تکمیل مدل پکیج ها

آماده سازی مدل پست و ترسیم آن در سایت دی بی دیاگرام و ترسیم مدل یوزر برای دسترسی های پرایمری کی

ساخت اپلیکیشن جدید برای پست

class post(models.Model):

*# image*

*#author*

    title = models.CharField(*max\_length*=255)

    content = models.TextField()

*# tag*

*# category*

    coounted\_viwes = models.IntegerField(*default*=0)

    status = models.BooleanField(*default*=False)

    published\_date = models.DateTimeField(*null*=True)

    created\_date = models.DateTimeField(*auto\_now\_add*=True)

    updated\_date = models.DateTimeField(*auto\_now*=True)

معرفی متا کلاس برای مدل

*from* django.db *import* models

class post(models.Model):

    title = models.CharField(*max\_length*=255)

    content = models.TextField()

    coounted\_viwes = models.IntegerField(*default*=0)

    status = models.BooleanField(*default*=False)

    published\_date = models.DateTimeField(*null*=True)

    created\_date = models.DateTimeField(*auto\_now\_add*=True)

    updated\_date = models.DateTimeField(*auto\_now*=True)

    class Meta:

        ordering = ['-created\_date']

    def \_\_str\_\_(*self*):

*return* self.title

و دکوراتورadmin.site.register کاستومایز کردن پنل ادمین برای مدل پست و سپس رجیستر کردن با حالت

@admin.register(post)

class PostAdmin(admin.ModelAdmin):

    list\_display = ('id','title', 'status', 'created\_date')*# چه فیلد هایی از آن سطر در صفحه نمایش داده شوند*

    list\_filter = ('status', )*# افزودن یک گزینه فیلتر بر اساس استتوس ها و یا هر فیلد دیگری*

    search\_fields = ('title', )*# یک سرچ فیلد میگذارد که یک عبارت را مثلا در ستون تایتل ها سرچ میکند*

    empty\_value\_display = '-empty-' *# اگر فیلدی خالی بود خالی نشان نده و بجای آن عبارتش را نشان بده*

    fields = ('title', 'content', 'published\_date') *# وقتی بر روی یک سطر کلیک میکنیم کدام بخش ها قابل تغییر باشند*

    ordering = ('created\_date',  )*# بر اساس تاریخ ایجاد صفحه ادمین را سورت میکند*

*#admin.site.register(post, PostAdmin)*

حال وقتی مدل را ساختیم چند پست با وضعیت انتشار ترو و فالس ایجاد میکنیم و میخواهیم حالا در صفحه به دو صورت زیر پست های در حال انتشار را ایجاد میکنیم

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

</head>

<body>

    {% *for* post in posts %}

        {% *if* post.status == 1 %}

            {{post.title}}

        {% *else* %}

            {{post.created\_date}}

        {% *endif* %}

    {% *endfor* %}

</body>

</html>

با استفاده از فیلتراسیون مدل

def b\_home(*req*):

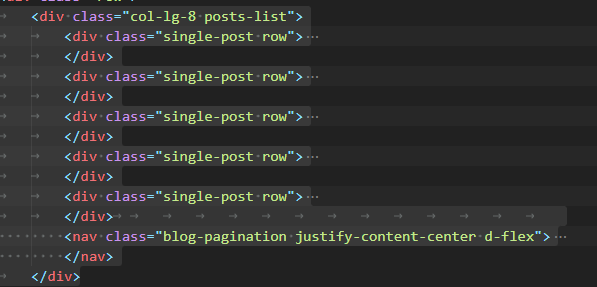
    posts=post.objects.filter(*status*=1)

    context = {

        'posts':posts

    }

قبل از اینکه پست ها را در صفحه اصلی نمایش دهیم در یک صفحه تستی نمایش میدهیم و سپس محتویات را به صفحه اصلی میبریم و سپس طبق فرم زیر آنرا میسازیم



و ارسال به تابع های ویوurlارسال پارامتر از طریق

برای ارسال پارامتر ابتدا مدل را برای نویسنده و عکس و کتگوری وتگ آماده میکنیم

class category(models.Model):

    name = models.CharField(max\_length=255)

    def  \_\_str\_\_(self):

        return self.name

class Tag(models.Model):

    name = models.CharField(max\_length=255)

    def  \_\_str\_\_(self):

        return self.name

from django.contrib.auth.models import User

class post(models.Model):

    image = models.ImageField(upload\_to = 'blog', default= 'default.jpg')

    author = models.ForeignKey(User, on\_delete=models.CASCADE)

    title = models.CharField(max\_length=255)

    content = models.TextField()

    tags = models.ManyToManyField(Tag)

    category = models.ManyToManyField(category)

    counted\_viwes = models.IntegerField(default = 0)

comment\_count = models.IntegerField(default = 0)

    status = models.BooleanField(default=False)

    published\_date = models.DateTimeField(null=True)

    created\_date = models.DateTimeField(auto\_now\_add=True)

    updated\_date = models.DateTimeField(auto\_now=True)

    class Meta:

        ordering = ['-created\_date']

    def \_\_str\_\_(self):

        return self.title

**Username**: {{ user.username }}

**User Full name**: {{ user.get\_full\_name }}

**Email**: {{ user.email }}

**Session Started at**: {{ user.last\_login }}

وقتی میخواهیم از فیلد نویسنده استفاده کنیم میتوانیم از مقادیر زیر استفاده کنیم

from .models import post, category, Tag

admin.site.register(category)

admin.site.register(Tag)

from django.urls import path

from .views import b\_home, b\_single, search

app\_name = 'blog'

urlpatterns = [

    path('', b\_home, name= 'blog-home'),

    path('category/<str:cat>', b\_home, name='home-with-cat’),

    path('tag/<str:tag>', b\_home, name='home-with-tag’),

    path('<str:username>', b\_home, name='home-with-author' ),

]

def b\_home(req, cat=None, username=None, tag=None):

    posts = post.objects.filter(status=1)

    if cat:

        posts = posts.filter(category\_\_name=cat)

    if username:

        posts = posts.filter(author\_\_username=username)

    if tag :

        posts = posts.filter(tags\_\_name=tag)

    context = {

        'posts':posts

    }

    return render(req,'blog/blog-home.html', context=context)

گاها ما نمیتوانیم پارامتری را از طریق تابع ویو فراخوانی کنیم چون از مقادیر آبجکت های درون صفحه نیستند و بصورت دستی توسط کاربر ارسال میشوند

مانند سرچ باکس ها . در اینگونه از مواقع ما پارا متر را از طریق متغییری در یو ار ال ارسال میکنیم

حالا در توابع دیگر نیازی نیست آن پارامتر را بررسی کنیم و درون خود ویو با متود گت از رکوئست بررسی میکنیم

    path('search/', search, name = ''home-with-search'),

def search(request):

    posts = post.objects.filter(status=1)

    if request.GET.get('search') is not None:

        key = request.GET.get('search')

        posts = posts.filter(content\_\_contains=key)

    context = {

        'posts':posts

    }

    return render(request, 'blog/blog-home.html', context=context)

<div class="single-sidebar-widget search-widget">

<form class="search-form" action="{% url 'blog:search' %}">

    <input placeholder="Search Posts" name="search" type="text" onfocus="this.placeholder = ''"

                                onblur="this.placeholder = 'Search Posts'">

     <button type="submit"><i class="fa fa-search"></i></button>

</form>

نکات اصلاحی در تگ های اچ تی ام ال : مثلا وقتی در حلقه کتگوری بینشان کاما بگذارین نمیخواهیم برای آخرین کتگوری کاما در نظر گرفته شود

دو حالت وجود دارد

<ul class="tags">

     {% for cat in post.category.all %}

           {% if not forloop.last %}

                 <li><a href="#">{{cat.name}}, </a></li>

           {% else %}

                 <li><a href="#">{{cat.name}}</a></li>

           {% endif %}

     {% endfor %}

</ul>

{{post.category.all|join:", "}}

بجاب آنکه از حلقه استفاده کنیم میتوانیم بصورت بالا بنویسیم اما در این صورت نمیتوان به کتگوری ها اکشن داد چون مجموع لیستش را میبیند

استفاده کرد Django template date formatبرای نمایش بهتر وضعیت تاریخ میتوان از

<p class="date col-lg-12 col-md-12 col-6"><a href="#">{{post.published\_date|date:"d M Y"}}

همچنین بین جدا کننده ها میتوان هر علامی گذاشت

<p class="date col-lg-12 col-md-12 col-6"><a href="#">{{post.published\_date|date:"d/M/Y"}}

مثلا در نحوه نمایش کانتنت در بلاگ هوم نباید متن کامل قرار بگیرد و گاها نیاز است بر اساس تعداد کارکتر یا تعداد کلمات این کار انجام شوند

{{post.content|truncatewords:2}}

{{post.content|truncatechars:20}}

تمرین چالشی برای نمایش بر اساس پابلیش دیتی که از تاریخ امروز کوچکتر باشد و راهنمایی

\_\_lte -> Less than or equal

\_\_gte -> Greater than or equal

\_\_lt -> Less than

\_\_gt -> Greater than

MyModel.objects.filter(field\_\_gte=5) # field ≥ 5

MyModel.objects.filter(field\_\_lte=5) # field ≤ 5

**Username**: {{ user.username }}

**User Full name**: {{ user.get\_full\_name }}

**Email**: {{ user.email }}

**Session Started at**: {{ user.last\_login }}

ایجاد متد در مدلاسیون و استفاده در صفحه

مد نظر باشد برای ایجاد متد برای مدل درون صفحه اچ تی ام ال نیز باید زمانی که حلقه ای بر روی یک مدل زده میشود تک تک آبجکت ها متد را صدا بزنیم

زیرا هر متد برای یک آبجکت از مدل میباشد

    def snipets(self):

        return self.content[:10] + '...'

                        <p>

                            {{post.snipets}}

                        </p>

    def titleedit(self):

        return self.title.upper()

                        <a class="posts-title" href={% url 'blog:blog-single' pid=post.id %}>

                            <h3>{{post.titleedit}}</h3>