(T7)比較LinqToObject的Skip、TakeWhile、SkipWhile。實作Paging資料分頁  
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c  
=======================================================================  
(T7)比較LinqToObject的Skip、TakeWhile、SkipWhile。實作Paging資料分頁  
=======================================================================  
0. Summary

-----------

1. New Project

1.1. Create New Project : Sample

-----------

2. Sample : Program.cs  
=======================================================================

0. Summary

1.

Enumerable.Take<TSource>

(IEnumerable<TSource> source, Int32 count)

Reference:

[https://msdn.microsoft.com/en-us/library/bb503062(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/bb503062%28v=vs.110%29.aspx)

Returns a specified number of contiguous鄰近的 elements

from the start of a sequence.

-----------------------------------------

2.

Enumerable.Skip<TSource>

(IEnumerable<TSource> source, Int32 count)

Reference:

[https://msdn.microsoft.com/en-us/library/bb358985(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/bb358985%28v=vs.110%29.aspx)

Bypasses a specified number of elements in a sequence

and then returns the remaining elements.

For the same argument value,

the Skip method returns all of the items

that the Take method would not return.

-----------------------------------------

3.

Enumerable.TakeWhile<TSource>

(IEnumerable<TSource> source, Func<TSource, Int32, Boolean> predicate)

Reference:

[https://msdn.microsoft.com/en-us/library/bb534804(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/bb534804%28v=vs.110%29.aspx)

Returns elements from a sequence as long as a specified condition is true.

3.1. Parameter

3.1.1.

source

Type: System.Collections.Generic.IEnumerable<TSource>

The sequence to return elements from.

3.1.2.

predicate

Type: System.Func<TSource, Int32, Boolean>

A function to test each source element for a condition;

the second parameter of the function represents

the **index** of the source element.

-----------------------------------------

4.

Enumerable.SkipWhile<TSource>

(IEnumerable<TSource> source, Func<TSource, Boolean> predicate)

Reference:

[https://msdn.microsoft.com/en-us/library/bb549075(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/bb549075%28v=vs.110%29.aspx)

Bypasses elements in a sequence as long as a specified condition is true

and then returns the remaining elements.

=============================================

1. New Project

1.1. Create New Project : Sample

File --> New --> Project... -->

Visual C# -->  **Console App** **(.Net Framework)** -->

Name: **Sample**







=============================================

2. Sample : Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using OnLineGame;

namespace ConsoleApp1

{

    class Program

    {

        static void Main(string[] args)

        {

            // 1. =============================================

            // TakeSample

            Console.WriteLine("1. TakeSample(); ======================= ");

            TakeSample();

            // 2. =============================================

            // SkipSample

            Console.WriteLine("2. SkipSample(); ======================= ");

            SkipSample();

            // 3. =============================================

            // TakeWhileSample

            Console.WriteLine("3. TakeWhileSample(); ======================= ");

            TakeWhileSample();

            // 4. =============================================

            // SkipWhileSample

            Console.WriteLine("4. SkipWhileSample(); ======================= ");

            SkipWhileSample();

            // 5. =============================================

            // GamerPaggingSample

            Console.WriteLine("5. GamerPaggingSample(); ======================= ");

            GamerPaggingSample();

            Console.ReadLine();

        }

       // 1. =============================================

        // TakeSample

        //Retrieves only the first 3 string values.

        static void TakeSample()

        {

            // 1.1. TakeSample: Lambda Expression Query ----------------

            Console.WriteLine("1.1. TakeSample: Lambda Expression Query ---------------- ");

            string[] strArr = { "ABCDE", "FIJ", "KLMN", "OP", "QRST", "UV", "WXYZ" };

            IEnumerable<string> strTakeArr = strArr.Take(3);

            foreach (string strTakeArrItem in strTakeArr)

            {

                Console.WriteLine(strTakeArrItem);

            }

            //1.2. TakeSample: SQL Like Query ----------------

            Console.WriteLine("1.2. TakeSample: SQL Like Query ---------------- ");

            IEnumerable<string> strTakeArr2 = (from strArrItem in strArr

                                               select strArrItem).Take(3);

            foreach (string strTakeArr2Item in strTakeArr2)

            {

                Console.WriteLine(strTakeArr2Item);

            }

        }

        // 1.1. TakeSample: Lambda Expression Query ----------------

        // ABCDE

        // FIJ

        // KLMN

        // 1.2. TakeSample: SQL Like Query ----------------

        // ABCDE

        // FIJ

        // KLMN

        // 2. =============================================

        // SkipSample

        //Skip the first 3 string values and take the rest

        static void SkipSample()

        {

            // 2.1. SkipSample: Lambda Expression Query ----------------

            Console.WriteLine("2.1. SkipSample: Lambda Expression Query ----------------");

            string[] strArr = { "ABCDE", "FIJ", "KLMN", "OP", "QRST", "UV", "WXYZ" };

            IEnumerable<string> strSkipArr = strArr.Skip(3);

            foreach (string strSkipArrItem in strSkipArr)

            {

                Console.WriteLine(strSkipArrItem);

            }

            //2.2. SkipSample: SQL Like Query ----------------

            Console.WriteLine("2.2. SkipSample: SQL Like Query ---------------- ");

            IEnumerable<string> strSkipArr2 = (from strArrItem in strArr

                                               select strArrItem).Skip(3);

            foreach (string strSkipArr2Item in strSkipArr2)

            {

                Console.WriteLine(strSkipArr2Item);

            }

        }

        // 2.1. SkipSample: Lambda Expression Query ----------------

        // OP

        // QRST

        // UV

        // WXYZ

        // 2.2. SkipSample: SQL Like Query ----------------

        // OP

        // QRST

        // UV

        // WXYZ

        // 3. =============================================

        // TakeWhileSample

        // As long as the condition is still true, then take it.

        private static void TakeWhileSample()

        {

            // 3.1. TakeWhileSample: Lambda Expression Query ----------------

            Console.WriteLine("3.1. TakeWhileSample: Lambda Expression Query ----------------");

            string[] strArr = { "ABCDE", "FIJ", "KLMN", "OP", "QRST", "UV", "WXYZ" };

            IEnumerable<string> strArrTakeWhileArr = strArr.TakeWhile(s => s.Length > 2);

            foreach (string strArrTakeWhileArrItem in strArrTakeWhileArr)

            {

                Console.WriteLine(strArrTakeWhileArrItem);

            }

            //3.2. TakeWhileSample: SQL Like Query ----------------

            Console.WriteLine("3.2. TakeWhileSample: SQL Like Query ---------------- ");

            IEnumerable<string> strArrTakeWhileArr2 = (from strArrItem in strArr

                                                       select strArrItem).TakeWhile(s => s.Length > 2);

            foreach (string strArrTakeWhileArr2Item in strArrTakeWhileArr2)

            {

                Console.WriteLine(strArrTakeWhileArr2Item);

            }

        }

        // 3.1. TakeWhileSample: Lambda Expression Query ----------------

        // ABCDE

        // FIJ

        // KLMN

        // 3.2. TakeWhileSample: SQL Like Query ----------------

        // ABCDE

        // FIJ

        // KLMN

        // 4. =============================================

        // SkipWhileSample

        // As long as the condition is still true, then skip it.

        static void SkipWhileSample()

        {

            // 4.1. SkipWhileSample: Lambda Expression Query ----------------

            Console.WriteLine("4.1. SkipWhileSample: Lambda Expression Query ----------------");

            string[] strArr = { "ABCDE", "FIJ", "KLMN", "OP", "QRST", "UV", "WXYZ" };

            IEnumerable<string> strArrSkipWhileArr = strArr.SkipWhile(s => s.Length > 2);

            foreach (string strArrSkipWhileArrItem in strArrSkipWhileArr)

            {

                Console.WriteLine(strArrSkipWhileArrItem);

            }

            //4.2. SkipWhileSample: SQL Like Query ----------------

            Console.WriteLine("4.2. SkipWhileSample: SQL Like Query ---------------- ");

            IEnumerable<string> strArrSkipWhileArr2 = (from strArrItem in strArr

                                                       select strArrItem).SkipWhile(s => s.Length > 2);

            foreach (string strArrSkipWhileArr2Item in strArrSkipWhileArr2)

            {

                Console.WriteLine(strArrSkipWhileArr2Item);

            }

        }

        // 4.1. SkipWhileSample: Lambda Expression Query ----------------

        // OP

        // QRST

        // UV

        // WXYZ

        // 4.2. SkipWhileSample: SQL Like Query ----------------

        // OP

        // QRST

        // UV

        // WXYZ

        // 5. =============================================

        // GamerPagging

        static void GamerPaggingSample()

        {

            int numberOfGamers = 27;

            int pageSize = 10;

            int pageNumber = 0;

            GamerPagging(numberOfGamers, pageSize, pageNumber);

            pageNumber = 1;

            GamerPagging(numberOfGamers, pageSize, pageNumber);

            pageNumber = 2;

            GamerPagging(numberOfGamers, pageSize, pageNumber);

            pageNumber = 3;

            GamerPagging(numberOfGamers, pageSize, pageNumber);

            pageNumber = 4;

            GamerPagging(numberOfGamers, pageSize, pageNumber);

        }

        //Create {numberOfGamers} Gamers List

        //Then Create Pagging

        //Each page have {pageSize} Gamers.

        //Set to {pageNumber} Page.

        //E.g.

        //Create 43 Gamers List

        //Then Create Pagging

        //Each page have 10 Gamers,

        //this will create 5 pages

        //Set to Page 3.

        //This will show Gamers21 to Gamers30

        static void GamerPagging(int numberOfGamers, int pageSize, int pageNumber)

        {

            List<Gamer> gamerList = GamerHelper.GetSampleGamers(numberOfGamers);

            //int pageNumber = 1;

            //int pageSize = 10;

            int numberOfPages =

                Convert.ToInt32(

                    Math.Ceiling((double)gamerList.Count / pageSize));

            if (pageNumber >= 1 && pageNumber <= numberOfPages)

            {

                IEnumerable<Gamer> gamersInPage =

                    gamerList.Skip((pageNumber - 1) \* pageSize)

                    .Take(pageSize);

                Console.WriteLine($"Page Number:{pageNumber}");

                foreach (Gamer gamersInPageItem in gamersInPage)

                {

                    Console.WriteLine(gamersInPageItem);

                }

                Console.WriteLine();

            }

            else

            {

                Console.WriteLine($"Invalid Page Number. Page number must be an integer between 1 and {numberOfPages}\r\n");

            }

        }

        // Invalid Page Number. Page number must be an integer between 1 and 3

        // Page Number:1

        // Id==1,Name==Name1

        // Id==2,Name==Name2

        // Id==3,Name==Name3

        // Id==4,Name==Name4

        // Id==5,Name==Name5

        // Id==6,Name==Name6

        // Id==7,Name==Name7

        // Id==8,Name==Name8

        // Id==9,Name==Name9

        // Id==10,Name==Name10

        // Page Number:2

        // Id==11,Name==Name11

        // Id==12,Name==Name12

        // Id==13,Name==Name13

        // Id==14,Name==Name14

        // Id==15,Name==Name15

        // Id==16,Name==Name16

        // Id==17,Name==Name17

        // Id==18,Name==Name18

        // Id==19,Name==Name19

        // Id==20,Name==Name20

        // Page Number:3

        // Id==21,Name==Name21

        // Id==22,Name==Name22

        // Id==23,Name==Name23

        // Id==24,Name==Name24

        // Id==25,Name==Name25

        // Id==26,Name==Name26

        // Id==27,Name==Name27

        // Invalid Page Number. Page number must be an integer between 1 and 3

    }

}

namespace OnLineGame

{

    public class Gamer

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public override string ToString()

        {

            return $"Id=={Id},Name=={Name}";

        }

    }

    public class GamerHelper

    {

        // Create a List<Gamer> which contains numberOfGamers gamers.

        public static List<Gamer> GetSampleGamers(int numberOfGamers)

        {

            //int numberOfGamers = 43;

            List<Gamer> gamerList = new List<Gamer>();

            for (int i = 1; i <= numberOfGamers; i++)

            {

                gamerList.Add(new Gamer { Id = i, Name = $"Name{i}" });

            }

            return gamerList;

        }

    }

}

Text

Description automatically generated





