# 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

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