TRAINING DAY16 REPORT

11 JULY 2025

What is Pagination?

Pagination is the process of dividing large sets of data or content into smaller, more manageable parts, usually shown page by page.

For example, when you browse an e-commerce site or scroll through Google search results, you don't see all items or results at once — they are split across pages (like *Page 1, Page 2,* etc.).

Why is Pagination Needed?

Pagination is important because:

1. **Improves performance:**

   Loading all records at once can slow down a web page, especially if the database has hundreds or thousands of entries.
2. **Enhances user experience:**

   It keeps the interface neat and easier to navigate.
3. **Reduces bandwidth usage:**

   The server only needs to send a few items per request.
4. **Supports scalable applications:**

   It helps handle large databases efficiently.

How Pagination Works (General Idea):

1. Fetch all records from the database.
2. Divide the records into groups (for example, 5 or 10 records per page).
3. Display one group (or *page*) at a time on the web page.
4. Provide buttons or links to move between pages (like *Next*, *Previous*, *Page 1, 2,* etc.).

Pagination in Django

Django provides a built-in class called **Paginator** from django.core.paginator, which automatically handles splitting data into pages.

Key Classes and Methods:

- Paginator(object_list, per_page) → Splits the data into pages.
- get_page(number) → Returns the data for a specific page number.
- Common attributes:
    - page_obj.has_next, page_obj.has_previous
    - page_obj.next_page_number()
    - page_obj.previous_page_number()
    - page_obj.number → Current page number
    - page_obj.paginator.num_pages → Total number of pages

Example: Implementing Pagination in Task Manager

Let's now see how pagination can be added to the **Task Manager** Django project.

1. Import Paginator

In views.py:

```
from django.shortcuts import render, redirect
from django.core.paginator import Paginator


```

2. Update the View

Suppose we have a list of tasks fetched from the database.

We use the Paginator class to divide them into smaller sets.

Code (views.py):

```
from django.shortcuts import render
from django.core.paginator import Paginator
from .models import Task

def home(request):
    task_list = Task.objects.all().order_by('-created_at')
    paginator = Paginator(task_list, 5)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    return render(request, 'tasks/home.html', {'page_obj': page_obj})

```

3. Update Template (home.html)

Add navigation controls below your tasks to move between pages.

Code (home.html):

```
<h1>My Tasks</h1>

{% for task in page_obj %}
<h3>{{ task.title }}</h3>
<p>{{ task.description }}</p>
<p>Created at: {{ task.created_at }}</p>
<p>Status: {% if task.completed %}Completed{% else %}Pending{% endif %}</p>
<hr>
{% endfor %}

<!-- Pagination controls -->
<div>
<span>Page {{ page_obj.number }} of {{ page_obj.paginator.num_pages }}</span><br><br>

{% if page_obj.has_previous %}
    <a href="?page={{ page_obj.previous_page_number }}">Previous</a>
{% endif %}

{% for num in page_obj.paginator.page_range %}
    {% if page_obj.number == num %}
    <strong>{{ num }}</strong>
    {% else %}
    <a href="?page={{ num }}">{{ num }}</a>
    {% endif %}
{% endfor %}

{% if page_obj.has_next %}
    <a href="?page={{ page_obj.next_page_number }}">Next</a>
{% endif %}
</div>
```

4. Testing

- Add several tasks from the Django admin or frontend.
- Access the homepage:

  http://127.0.0.1:8000/
- Only 5 tasks will appear on the first page.
- Click “Next” or a page number to view more.