



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

Django – How to add multiple submit button in single form?

Last Updated : 12 Dec, 2021

Prerequisites:

- [Python](#)
- [Pip](#)
- [Django](#)

When we submit the data using the HTML form, it sends the data to the server at a particular URL which is defined in the action attribute. To perform different actions with a single HTML form, we just need to add multiple submit buttons in the form. For example, If you go through the code of the newsletter app, you will find subscribe and unsubscribe buttons in a single HTML form but both perform different actions.

In this tutorial, we are going to make a newsletter app using Django. We will add subscribe and unsubscribe buttons in a single HTML form. Moreover, We will add or remove the email address of the user from our database according to the user clicks on the subscribe or unsubscribe button.

To create the simple newsletter Django app, follow the below steps.

Create a new Django project

To start a new Django project, your local computer should be satisfied with the prerequisites are given above. Users can use the below command to start a new project.

```
django-admin startproject newsletter
```

Next, Go to the project directory.

```
cd newsletter
```

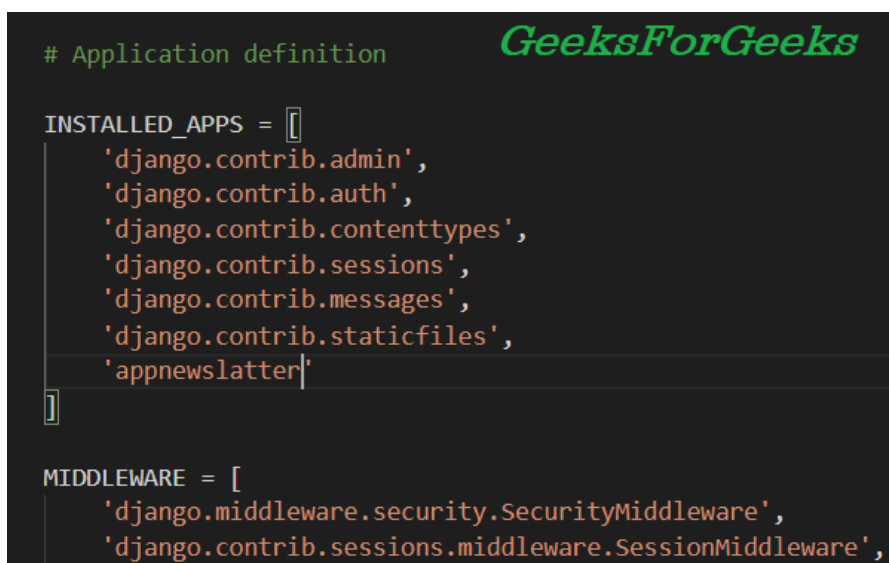
Start a newsletter app

To start a new app inside the newsletter project, run the below command inside the project directory.

```
django-admin startapp appnewsletter
```

Now, add “appnewsletter” inside the installed_apps section in settings.py of the newsletter project.

path: newsletter/settings.py



```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'appnewsletter'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
```

Next, Edit the `urls.py` inside the newsletter folder and add the below code.

Path: newsletter/urls.py

Filename: urls.py

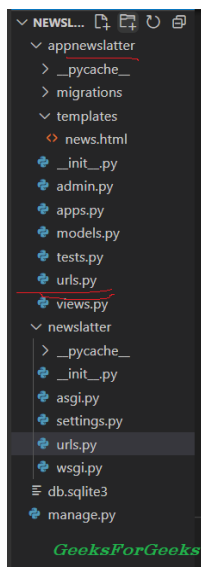
Python3

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),

    #including URLS of appnewsletter app
    path("", include('appnewsletter.urls')),
]
```

Create a new `urls.py` file inside the “appnewsletter” folder. Now, your project directory should look like the below image.



Edit the `urls.py` inside the `appnewsletter` folder. We are adding the URL for the home page.

Path: `appnewslattere/urls.py`

Filename: `urls.py`

Python3

```
from django.urls import path
from . import views

urlpatterns = [

    # URL to open home page
    path("", views.home, name='home'),
]
```

Create a “templates” folder inside the `appnewsletter` folder to store the HTML templates.

path: `appnewsletter/templates`

Create a news.html file inside the templates folder to add a form for our newsletter app.

Path: appnewsletter/templates/news.html

Add the below code inside the news.html file which includes a single form with 2 submit buttons. Every submits button has a unique name. In the views.py file, We will identify from which button users have sent the post request using the name of the button.

Filename: news.html

HTML

```
<!DOCTYPE html>
<html>

  <head>
    <title>NewsLatter</title>
  </head>

  <body>

    <!--showing success message-->
    {% if messages %}
      <ul class="messages">
        {% for message in messages %}
          <li{% if message.tags %} class="{ { message.tags } }"{% endif %}>
            {{ message }}</li>
          {% endfor %}
        </ul>
      {% endif %}

    <!--Form with multiple submit buttons-->
    <form action="" method="POST">
      <label for="email">Enter your email:</label>
      <input type="email" id="email" name="email" />
      <br> <br>
      <button type="submit" name="subscribe">Subscribe</button>
      <button type="submit" name="unsubscribe">Unsubscribe</button>
    </form>

  </body>
</html>
```

Now, we need to create a table in the database to store the email of the users. We will edit the models.py file.

path: appnewlatter/models.py

Filename: models.py

Python3

```
from django.db import models

# creating database model to store email
class newslatteremail(models.Model):
    userEmail = models.EmailField(max_length=254)

    def __str__(self):
        return self.userEmail
```

Register the created model inside the admin.py file.

Path: appnewslatter/admin.py

Filename: admin.py

Python3

```
from django.contrib import admin
from .models import newslatteremail

# registering the model
admin.site.register(newslatteremail)
```

After registering the model, we need to migrate it. Users need to run the below 2 commands one by one.

```
python manage.py makemigrations
```

```
python manage.py migrate
```

Now, we will edit the views.py file and add the code to handle requests from subscribe and unsubscribe buttons. Here, we check that from which submit button we get the post request with the help of the name attribute of the button.

Syntax:

```
if 'name_of_button' in request.POST:  
    # perform some action
```

Example:

```
if 'subscribe' in request.POST:  
    # add the user email in database  
if 'unsubscribe' in request.POST:  
    # remove the user email from database
```

Copy/paste the below code inside the views.py file.

Path: appnewsletter/views.py

Filename: views.py

Python3

```
from django.shortcuts import render  
from django.contrib import messages  
from .models import newsletteremail  
  
def home(request):  
  
    # if post request comes from the subscribe button  
    # then saving user email in our database  
    if 'subscribe' in request.POST:  
        email = newsletteremail()  
        email.userEmail = request.POST.get("email")  
        email.save()  
        messages.info(  
            request, 'You have successfully subscribed to our newsletter.')  
  
    # if post request comes from the unsubscribe button  
    # then deleting the user email from our database
```

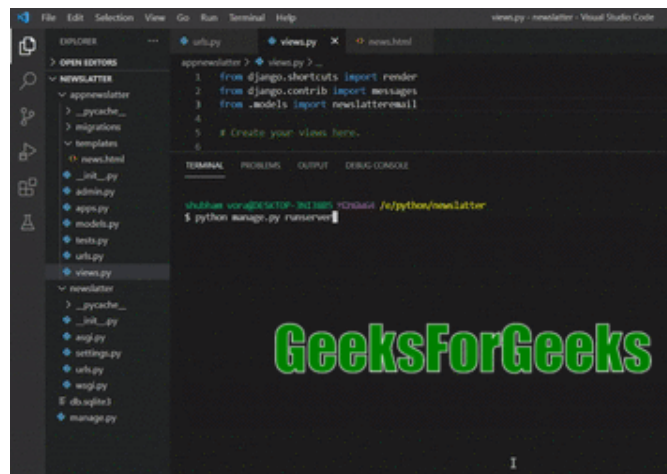
```
if 'unsubscribe' in request.POST:
    newsletteremail.objects.get(
        userEmail=request.POST.get("email")).delete()
    messages.info(request, 'sorry to see you!!!')

return render(request, 'news.html')
```

Finally, we have created a simple newsletter app and learned about how we can add multiple submit buttons in a single HTML form. Let's run the project and see the output. Users can run the Django app using the below command.

```
Python manage.py runserver
```

Output:



shub...



2

Previous Article

How to submit a form without using
submit button using PHP ?

Next Article

Similar Reads

Handle Multiple Forms on a Single Page in Django

One common scenario in web development is handling multiple forms on a single page. This might be required when a user needs to fill out various types of...

6 min read

How to Add Multiple Arguments to Custom Template Filter in a Django...

In Django, template filters are useful for modifying data before displaying it in templates. While built-in filters cover common cases, custom logic is sometimes...

4 min read

How to Add Multiple Objects to ManyToMany Relationship at Once in Django?

In Django, the ManyToManyField allows for the creation of relationships where multiple records in one table can be associated with multiple records in another...

4 min read

submit() element method - Selenium Python

Selenium's Python Module is built to perform automated testing with Python. Selenium Python bindings provides a simple API to write functional/acceptance...

2 min read

Django QuerySet.values() for Single Object

In Django, QuerySet.Values() method helps us get specific fields from the database as dictionaries, making our queries simpler. This can be incredibly usef...

3 min read

Adding Tags Using Django-Taggit in Django Project

Django-Taggit is a Django application which is used to add tags to blogs, articles etc. It makes very easy for us to make adding the tags functionality to our django...

2 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

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

Integrating Django with Reactjs using Django REST Framework

In this article, we will learn the process of communicating between the Django Backend and React js frontend using the Django REST Framework. For the sake...

15+ min read

How to Fix Django "ImportError: Cannot Import Name 'six' from..."

After Django 3.0, django.utils.six was removed. The "ImportError: Cannot Import Name 'six' from 'django.utils'" error occurs because of the dependency issue. Thi...

3 min read

Article Tags :[Python](#)[Django-forms](#)[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](#)

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

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

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

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

| | |
|-----------------|-----------------|
| Chemistry | Java |
| Biology | C++ |
| Social Science | Web Development |
| English Grammar | Data Science |
| Commerce | CS Subjects |
| World GK | |

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved