1. Why do we need Paging in Real-time Applications?

Suppose we have a data source with many records and need to display those records in a view. We can display all the records in a view at once. If we do so, then we will get the following disadvantages.

* Network issue as huge data is traveled over the network.
* Memory Management (Due to heavy data showing in the view, it may cause memory issues; the page may become unresponsive, and sometimes the page becomes very slow).
* Performance (We will get bad performance as it takes more time to travel in the network and more time to display the data on the page).

So, to solve the above problems, we need to go for Paging.

1. What is paging?

Paging, in the context of software development and data management, refers to dividing a large dataset or collection of information into smaller, more manageable segments or “pages.” This technique is commonly used in user interfaces and data processing to handle and display large amounts of data without overwhelming the system or the user.

On the page, we need to display a certain number of records. The next records can be visible with Next and Previous buttons, Page Scrolling, Page Number, or other techniques.

1. Advantages of using Paging:

We will get the following advantages

* **Faster Data Transfer**. This is because fewer data will be traveled over the network.
* **Improve Memory Management**. This is because we are not storing all the data on the page.
* **Better Performance**. As less data is transferred over the network and we display fewer records on a page, it will definitely give you a better performance. Reduces the load on servers and databases, as only a portion of the data is processed and transmitted simultaneously.
* **Enhanced User Experience**: Makes data consumption manageable for users, providing a better, cleaner UI.
* **Scalability**: Facilitates handling growing datasets without significantly impacting performance or user experience.

1. What is the Drawback of Paging?

In a Client-Server Architecture, the number of requests between the client and server increases. In such cases, you may get the data at once from the server (from any API or Databases), store it locally, and then implement the paging on the client side.

1. How to Implement Paging using C#?

Paging in C# using LINQ can be efficiently achieved by combining the LINQ Skip and Take methods. This approach is particularly useful in scenarios where you need to display a large dataset in smaller, more manageable chunks, such as in web and desktop applications where data is presented in a paginated format. In C#, we can implement the paging using the LINQ Skip and Take Method.

* Skip Method: Skip bypasses a specified number of elements in a sequence and then returns the remaining elements.
* Take Method: Take is used to select a specified number of elements from a sequence starting from the beginning.

Together, Skip and Take can be used to implement paging. For example, in a scenario where you have a list of items and want to display them on pages, you can use these methods to retrieve only the subset of items that should be shown on the current page.

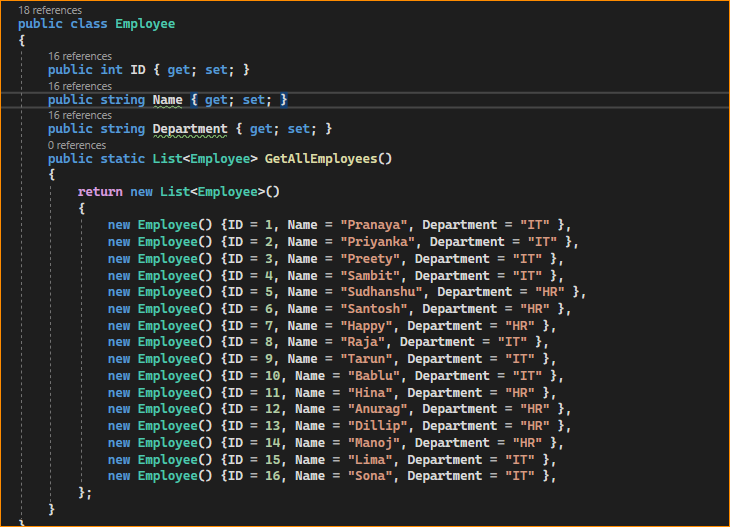
To implement Paging, we need to understand two things: one is Page Number (PN), and the other one is the Number of Records per Page (NRP). Let’s say Page Number = PN and Number Of Records Per Page = NRP, then you need to use the following formula to implement Paging.

**Result = DataSource.Skip((PN – 1) \* NRP).Take(NRP)**

So, basically, from the Data Source, we need to skip (PN – 1) \* NRP number of records from the beginning and return NRP from the data source.

1. Example to Implement Paging using LINQ Skip and Take Method in C#

Let us see how we can implement Paging in C# using LINQ Skip and Take Methods. Here, we are going to use the following Employee class. So, create a class file named Employee.cs and copy and paste the following code. The Employee class has 3 properties, i.e., ID, Name, and Department. The following class also has one method called GetAllEmployees(), which returns a list of all employees and will be our data source.



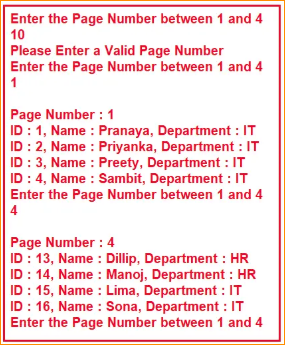
1. Business Requirement:

First, the program should prompt the user to enter the Page Number.

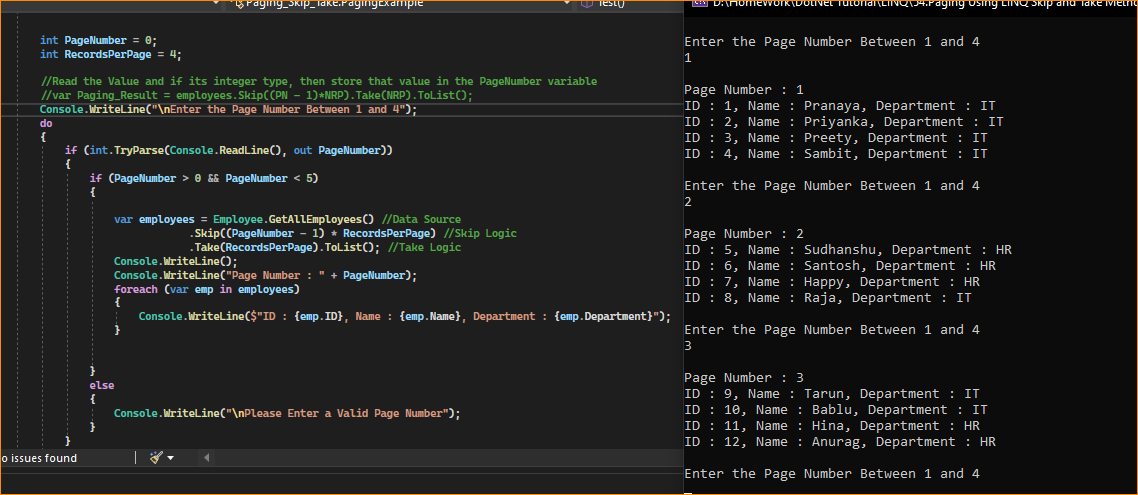
The Page Number must be between 1 and 4. If the user does not enter a Valid Page Number, then the program should prompt the user to enter a Valid Page Number.

The program should display the data once the user enters a valid page number.

For example, the output of the program is something that looks as shown in the below image.



In the following C# Console Application, we implement Paging using the LINQ Skip and Take Methods. The condition in the do-while loop puts the program in an infinite loop. To end the program, close the console window. This problem we will not face when implementing Paging in Web Applications. The following example code is self-explained, so please go through the comment lines for a better understanding.



1. Key Points

* Performance Considerations: For large datasets, especially when dealing with databases (like using LINQ to SQL or Entity Framework), it’s as efficient as Skip and Take translate into SQL queries that handle paging at the database level, retrieving only the necessary subset of data.
* Zero-based Indexing: Remember that Skip is zero-based, so if you use one-based page numbering (common in UIs), you need to adjust the calculation accordingly (as shown in the example).
* Applicability: This pattern is commonly used in web applications to display search results, list items, or any data set where pagination is required for better user experience and performance.

1. Why Do We Need Paging?

* Data Management: In large datasets, loading and displaying all data simultaneously can be inefficient and impractical. Paging allows loading and displaying data in smaller chunks, reducing memory usage and improving performance.
* User Interface: In UI design, paging enhances the user experience. Displaying thousands of rows in a table or list can be overwhelming and hard to navigate. By using pages, the data is broken down into more digestible sections.
* Web and Application Development: Paging is crucial in web development, especially for sites that handle large amounts of data, like e-commerce sites, social media platforms, and search engines. It’s implemented in both front-end and back-end development.
* Database Queries: Paging is used to handle large query results in databases. It allows for querying a specific subset of data at a time (e.g., rows 1-10 in one query, rows 11-20 in another).
* API Design: For RESTful APIs and web services, paging is a common practice in handling requests for large data sets. It prevents server overload and improves the responsiveness of the API.
* Reduced Server Load: When paging is used in web applications, it can reduce the load on the server by fetching and rendering only the data required for the current page. This can lead to better application performance and scalability.
* Consistency: Paging maintains a consistent user experience by presenting data in manageable chunks, regardless of the dataset’s size. Users can navigate through pages with the expectation of similar content and layout.