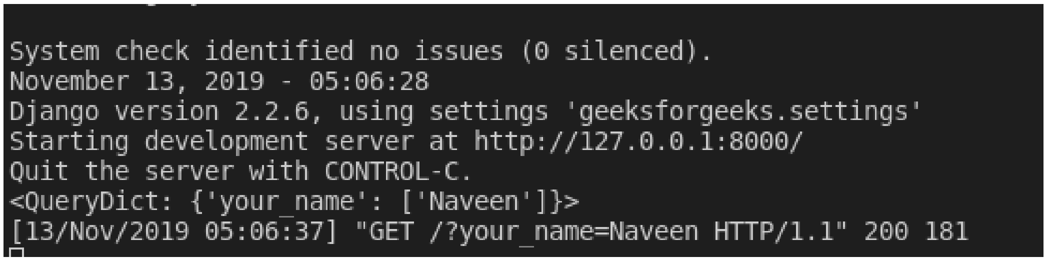
The above URL is appended with a **name** attribute of the input tag and the name entered in the form. This is how the GET request works whatever be the number of inputs they would be appended to the URL to send the data to the back end of an application. Let's check how to finally get this data in our view so that logic could be applied based on input.

In views.py

Python3

```python
from django.shortcuts import render

# Create your views here.
def home_view(request):
    print(request.GET)
    return render(request, "home.html")
```

Now when we fill the form we can see the output in the terminal as below:



```
System check identified no issues (0 silenced).
November 13, 2019 - 05:06:28
Django version 2.2.6, using settings 'geeksforgeeks.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
<QueryDict: {'your_name': ['Naveen']}>
[13/Nov/2019 05:06:37] "GET /?your_name=Naveen HTTP/1.1" 200 181
```

request.GET returns a query dictionary that one can access like any other python dictionary and finally use its data for applying some logic.

Similarly, if the method of transmission is POST, you can use request.POST as query dictionary for rendering the data from the form into views.

In ***home.html***

HTML

```html
<form action = "" method = "POST">
    {% csrf_token %}
    <label for="your_name">Your name: </label>
    <input id="your_name" type="text" name="your_name">
    <input type="submit" value="OK">
</form>
```

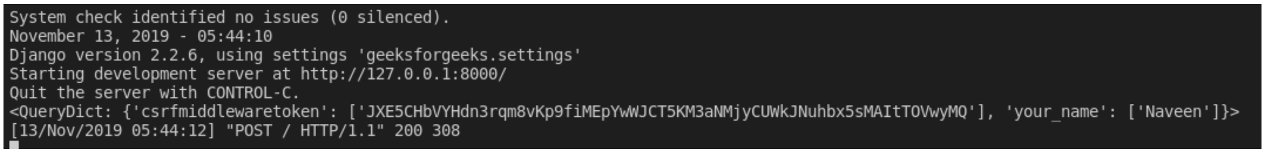
Note that whenever we create a form request, Django requires you to add {% csrf_token %} in form for security purposes

Now, in views.py let's check what request.POST has got.

Python3

```
from django.shortcuts import render

# Create your views here.
def home_view(request):
    print(request.POST)
    return render(request, "home.html")
```

Now when we submit the form it shows the data as below.



```
System check identified no issues (0 silenced).
November 13, 2019 - 05:44:10
Django version 2.2.6, using settings 'geeksforgeeks.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
<QueryDict: {'csrfmiddlewaretoken': ['JXE5CHbVYHdn3rqm8vKp9fiMEpYwWJCT5KM3aNMjyCUWkJNuhbx5sMAItTOVwyMQ'], 'your_name': ['Naveen']}>
[13/Nov/2019 05:44:12] "POST / HTTP/1.1" 200 308
```

This way one can use this data for querying into the database or for processing using some logical operation and pass using the context dictionary to the template.

N Nave... 37

Previous Article

{{ form.as_ul }} - Render Django Forms as list

Next Article

Django form field custom widgets

Similar Reads

Styling Django Forms with django-crispy-forms

Django by default doesn't provide any Django form styling method due to which it takes a lot of effort and precious time to beautifully style a form. django-crispy...

1 min read

{{ form.as_p }} - Render Django Forms as paragraph

Django forms are an advanced set of HTML forms that can be created using python and support all features of HTML forms in a pythonic way. Rendering...

2 min read

{{ form.as_table }} - Render Django Forms as table

Django forms are an advanced set of HTML forms that can be created using python and support all features of HTML forms in a pythonic way. Rendering...

2 min read

{{ form.as_ul }} - Render Django Forms as list

Django forms are an advanced set of HTML forms that can be created using python and support all features of HTML forms in a pythonic way. Rendering...

2 min read

Render a HTML Template as Response - Django Views

A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response. This article revolves around how to render...

3 min read

How to customize Django forms using Django Widget Tweaks ?

Django forms are a great feature to create usable forms with just few lines of code. But Django doesn't easily let us edit the form for good designs. Here, we...

3 min read

Render Model in Django Admin Interface

Rendering model in admin refers to adding the model to the admin interface so that data can be manipulated easily using admin interface. Django's ORM...

3 min read

Render Django Form Fields Manually

Django form fields have several built-in methods to ease the work of the developer but sometimes one needs to implement things manually for...

5 min read

How to Add HTML Attributes to Input Fields in Django Forms?

Django provides an easy-to-use system for creating and managing forms through its forms.Form and forms.ModelForm classes. These classes allow us to create...

4 min read

How to create a form using Django Forms ?

Django forms are an advanced set of HTML forms that can be created using python and support all features of HTML forms in a pythonic way. This post...

3 min read

Article Tags : Python Django-forms Python Django

Practice Tags : python

GeeksforGeeks
Sanchhaya Education Private Limited

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

Languages
Python
Java
C++
PHP
GoLang

GFG Corporate Solution

Placement Training Program

GeeksforGeeks Community

SQL

R Language

Android Tutorial

Tutorials Archive

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

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

Python Tutorial

Python Programming Examples

Python Projects

Python Tkinter

Web Scraping

OpenCV Tutorial

Python Interview Question

Django

Computer Science

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Software Development

Software Testing

DevOps

Git

Linux

AWS

Docker

Kubernetes

Azure

GCP

DevOps Roadmap

System Design

High Level Design

Low Level Design

UML Diagrams

Interview Guide

Design Patterns

OOAD

System Design Bootcamp

Interview Questions

Inteview Preparation

Competitive Programming

Top DS or Algo for CP

Company-Wise Recruitment Process

Company-Wise Preparation

Aptitude Preparation

Puzzles

School Subjects

Mathematics

Physics

GeeksforGeeks Videos

DSA

Python

Chemistry

Biology

Social Science

English Grammar

Commerce

World GK

Java

C++

Web Development

Data Science

CS Subjects