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
```

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

```
@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

```
<div class="col-lg-8 posts-list">
    <div class="single-post row">...
    </div>
    <div class="single-post row">...
    </div>
    <div class="single-post row">...
    </div>
    <div class="single-post row">...
    </div>
    <div class="single-post row">...
    </div>
    <nav class="blog-pagination justify-content-center d-flex">...
    </nav>
</div>
```

ارسال پارامتر از طریق url و ارسال به تابع های ویو

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)
    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

برای نمایش بهتر وضعیت تاریخ میتوان از `Django template date format` استفاده کرد

```
<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>
```