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)*

با استفاده از فیلتراسیون مدل

def b\_home(*req*):

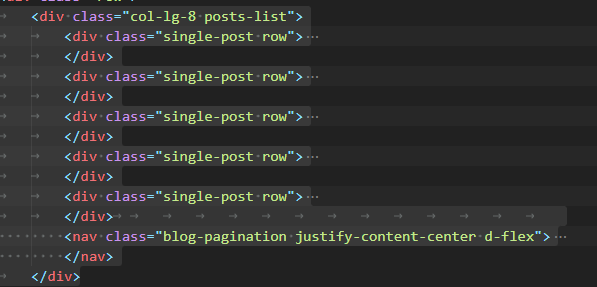
    posts=post.objects.filter(*status*=1)

    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

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

https://docs.djangoproject.com/en/4.0/ref/templates/builtins/#std-templatefilter-date

<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"}}

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 }}

from django.shortcuts import render

from .models import Post

from django.utils import timezone

# Create your views here.

def blog\_home(req):

    posts = Post.objects.filter(published\_date\_\_lte=timezone.now())

    context = {

        'posts' : posts

    }

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

ایجاد متد در مدلاسیون و استفاده در صفحه

تکمیلی

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>

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

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}}

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>

ارسال پارامتر به ویو بصورت مستقیم و بصورت ارسال از طریق یو ار ال

زمانی پارامتری را از طریق یو ار ال ارسال میکنیم که از اتربیوت های استاندارد یک آبجکت نباشد و ورودی توسط کاربر باشد و هر مقداری میتواند باشد

میخواهیم انواع فیلتراسیون را در صفحه بلاگ هوم پیاده سازی کنیم

ابتدا در یو ار ال ها بصورت ساده تعریف میکنیم

from django.urls import path

from .views import \*

app\_name='blog'

urlpatterns = [

    path('', blog\_home , name='blog\_home'),

    path('', blog\_home, name='blog\_home\_with\_tag'),

]

سپس در کد اچ تی ام ال

{% for cat in post.category.all  %}

    <li><a href="{% url 'blog:blog\_home\_with\_tag' %}">{{cat.name}}</a></li>

{% endfor %}

حالا میبینیم که با کلیک بر روی هر تگ به همان صفحه بلاگ می آید . پس لازم است برایش نام آن تگ ارسال شود

{% for cat in post.tag.all  %}

    <li><a href="{% url 'blog:blog\_home\_with\_tag' tag=tag.name %}">{{tag.name}}</a></li>

{% endfor %}

و مجدد در یو ار ال ادیت میکنیم

from django.urls import path

from .views import \*

app\_name='blog'

urlpatterns = [

    path('', blog\_home , name='blog\_home'),

    path('/tag/<str:tag>', blog\_home, name='blog\_home\_with\_tag'),

]

و حالا در ویو تعریفش میکنیم اما با این فرق که مقدار دیفالت نمیدهیم تا اشکال کار را ببینیم

def blog\_home(req, tag):

    posts = Post.objects.filter(tag\_\_name=tag)

    context = {

        'posts' : posts

    }

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

و حال ویو را اصلاح میکنیم

def blog\_home(req, tag=None):

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

    if tag:

        posts = Post.objects.filter(tag\_\_name=tag)

    context = {

        'posts' : posts

    }

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

و حالا برای یوزرنیمو پست هایی با نویسنده یکسان

<div class="user-details row">

   <p class="user-name col-lg-12 col-md-12 col-6"><a href="{% url 'blog:blog\_home\_with\_username' username=post.author.username %}">{{post.author.username}}</a>

<span class="lnr lnr-user"></span></p>

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

<span class="lnr lnr-calendar-full"></span></p>

   <p class="view col-lg-12 col-md-12 col-6">{{post.counted\_views}}</a>

<span class="lnr lnr-eye"></span></p>

   <p class="comments col-lg-12 col-md-12 col-6">{{post.comments}}</a>

<span class="lnr lnr-bubble"></span></p>

</div>

from django.urls import path

from .views import \*

app\_name='blog'

urlpatterns = [

    path('', blog\_home , name='blog\_home'),

    path('/tag/<str:tag>', blog\_home, name='blog\_home\_with\_tag'),

    path('/username/<str:username>', blog\_home, name='blog\_home\_with\_author'),

]

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

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

    if tag:

        posts = Post.objects.filter(tag\_\_name=tag)

    if username:

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

    context = {

        'posts' : posts

    }

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

و در بخش کتگوری

کتگوری در یک صفحه دیگر رفته که از طریق بلاگ هوم اینکلود شده پس اگر آبجکت های کتگوری را به صفحه هوم بفرستیم به اینکلود هم میرود

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

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

    category = Category.objects.all()

    if tag:

        posts = Post.objects.filter(tag\_\_name=tag)

    if username:

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

    if cat:

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

    context = {

        'posts' : posts,

        'category' : category

    }

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

from django.urls import path

from .views import \*

app\_name='blog'

urlpatterns = [

    path('', blog\_home , name='blog\_home'),

    path('tag/<str:tag>', blog\_home, name='blog\_home\_with\_tag'),

    path('username/<str:username>', blog\_home, name='blog\_home\_with\_username'),

    path('category/<str:cat>', blog\_home, name='blog\_home\_with\_category'),

]

{% for cat in category %}

   <li>

     <a href="{% url 'blog:blog\_home\_with\_category' cat=cat.name %}" class="d-flex justify-content-between">

     <p>{{cat.name}}</p>

     </a>

 </li>

{% endfor %}

و حالا ارسال پارامتر از طریق یو ار ال برای پست هایی که در محتوایشان یک عبارتی را دارند که در سرچ باکس میباشد

گاها ممکن است برای ارسال پارامتر مقادیری ارسال شوند که در استاندارد آبجکت ها نیست و نیاز است خودمان کلید و ولیو آن را بسازیم و یا از طریق یک فرم کلید و ولیو ساخته میشوند و ما با روش زیر دریافتش میکنیم

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

            <form class="search-form" action="{% url 'blog:blog\_home\_with\_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>

</div>

from django.urls import path

from .views import \*

app\_name='blog'

urlpatterns = [

    path('', blog\_home , name='blog\_home'),

    path('tag/<str:tag>', blog\_home, name='blog\_home\_with\_tag'),

    path('username/<str:username>', blog\_home, name='blog\_home\_with\_username'),

    path('category/<str:cat>', blog\_home, name='blog\_home\_with\_category'),

    path('search/', blog\_home, name='blog\_home\_with\_search'),

]

تا اینجا رکوئست گت را پرینت میکنیم تا کاربر متوجه ارسال پارامتر بشود

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

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

    category = Category.objects.all()

    if tag:

        posts = Post.objects.filter(tag\_\_name=tag)

    if username:

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

    if cat:

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

    if req.GET.get('search'):

        posts = Post.objects.filter(content\_\_contains=req.GET.get('search'))

    context = {

        'posts' : posts,

        'category' : category

    }

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

که 4 پست آخر را نشان میدهدpopular post بخش

اگر در کلاس متای مدل باشد بر اساس آخرین پست ها بر اساس تاریخ انتشار لیست میکند و اگر در کلاس متا -published\_date مد نظر باشد بر اساس

چیز دیگری باشد اوردرینگ را در خود دریافت آبجکت ها پیاده سازی میکنیم

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

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

    category = Category.objects.all()

    last\_four\_posts = Post.objects.filter(status=True).order\_by('-published\_date')[:3]

    if tag:

        posts = Post.objects.filter(tag\_\_name=tag)

    if username:

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

    if cat:

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

    if req.GET.get('search'):

        posts = Post.objects.filter(content\_\_contains=req.GET.get('search'))

    context = {

        'posts' : posts,

        'category' : category,

        'last\_four\_posts' : last\_four\_posts,

    }

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

            <div class="popular-post-list">

                {% for post in last\_four\_posts %}

                <div class="single-post-list d-flex flex-row align-items-center">

                    <div class="thumb">

                        <img class="img-fluid" src="{{post.image.url}}" alt="">

                    </div>

                    <div class="details">

                        <a href="blog-single.html">

                            <h6>{{post.title}}</h6>

                        </a>

                        <p>Created at {{post.published\_date|date:"d M"}}</p>

                    </div>

                </div>

                {% endfor %}

            </div>

برای کادر تبلیغات بنا به نیاز سایت یک مدل میسازیم

برای اینکار من از یک اپلیکیشن جداگانه استفاده میکنم

ساخته میشودAadvertising اپ

در مدل

from django.db import models

# Create your models here.

class Adv(models.Model):

    name = models.CharField(max\_length=200, default="empty")

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

    link = models.CharField(max\_length=200, default="#")

    class Meta:

        ordering = ('name',)

    def \_\_str\_\_(self):

        return self.name

در ادمین

from django.contrib import admin

from .models import Adv

admin.site.register(Adv)

و مایگرشن

حالا در صفحه بلاگ هوم مثلا تبلیغات وجود دارد

from django.shortcuts import render

from .models import Post, Category

from django.utils import timezone

from advertising.models import Adv

# Create your views here.

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

    adv = Adv.objects.all()[1]

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

    category = Category.objects.all()

    last\_four\_posts = Post.objects.filter(status=True).order\_by('-published\_date')[:3]

    if tag:

        posts = Post.objects.filter(tag\_\_name=tag)

    if username:

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

    if cat:

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

    if req.GET.get('search'):

        posts = Post.objects.filter(content\_\_contains=req.GET.get('search'))

    context = {

        'posts' : posts,

        'category' : category,

        'last\_four\_posts' : last\_four\_posts,

        'ADV' : adv,

    }

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

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

            <a href="{{ADV.link}}"><img class="img-fluid" src="{{ADV.image.url}}" alt=""></a>

</div>

در صفحه هوم 6 پست آخر در پایین صفحه نمایش داده میشوند

from django.shortcuts import render

from .models import \*

from blog.models import Post

def home(req):

    last\_six\_posts = Post.objects.filter(status=True).order\_by('-published\_date')[:6]

    cheap = CheapPackage.objects.all()

    luxuray = LuxuryPackage.objects.all()

    camp = CampingPackage.objects.all()

    context = {'cheap':cheap,'luxuray':luxuray,'camp':camp, 'six\_post':last\_six\_posts}

    return render(req,"home/index.html",context=context)

def about(req):

    return render(req,"home/about.html")

def contact(req):

    return render(req,"home/contact.html")

و در صفحه هوم

            <div class="active-recent-blog-carusel">

                {% for post in six\_post %}

                <div class="single-recent-blog-post item">

                    <div class="thumb">

                        <img class="img-fluid" src="{{post.image.url}}" alt="">

                    </div>

                    <div class="details">

                        <div class="tags">

                            <ul>

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

                                <li>

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

                                </li>

                                {% endfor %}

                            </ul>

                        </div>

                        <a href="#">

                            <h4 class="title">{{post.title}}</h4>

                        </a>

                        <p>

                            {{post.content}}

                        </p>

                        <h6 class="date">{{post.published\_date|date:"d M Y"}}</h6>

                    </div>

                </div>

                {% endfor %}