ToDo webapp using Django

Last Updated : 25 Jul, 2024

Django is a high-level Python Web framework-based web framework that allows rapid development and clean, pragmatic design. today we will create a todo app to understand the basics of Django. In this web app, one can create notes like Google Keep or Evernote.

Basic setup

Create a virtual environment, and install the packages:

```
pip install django-crispy-forms
pip install Django
```

Step 1: Start a project with the following command

```
django-admin startproject todo_site
```

Step 2: Change the directory to todo_site.

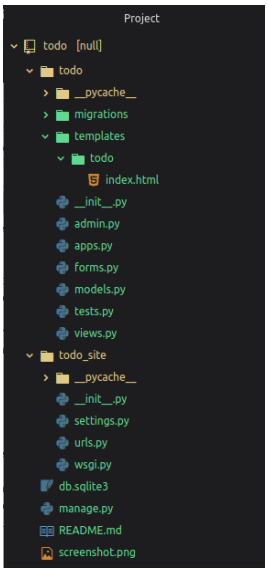
```
cd todo_site
```

Let's create an app now

Step 3: Create an app with the following command.

```
python manage.py startapp todo
```

The directory structure should look like this:



Step 4: Now add the todo app in your todo_site in **settings.py**

```
# Application definition

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

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Step 5: Edit urls.py file in todo_site

Python3

```
from django.contrib import admin
from django.urls import path
from todo import views

urlpatterns = [

######################home_page#################################
#############
```

```
    path('', views.index, name="todo"),
    ####################give id no. item_id name or
item_id=i.id ############
    # pass item_id as primary key to remove that the todo
with given id
    path('del/<str:item_id>', views.remove, name="del"),

############################################################
############
    path('admin/', admin.site.urls),
]
```

Step 6: Edit models.py in todo

Python3

```
from django.db import models
from django.utils import timezone


class Todo(models.Model):
    title = models.CharField(max_length=100)
    details = models.TextField()
    date = models.DateTimeField(default=timezone.now)

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

Step 7: Edit views.py in todo

Python3

```
from django.shortcuts import render, redirect
from django.contrib import messages

# import todo form and models

from .forms import TodoForm
from .models import Todo

#############################################
```

```
11
12  def index(request):
13
14      item_list = Todo.objects.order_by("-date")
15      if request.method == "POST":
16          form = TodoForm(request.POST)
17          if form.is_valid():
18              form.save()
19              return redirect('todo')
20      form = TodoForm()
21
22      page = {
23          "forms": form,
24          "list": item_list,
25          "title": "TODO LIST",
26      }
27      return render(request, 'todo/index.html', page)
28
29
30  ### function to remove item, it receive todo item_id as
    primary key from url ##
31  def remove(request, item_id):
32      item = Todo.objects.get(id=item_id)
33      item.delete()
34      messages.info(request, "item removed !!!")
35      return redirect('todo')
```

Step 8: Now create a forms.py in todo

Python3

```
1  from django import forms
2  from .models import Todo
3
4
5  class TodoForm(forms.ModelForm):
6      class Meta:
7          model = Todo
8          fields = "__all__"
```

Step 9: Register models to admin

admin.py

```
from django.contrib import admin
from .models import Todo
# Register your models here.
admin.site.register(Todo)

```

Step 10: Create templates/todo/index.html

HTML

```
<!DOCTYPE html>
<html lang="en" dir="ltr">

<head>

  <meta charset="utf-8">
  <title>{{title}}</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
  <!--style-->
  <style>
  .card {

   box-shadow: 0 4px 8px 0 rgba(0,0,0,0.5),
               0 6px 20px 0 rgba(0,0,0,0.39);
   background: lightpink;
   margin-bottom : 5%;
   border-radius: 25px;
   padding : 2%;
   overflow: auto;
   resize: both;
   text-overflow: ellipsis;
  }
```

```
    .card:hover{
      background: lightblue;
    }

    .submit_form{

      text-align: center;
      padding: 3%;
      background: pink;
      border-radius: 25px;
      box-shadow: 0 4px 8px 0 rgba(0,0,0,0.4),
                  0 6px 20px 0 rgba(0,0,0,0.36);
    }
    </style>

</head>

<body  class="container-fluid">

    {% if messages %}
    {% for message in messages %}
    <div class="alert alert-info">
      <strong>{{message}}</strong>
    </div>
    {% endfor %}
    {% endif %}

    <center class="row">
      <h1><i>__TODO LIST__</i></h1>
      <hr />
    </center>

    <div class="row">

      <div class="col-md-8">

        {% for i in list %}
        <div class="card">
          <center><b>{{i.title}}</b></center>
          <hr/>
          {{i.date}}
          <hr/>
          {{i.details}}
```

```
            <br />
            <br />
            <form action="/del/{{i.id}}" method="POST" style="
    padding-right: 4%; padding-bottom: 3%;">
              {% csrf_token %}
              <button value="remove" type="submit"  class="btn
    btn-primary" style="float: right;"><span class="glyphicon
    glyphicon-trash"></span>   remove</button>
            </form>
          </div>
          {% endfor%}
        </div>
        <div class="col-md-1"> </div>
        <div class="col-md-3" >
          <div  class="submit_form">
          <form  method="POST">
            {% csrf_token %}
            {{forms}}
            <center>
            <input type="submit" class="btn btn-default"
    value="submit" />
          </center>
          </form>
        </div>
      </div>
    </div>
    </body>

    </html>
```

Step 11: Migrations Files to the Database

```
python manage.py makemigrations
python manage.py migrate
```

Step 12: Start the server by typing the following command in the terminal

```
python manage.py runserver
```

Output:

Open the web browser and enter http://127.0.0.1:8000/ as the URL.

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!

itsvin...

44

Previous Article

Django Sign Up and login with confirmation Email | Python

Next Article

How to Send Email with Django

Similar Reads

Twitter Sentiment Analysis WebApp Using Flask

This is a web app made using Python and Flask Framework. It has a registration system and a dashboard. Users can enter keywords to retrieve live Twitter text...

15+ min read

Daily Latest News webapp Using PyWebio in Python

In this article, we are going to create a web app to get gaily News Using PyWebio As we all download an app for receiving daily news but as a python lovers we tr...

5 min read

Create a Simple Sentiment Analysis WebApp using Streamlit

In this article, we will see how we can create a simple Sentiment Analysis webApp using with the help of Streamlit Module in Python. Required Modules:...

4 min read

Multiplication Quiz Webapp using HTML CSS and JavaScript

In this article, we will see how to create a multiplication quiz web app using HTML, CSS, and JavaScript. Description of Web App: In this web app, random...

3 min read

Adding Lottie animation in Streamlit WebApp

In this article, we will see how we can embed Lottie Animation in our Streamlit WebApp using Python. What is Lottie Animation? An animation file format calle...

5 min read

Todo list app using Flask | Python

There are many frameworks that allow building your webpage using Python, like Django, flask, etc. Flask is a web application framework written in Python. Flask...

3 min read

Python | ToDo GUI Application using Tkinter

Prerequisites : Introduction to tkinter Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, Tkinter is...

5 min read

How to make a Todo List CLI application using Python ?

In this article, we see how to make a Command Line application todo list in python. Todo list is a basic application in which users can add items. It's a list of...

7 min read

How to make a Todo App using PHP & MySQL ?

To create a Todo App using PHP and MySQL, you'll first need to set up a MySQL database to store the tasks. Then, use PHP to create a web interface where user...

4 min read

Build a Todo List App using VueJS

This article provides an in-depth exploration of creating a Todo List application using Vue.js. It covers various aspects of building the application, whether you're...

4 min read

Article Tags : HTML Python Technical Scripter Web Technologies +2 More

Practice Tags : python