How to add Pagination in Django Project?

- Pagination system is one of the most common features in blogs, search engine , list of result etc.
- Seeing the popularity of pagination system django developers have build a Paginator class so that web developers do not have to think of the logic to make paginators.
- Paginator Class live in django/core/paginator.py . So to use Paginator Class we first need to import it from **django.core.paginator**
  from django.core.paginator import Paginator
- **Syntax :**
  p = Paginator(list_of_objects , no_of_objects_per_page)
- Here, The first argument is the list of objects which will be distributed over pages.
- The second argument denotes the number of objects that will be displayed on each page. These two arguments are required.
- However Paginator Class takes two more optional arguments which are listed below –
  - **orphans** – its value must be an integer less than the no_of_objects_per_page value. It tells if the last page has minimum number of objects. If the number of remaining objects in the last page is less than or equal to the value of this argument then those objects will be added to the previous page. Default value is 0.
  - **allow_empty_first_page** – It takes Boolean values. Whether or not the first page is allowed to be empty.
- **Note : the first argument need not to be a list. Instead it can be a tuple, queryset or other sliceable object with a count() or __len__() method.**

models.py

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
from django.urls import reverse

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()
    date_posted = models.DateTimeField(default=timezone.now)
    author = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.title
```

How to use Paginator Class?

- Suppose we are developing a blogging website. We have defined Post model in models.py and we have creates 8 such posts. Now in views.py, we have written the following code.

views.py

```
from django.shortcuts import render

from .models import Post

from django.core.paginator import Paginator
```

```
# Create your views here.
def index(request):
        posts = Post.objects.all() # fetching all post objects from database
        p = Paginator(posts, 5)    # creating a paginator object
        # getting the desired page number from url
        page_number = request.GET.get('page')
        try:
                page_obj = p.get_page(page_number) # returns the desired page object
        except PageNotAnInteger:
                # if page_number is not an integer then assign the first page
                page_obj = p.page(1)
        except EmptyPage:
                # if page is empty then return last page
                page_obj = p.page(p.num_pages)
    // creating  context object  using  page_obj
        context = {'page_obj': page_obj}
        # sending the page object to index.html
        return render(request, 'index.html', context)
```

- Here, At the third line , Paginator Class is imported. In the index function we have constructed a paginator object called **p** .
- This paginator creates page objects. Each Page object will have equal number of post objects.
- Then we retrieved the desired page number from the query parameter 'page' from a GET request. This page number is used to extract the correct page object.

Now in index.html :-

```
<!DOCTYPE html>
< html lang="en">
<head>
        <meta charset="UTF-8">
        < meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title> Django Paginator example</title >
</head >
<body>

  <div class="container">
  {% for post in page_obj.object_list %}
    {# note that the list of posts are in the   page_obj.object_list not page_obj #}
    <h1>{{post.title}}</h1>
```

```
      <small>{{post.author}}</small>
      <p>{{post.content}}</p>    <hr/>
   {% endfor %}
</div>

  <center style="margin-bottom: 100px;">
  {% if page_obj.has_previous %}
     <a href="?page={{ page_obj.previous_page_number }}"
        class="btn btn-outline-success">  Previous </a>
  {% endif %}

 <a href="" class="btn btn-outline-danger active">   {{ page_obj.number }} </a>

 {% if page_obj.has_next %}
    <a href="?page={{ page_obj.next_page_number }}"  class="btn btn-outline-success">  Next </a>
  {% endif %}
</center>
</body>
</html>
```

Now Execute Project and send the request from browser.

Methods :

1. Paginator.get_page(page_number)

- It takes a number argument and returns a Page object with the given 1-based index.
- If the passed argument is not a number then it returns the first page.
- If the page number is negative or a number that is greater than the number of pages then it returns the last page.
- Throws **EmptyPage** error if the object list of the page is empty and allow_empty_first_page is set to false in Paginator object.

2. Paginator.page(number)

- It also does the same thing but raises **InvalidPage** error if the page of that number does not exist.

Attributes :

Paginator.count

- returns the total number of objects across all pages.

Paginator.num_pages

- returns the total number of pages

Paginator.page_range

- returns a 1-based range iterator

However, as Paginator class uses Page class to distribute objects, It will be better if we know more about the **Page** Class.

Page Class :

- Generally Page object is used in **Paginator** Class. You rarely have to construct it manually.
  Syntax : **page = Page(object_list , number, paginator)**

- Here **object_list** is the list of objects , **number** argument is used to number the Page object. **paginator** is the Paginator objects for whom this page object is constructed.

Attributes :

Page.object_list

- returns the list of objects

Page.number

- returns the number of the page object

Page.paginator

- returns the corresponding paginator object

Methods :

Page.has_next()

- returns True if the next Page object exits else returns False

Page.has_previous()

- returns True if the previous Page object exits else returns False

Page.has_other_pages()

- Returns True if there's a next or previous page.

Page.next_page_number()

- Returns the next page number. Raises InvalidPage if next page doesn't exist.

Page.previous_page_number()

- Returns the previous page number. Raises InvalidPage if previous page doesn't exist.

Page.start_index()

- Returns the 1-based index of the first object on the page, relative to all of the objects in the paginator's list

Page.end_index()

- Returns the 1-based index of the last object on the page, relative to all of the objects in the paginator's list.