DjangoLovers

قسمت اول: )ایجاد فولدر، نصب جنگو، ایجاد پروژه، انتقال به گیت هاپ(

1. ایجاد فولدری به نام DjangoLovers
2. باز کردن cmd داخل فولدر و ایجاد محیط مجازی: python -m venv env
3. داخل پوشه env و بعد از آن scripts شده و محیط مجازی را اکتیو میکنیم: activate
4. به پوشه اصلی برگشته و برای نصب جنگو اقدام میکنیم: python -m pip install Django
5. پروژه جدید را ایجاد میکنیم: django-admin startproject djangolovers .
6. وارد فولدر djangolovers می شویم و سرور را فعال میکنیم: manage.py runserver
7. حال به پورت <http://localhost:8000/> رفته پروژه را بالا می آوری.
8. در فولدر اصلی پروژه در cmd دستور git init را برای فایل گیت تایپ میکنیم.
9. برای اضافه کردن همه فایل ها به پروژه دستور git add . را تایپ میکنیم.
10. در cmd یک توضیح برای ارسال به گیت هاب مینویسیم: git commit -m 'start project'
11. در گیت هاپ یک ریپازیتوری جدید میسازیم به نام djangolovers
12. پروژه را به داخل ریپازیتوری میفرستیم :

git remote add origin https://github.com/SeyedAmirSamavatian/djangolovers.git

git branch -M main

git push -u origin main

قسمت دوم : (صفحه اول، برخی تنظیمات )

1. تنظیمات settings.py :

Import os

MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

MEDIA\_URL = '/media/'

1. تنظیم urls.py در اپ اصلی:

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root= settings.MEDIA\_ROOT)

1. در تنظیمات settings.py مقدار DIRS در TEMPLATES را برابر 'templates' قرار میدهیم.
2. در فولدر اصلی پروژه یک فولدر با نام templates میسازیم و بعد هم یک پوشه به نام media در پوشه اصلی پروژه میسازیم و در داخل آن یک فولدر به نام imagesSite میسازیم و عکس پس زمینه صفحه اول سایت را در آن قرار میدهیم.
3. در پوشه templates یک فایل به نام index.html قرار میدهیم.
4. در فایل urls.py اپ اصلی آدرس زیر را اضافه میکنیم:

from views import welcome

urlpatterns = [

…..

path('/', welcome , name='welcome'),

]

1. در اپ اصلی فایل views.py را ایجاد کرده و فانکشن زیر را مینویسیم:

from django.shortcuts import render

def welcome(request):

return render(request,'index.html')

1. فایل فونت وزیر را دانلود میکنیم و در فولدر media و فولدر fonts میریزیم.
2. در فایل index.html کد های زیر را مینویسیم:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Django Lovers</title>

</head>

<style>

@font-face {

font-family: "vazir";

src: url("media/fonts/Vazir-Light.ttf");

}

.center{

display: flex;

align-items: center;

justify-content: center;

text-align: center;

flex-direction: column;

}

a{

width: 80px;

background: #7DC0F9;

text-decoration: none;

cursor: pointer;

border: solid 2px #ddd;

border-radius: 5px;

padding: 5px;

font-size: 25px;

margin: 35px;

font-family: 'vazir';

color: #fff;

box-shadow:2px 2px 4px rgba(0, 0, 0, 0.3);

}

p{

font-size:25px;

margin: 0 30px;

font-family: 'vazir';

}

div{

margin: 10px;

}

</style>

<body style="background-size: 100% 100%; background-image: url('media/imagesSite/djangoBack.jpg'); background-attachment: fixed; background-repeat: no-repeat;">

<div style="width: 100%; height: 100%; display: flex; align-items: center;justify-content: center;">

<div class="center" style="width: 50%;height: 100%;right:0%;bottom: 0px;position: absolute;">

<div class="center" style="width:60%;height: 100%;display: flex; flex-direction: column" >

<div>

<img width="150px" src="media/imagesSite/django-python-logo.png" alt="">

</div>

<div class="center" style="flex-direction: row;">

<p style="font-size:50px;">Django Lovers</p>

</div>

<div class="center" style="flex-direction: row;">

<p>...به ما بپیوندید </p>

<img width=80px src="media/imagesSite/cartoon.jpg" alt="">

</div>

<div class="center" style="flex-direction: row;">

<a href="" class="center">ثبت نام</a>

<a href="" class="center">ورود</a>

</div>

</div>

</div>

</div>

</body>

</html>

10\_ انتقال به github :

Git add .

Git commit -m "part 2"

git remote add origin https://github.com/SeyedAmirSamavatian/djangolovers.git

git push -u origin main

قسمت سوم (static و bootstap و css و base.html و ... ) :

1. فولدر static را در پوشه اصلی پروژه میسازیم
2. در setting.py کد زیر را مینویسیم:

STATIC\_URL = "static/" پیشفرض

STATICFILES\_DIRS = [BASE\_DIR, "satatic"] دستی وارد شود

1. مسیر فایل static را در فایل urls.py اپ اصلی اضافه میکنیم:

if settings.DEBUG:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

1. در فولدر static فایل base.css و base.js را میسازیم.
2. در پوشه templates یک فایل به نام base.html میسازیم.
3. فولدر static را در فایل base.html وارد میکنیم: {% load static %}
4. لینک css های فایل base.html را از پوشه static به قسمت هدر این فایل اضافه می کنیم:

<link rel="stylesheet" href="{% static 'base.css' %}">

1. در فایل base.html در قسمت هدر Link و در پایین صفحه script مربوط به بوت استرپ 5.3.0 را اضافه میکنیم و فایل base.js را هم در آخر اضافه میکنیم:

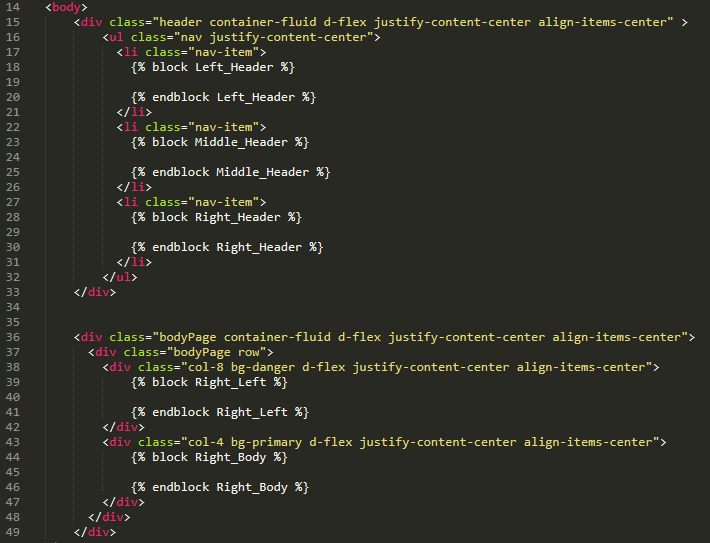
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe" crossorigin="anonymous"></script>

<script src="{% static 'base.js' %}"></script>

1. در قسمت تگ title هم {% block title %} {% endblock title %} را مینویسیم.

10\_ داخل تگ body هم کد های زیر را مینویسیم:



11 – داخل فایل base.css هم کد های زیر را مینویسیم:

.header{

background: #CFC8C8;

height:50px;

width: 80% !important;

margin:30px 0;

border-radius: 5px;

}

.bodyPage{

width: 85% !important;

height: 80vh;

padding: 0 50px;

}

.bodyPage\_left,.bodyPage\_right{

height: 90%;

margin: 0 20px!important;

}

12\_ انتقال به github :

git add .

git commit -m "part 3"

git push -u origin main

قسمت چهارم (database و Admin و login و registry ) :

1. نصب mysql : pip install mysqlclient
2. ساخت یک دیتابیس به نام jangolovers در wampserver
3. تنظیم دیتابیس mysql در settings.py :

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': ' mydatabase',

'USER': 'root',

'PASSWORD': '',

'HOST': '127.0.0.1',

'PORT': '3306',

}

}

1. میگریشن و میگریت:

Manage.py makemigrations

Manage.py migrate

1. ایجاد سوپر یوزر: python manage.py createsuperuser
2. ساخت اَپ accounts : manage.py startapp accounts
3. اضافه کردن نام اَپ در فایل settings.py و اضافه کردن آدرس در urls.py اَپ اصلی.

settings.py:

INSTALLED\_APPS = [

…..

'accounts',

]

----------------------------------------------------------------------------

urls.py:

from django.urls import path , include

urlpatterns = [

…

path('accounts/', include( 'accounts.urls' )),

]

1. ساخت فایل urls.py در اَپ accounts و ایمپرت کردن path و ساخت آدرس registery برای صفحه ثبت نام و ایمپرت فانکشن رجیستری از فایل views.py:

from django.urls import path

from .views import registery

urlpatterns = [

path('registery/', registery , name='registery'),

]

1. ساخت فانکشن registery در صفحه views.py :

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login, logout

from django.contrib import messages

from .forms import RegisteryUserForm

def registery(request):

if request.method == 'POST':

form = RegisteryUserForm (request.POST)

if form.is\_valid():

email = form.cleaned\_data['email']

user\_email\_count = User.objects.filter(email = email).count()

if user\_email\_count == 0:

form.save()

username = form.cleaned\_data['username']

password = form.cleaned\_data['password1']

user = authenticate(username = username , password = password)

login(request, user)

return redirect('welcome')

else:

messages.success(request, (' کاربر با این ایمیل قبلا ثبت نام کرده است '))

else:

form = RegisteryUserForm ()

return render(request, 'register.html' , {'form' : form})

10\_ در فایل models.py از اَپ accounts کد های زیر را اضافه میکنیم:

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

image = models.ImageField(upload\_to='profile\_pics', blank=True)

def\_\_str\_\_(self):

return self.user.username + '->' + self.user.email

11\_ ساخت فایل forms.py در اَپ accounts :

from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

from .models import Profile

class RegisteryUserForm(UserCreationForm):

email = forms.EmailField(label='ایمیل')

image = forms.ImageField(required =False)

class Meta:

model = User

fields = ( 'first\_name', 'last\_name', 'username', 'email', 'password1', 'password2', 'image' )

def save(self, commit=True):

user = super(RegistrationForm, self).save(commit=False)

user.email = self.cleaned\_data['email']

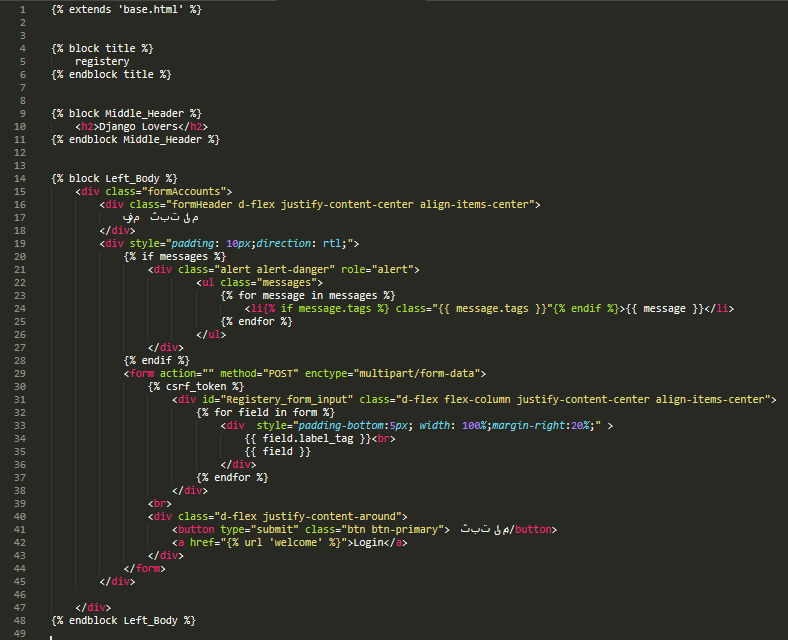
if commit:

user.save()

profile = Profile(user=user, image=self.cleaned\_data.get('image'))

profile.save();

return user

12 – ساخت فایل registery.html در templates :

13 – اضافه کردن کد های زیر به base.css :

.bodyPage\_left,.bodyPage\_right{

height: 80%;

margin: 0 20px!important;

}

.formAccounts{

width: 90%;

height: 100%;

background: #F5F5F5;

border-radius: 15px;

}

.formHeader{

width: 100%;

height: 30px;

background: #CFC8C8;

border-radius: 15px 15px 0 0;

}

14\_ بعد از انجام تنظیمات بالا باید برای کار با عکس ها اول پکیج Pillow را نصب کرد و بعد یکبار دیگر برای شناساندن مدل پروفایلی که در models.py نوشتیم عملیات میگریشن را انجام میدهیم:

python -m pip install Pillow

manage.py makemigrations

manage.py migrate

برای دیدن اطلاعات پروفایل در پنل ادمین در اَپ accounts در فایل admin.py کد زیر را وارد میکنیم:

from .models import Profile

admin.site.register(Profile)

برای نمایش بهتر فیلد های فرم در فایل base.js کد های زیر را مینویسیم:

const Registery\_form\_input = document.getElementById('Registery\_form\_input');

const inputList = Registery\_form\_input.getElementsByTagName('input');

if(Registery\_form\_input && inputList){

for (let i = 0; i < inputList.length; i++) {

inputList[i].classList.add('form-control');

inputList[i].style.width = '80%';

}

}

انتقال به github :

git add .

git commit -m "part 4"

git push -u origin main

قسمت پنجم( فرم لاگین ) :

1 – در فایل urls.py درون اَپ accounts آدرس login را اضافه میکنیم و فانکشن آن را از views.py ایمپرت میکنیم :

from django.urls import path

from .views import registery , login

urlpatterns = [

path('registery/', registery , name='registery'),

path('login/', login\_view, name='login'),

]

2 – اصلاح فانکشن RegisteryUserForm در فایل forms.py برای حذف username و جایگزین کردن ایمیل به جای آن:

class RegisteryUserForm(UserCreationForm):

email = forms.EmailField(label='ایمیل')

image = forms.ImageField(required =False)

class Meta:

model = User

fields = ('first\_name', 'last\_name','email', 'password1', 'password2', 'image') # username حذف شد

def save(self, commit=True):

user = super(RegisteryUserForm, self).save(commit=False)

user.email = self.cleaned\_data['email']

user.username = self.cleaned\_data['email']

if commit:

user.save()

profile = Profile(user=user, image=self.cleaned\_data.get('image'))

profile.save();

return user

همچنین فانکشن registery در فایل views.py راهم اصلاح کرده و مقدار username را مانند زیر قرار میدهیم:

username = form.cleaned\_data['email']

3 – فانکشن لاگین در فایل views.py را مینویسیم:

from .forms import RegisteryUserForm, LoginForm

def login\_view(request):

if request.method == 'POST':

form = LoginForm(request.POST)

if form.is\_valid():

email = form.cleaned\_data['email']

password = form.cleaned\_data['password']

user = authenticate(request, username = email, password = password)

if user is not None:

login(request, user)

return redirect('welcome')

else:

messages.success(request, 'ایمیل یا پسورد اشتباه است')

else:

form = LoginForm()

return render(request, "login.html", {'form':form})

4 – ساخت فایل login.html در templates :

