# ساخت نمای کلی برنامه شبکه اجتماعی با جنگو:

همانطور که میدانید migrate یک دستور sql درست میکند و تغییرات model را روی دیتابیس اعمال میکند اگر بخواهیم این دستورات sqlی را ببینیم میتوانیم از دستور زیر استفاده کنیم :

python manage.py sqlmigrate name\_app name\_migration

python manage.py sqlmigrate posts 0001\_initial

===================================================================================ساخت جدول دیتابیس پست ها

===================================================================================آموزش فیلد lookups و شل جنگو:

fields lookup :

فیلد lookupها امکاناتی هستند که وقتی شما از get , filter و exclude استفاده میکنید نتایج درخواست شما را دقیق‌تر نمایش دهد مثلاً :

exact : دقیقاً همان چیز را برای من بیار iexact : دقیقاً ماننده اگذکتت بدون حساس بودن به حروف بزگ و کوچک

contains : شامل شدن icontains: بدون حساس بودن به حروف بزگ و کوچک

https://docs.djangoproject.com/en/3.1/ref/models/querysets/

============================================================================================نشان دادن پست ها:

دقت شود که در reverse که معمولاً در get\_absolute\_url استفاده می‌شود نمیتوانیم از fields lookupها استفاده کنیم و باید از روش dot notation استفاده کرد

============================================================================================آموزش ثبت نام و لاگین کاربر در جنگو

[https://docs.djangoproject.com/en/3.1/topics/auth/default/#auth-web-requests](https://docs.djangoproject.com/en/3.1/topics/auth/default/" \l "auth-web-requests)

===================================================================================آموزش سفارشی سازی فرم ها در جنگو

مقداری به نام for در تگ label وجود دارد که اگر مقدار دهی شود خودش را به اون input مورد نظر متصل میکند که اگر بر روی label کلیک کردید input متصل فعال شود

<label for="{{ form.username.id\_for\_label }}">Username</label>  
{{ form.username }}<br/>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[https://docs.djangoproject.com/en/3.1/ref/forms/fields/#charfield](https://docs.djangoproject.com/en/3.1/ref/forms/fields/" \l "charfield)

seaarch :error\_messages

messages = {  
 'required':'این فیلد اجباری است',  
 'invalid':'لطفا یک ایمیل معتبر وارد کنید',  
 'max\_length':'تعداد کاراکترها بیشتر از حد مجاز است'  
}

username = forms.CharField(error\_messages=messages,…)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

https://docs.djangoproject.com/en/3.1/topics/forms/

============================================================================================

آموزش گسترش User پیشفرض جنگو

[https://docs.djangoproject.com/en/3.1/topics/auth/customizing/#extending-the-existing-user-model](https://docs.djangoproject.com/en/3.1/topics/auth/customizing/" \l "extending-the-existing-user-model)

روش admin.StackedInline به صورت خطی خطی در کنار هم قرار میدهد

روش admin.TabularInline به صورت کنار هم نمایش میدهد

from django.contrib.auth.admin import UserAdmin  
from django.contrib.auth.models import User

from .models import Profile  
  
class ProfileInline(admin.StackedInline):  
 model = Profile  
 can\_delete = False

class ExtendedProfileAdmin(UserAdmin):  
 inlines = (ProfileInline,)  
  
admin.site.unregister(User)  
admin.site.register(User, ExtendedProfileAdmin)

==================================================================آموزش سیگنال ها در جنگو

در این پروژه درست است که profile را به user متصل کرده‌ایم اما در‌واقع وجود ندارد چرا که موقع register اصلاً profile رو صدا نزدیم میتوانیم این کار را در فانکشن user\_register به صورت زیر انجام دهیم :

def user\_register(request):  
 …….  
 user = User.objects.create\_user(cd['username'], cd['email'], cd['password'])

p1 = Profile(user=user)

p1.save()  
 login(request, user)  
 ……..

حال وقتی user جدیدی ساخته می‌شود profileش هم ساخته می‌شود و بعداً میتوانید bio و age را مقدار دهی کنید

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

اما روش بهتر استفاده از signal های جنگو میباشد

https://docs.djangoproject.com/en/3.1/topics/signals/

مفهوم کلی به این صورت است که داخل برنامه تون یک اتفاقی رخ میدهد بعد میخواهید به دلیل اینکه این اتفاق خاص رخ داده است یک سیگنالی به قسمت دیگه ای از برنامه تون ارسال شود و اون قسمت کار خاصی رو انجام دهد

به دو روش انجام می‌شود روش اول ()connect و روش دوم استفاده از دکوریتور

def save\_profile(sender, \*\*kwargs):  
 if kwargs['created']:  
 p1 = Profile(user=kwargs['instance'])  
 p1.save()  
  
post\_save.connect(save\_profile, sender=User)

connect دو مقدار اصلی دریافت میکند . اولین مقدار receiver نام دارد در این مثال بعد از اینکه چیزی ذخیره شد (post\_save) تابع save\_profile سیگنالی از طرف User دریافت میکند (توجه شود که تابع save\_profile حتماً حتماً باید sender و kwargs جزو پارامتر های ورودیش باشند) خوبی سیگنال ها این است که داخل kwargs اصلاعات مهمی رو قرار میدهد که ما میتوانیم از آن‌ها استفاده کنیم . اولین مقدار created است که اگر کار ساخت user جدید به خوبی انجام شد مقدارش true و اگر user جدید ساخته نشد مقدارش false میباشد دومین مقدار instance میباشد که دقیقاً همان objectی که ساخته شده است را در خود ذخیره کرده است

==================================================================دسترسی و تغییر مدل پروفایل:

در shell توجه شود که اگر بخواهید به age از طریق user دسترسی پیدا کنید باید اینگونه رفتار کنید :

from django.contrib.auth import User

user = User.objects.get(username = ‘kevin’)

user.age = 20

user.profile.age = 20

user.save()

user.profile.save()

خط‌های سبز درست و خط‌های قرمز اشتباه میباشد \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

اگر رابطه‌ای که بین profile و User وجود دارد به صورت Foreingkey بود آنگاه هنگام مفدار دهی باید به صورت زیر رفتار میکردید :

user.profile\_set.age = 20

اما میتوانید به foreingkey ها یک releated\_name بدهید که کار آسان‌تر شود

==================================================================ساخت داشبورد کاربر

===========================================================نشان دادن پست ها در داشبورد

در جنگو هنگامیکه در templateها از تگ for استفاده میکنید داخل تگ for تگ دیگری به نام empty وجود دارد که اگر داخل for چیزی وجود نداشت کاری که در تگ empty قرار دادید انجام شود :

{% for post in posts %}  
 <div class="card">  
 <p class="card-body"><a href="{{ post.get\_absolute\_url }}">{{ post.body }}</a></p>  
 </div>  
{% empty %}  
 <p>This user has no posts</p>  
{% endfor %}

=========================================================== آموزش ModelForm در جنگو

ModelFormها در جنگو روشی برای ساخت فرم ها به طور مستقیم از modelها است

https://docs.djangoproject.com/en/3.1/topics/forms/modelforms/

به دلیل اینکه از modelform ها استفاده میکنیم دیگر لازم نیست clean data کنیم و تک تک مقادیر را ارسال کنیم خوده فیلدی که در modelform قرار داشت میداند که به چه فیلدی در model مربوط می‌شود

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

به دلیل اینکه در modelform تنها body را از کاربر گرفتیم و مقادیر دیگر را باید خودمان به صورت اتوماتیک ذخیره کنیم باید به روش زیر عمل کنید :

def add\_post(request, user\_id):

……….  
 if form.is\_valid():  
 new\_post = form.save(commit=False)  
 new\_post.user = request.user  
 new\_post.slug = slugify(form.cleaned\_data['body'][:30])  
 new\_post.save()  
 …….

همان‌طور که مشاهده میکنید بعد از اینکه از شرط valid بودن گذشت میگوییم که save شو . اما به دلیل اینکه هنوز slug و user مشخص نیست باید صبر کند .commit=false دقیقاً یعنی اینکه form جدید را ذخیره کن اما صبر کن تا مقادیر دیگر نیز وارد کنیم در آخر new\_post را save میکنیم

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Django Utils

https://docs.djangoproject.com/en/3.1/ref/utils/

در این بخش از document توضیحات کاملی درباره ابزارت خوبی که جنگو در اختیار ما قرار میدهد وجود دارد

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

slugify :

این یکی از ابزارات جنگو میباشد که textی را میگیرد و آن را به صورت slug تحویل میدهد

new\_post.slug = slugify(form.cleaned\_data['body'][:30])

=========================================================== آموزش محدود کردن دسترسی کاربران در جنگو

برای اینکار در جنگو میتوان از دکوراتور login\_required استفاده کرد. این دکوراتور وضعیت لاگین کاربر را چک کرده و اگر کاربر لاگین نکرده بود به صفحه لاگین منتقل خواهد شد

[https://docs.djangoproject.com/en/3.1/topics/auth/default/#limiting-access-to-logged-in-users](https://docs.djangoproject.com/en/3.1/topics/auth/default/" \l "limiting-access-to-logged-in-users)

هنگامیکه از دکوراتور login\_required استفاده میکنید جنگو به طور پیش‌فرض کاربری که login نکرده باشد را به صفحه accounts/login هدایت میکند اما چون در این پروژه اسم app ما account میباشد و آدرس url مربوط به login برابره account/login میباشد به ارور میخوریم دور روش وجود دارد برای حل این مشکل :

1)

@login\_required(‘account:login’)

2)

in settings.py ===> LOGIN\_URL = 'account:login'

@login\_required

روش دوم به این صورت است که آدرس کلی login\_url را داخل پروژه به کلی تغییر میدهد

اشکالی که اکنون وجود دارد این است که بعد از اینکه کاربر login کرد به صفحه all\_post فرستاده می‌شود اما باید به صفحه‌ای که درخواست کرده بود و نیاز به login داشت redirect شود . دکوریتور login\_required پارامتری دارد به نام next که میتوانیم از آن استفاده کنیم به این صورت که :

def user\_login(request):  
 next = request.GET.get('next')  
 …………………………..  
 login(request, user)  
 messages.success(request, 'you logged in successfully', 'success')  
 if next:  
 return redirect(next)  
 return redirect('posts:all\_posts')  
 else:  
 messages.error(request, 'wrong username or password', 'warning')  
 ………………………………..

در این صورت که در فانکشن user\_login از request با متد GET مقدار next را بیرون میکشیم (توجه شود که اگر وجود نداشت None است) حال هنگام redirect بررسی میکنیم که next وجود داشت یا نه اگر وجود داشت آن را به next ریدارکت کن اما اگر وجود نداشت به صفحه all\_post هدایت کن

==================================================================تغییر برنامه

in account/views.py

@login\_required  
def user\_dashboard(request, user\_id):  
 user = get\_object\_or\_404(User, pk=user\_id)  
 posts = Post.objects.filter(user=user)  
 self\_dash = False  
 if request.user.id == user\_id:  
 self\_dash = True  
 return render(request, 'account/dashboard.html', {'user':user, 'posts':posts, 'self\_dash':self\_dash})

in account/templates/account/dashboard.html

{% if self\_dash %}  
<a href="{% url 'posts:add\_post' request.user.id %}">Add post</a>  
{% endif %}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

in posts/views.py

def add\_post(request, user\_id):  
 if request.user.id == user\_id:  
 ……………………..  
 else:  
 return redirect('posts:all\_posts')

این تغییرات به این منظور است که هر کسی نتواند برای دیگری post بگذارد و هر کسی بتواند dashboard خودش را تنها ببیند

==================================================================آموزش حذف پست ها در جنگو

==================================================================آموزش ویرایش پست ها در جنگو

برای ویرایش پیشنهاد می‌شود که حتماً از ModelForm استفاده کنید چرا که میتوانید یک instance به این modelform ها ارسال کنید تا بتوانید مقدادیر را مقدار دهی کنید :

in posts/views.py ====> def post\_edit

==================================================================ساخت model نظرات

در مدل ها اگر دو مدل به صورت foreignkey به هم ربط داشت و مدلی که باید در foreignkey قرار بگیرد هنوز ساخته نشده بود باید داخل ’’ قرار گیرد

اگر در مدل ها یک مدل به خودش رابطه foreignkey داشت میباید یا اسم مدل را داخل ’ ’ قرار دهیم یا از ’self’ استفاده کنیم

==================================================================آموزش backward relation در جنگو

در پروژه مدل comment را به post ربط داده‌ایم یعنی از طریق comment میتوانیم به راحتی به post برسیم

حال اگر بخواهیم بصورت برعکس در یک رابطه حرکت کنیم باید بصورت زیر عمل کنیم :

p1= Post.objects.last()

p1.comment\_set.all()

به وسیله set\_ میتوانیم یک رابطه را به صورت برگشتی طی کنیم

توجه شود یک setـ در رابطه‌های چند به چند و یک به چند استفاده می‌شود ولی در رابطه‌های یک به یک کافیست اسم مدل را بیاوریم (در رابطه‌های یک به یک انگار که رابطه را برگشت طی نمیکنیم )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

توجه شود که میتوانیم از related\_name استفاده کنید و از روش بالا استفاده نکنیم

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

اگر بخواهید مدلتون را جوری تغییر دهید که نتوانید به صورت برگشت طی کرد related\_name را برابر +قرار میدهید

class Comment(models.Model):  
 ……...  
 post = models.ForeignKey(Post, on\_delete=models.CASCADE, related\_name='+')

……..

==================================================================نشان دادن کامنت ها

https://docs.djangoproject.com/en/3.1/ref/templates/builtins/

==================================================================ساخت فرم کامنت

==================================================================آموزش متا کلاس ها در model جنگو

https://docs.djangoproject.com/en/3.1/ref/models/options/

در این قسمت میخواهیم که comment ها را بر از جدید به قدیم چاپ کند روش اول کلاس متا در مدل بود که به صورت زیر است :

class Comment(models.Model):

……..

class Meta:

ordering = (‘-created’,)

……….

روش دوم در view ها میباشد که به صورت زیر استفاده می‌شود :

def post\_detail(request, year, month, day, slug):

……..

comments = Comment.objects.filter(post=post, is\_reply=False).order\_by(‘-created’)

………

=========================================================== آموزش اضافه کردن ویژگی های بیشتر به modelForm در جنگو

[https://docs.djangoproject.com/en/3.1/topics/forms/modelforms/#overriding-the-default-fields](https://docs.djangoproject.com/en/3.1/topics/forms/modelforms/" \l "overriding-the-default-fields)

in posts/forms.py =======> class AddCommentForm

=========================================================== نمایش پاسخ کامنت ها

{% for comment in comments %}

…….

{% for reply in commnet.rcomment.all %}

………..

<p> reply.body </p>

……….

{% endfor %}

………..

{% endfor %}

===========================================================ساخت فرم پاسخ کامنت

=========================================================== ویرایش پروفایل کاربر

=========================================================== ویرایش مدل user

[https://docs.djangoproject.com/en/3.1/topics/forms/modelforms/#providing-initial-values](https://docs.djangoproject.com/en/3.1/topics/forms/modelforms/" \l "providing-initial-values)

in account/views.py ====> def edit\_profile

in account/forms.py =====> class EditProfileForm

===========================================================

در این قسمت با نحوه لاگین کاربر با شماره تلفن در جنگو آشنا میشید. برای اینکار شما نیاز دارید که شماره تلفن کاربر رو ذخیره کنید. بعد از اون به یک سرویس دهنده ارسال پیامک نیاز دارید. ساختار کلی به این شکل هستش که با استفاده از ماژول random پایتون یک عدد تصادفی انتخاب کرده و به کاربر ارسال میشود. یعد از اون کاربر یه یک صفحه دیگه هدایت میشه که باید کد دریافتی رو وارد کنه. اگر کد وارد شده درست بود کاربر لاگین خواهد شد

زمانی که میخواید از یک سرویس دهنده ارسال پیامک استفاده کنید، اون سرویس دهنده یک api در اختیار شما قرار میده و میتونید براساس اون api در جنگو به کاربر پیامک ارسال کنید.

زمانی که پیامک به کاربر ارسال شد، کاربر به یک صفحه دیگر منتقل شده و یک فرم به او نشان داده میشود.کاربر کد دریافتی را وارد کرده و اگر کد مطابقت داشت، کاربر لاگین میشود.