(T32)討論 YieldReturn 的 Filter、Total
CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92
==========
(T32)討論 YieldReturn 的 Filter、Total
(T32-1)討論 YieldReturn 的 Filter
(T32-2)討論 YieldReturn 的 Total
==========
1. Introduction
---
2. OnlineGame Solution
2.1. OnlineGame Solution
2.2. OnlineGame.Library - Class Library (.Net Framework)
2.3. OnlineGame.ConsoleApp - ConsoleApp (.NET Framework)
---
3. OnlineGame Solution
3.1. OnlineGame.Library/S01IntCollection.cs
3.2. OnlineGame.ConsoleApp/Program.cs
==========

1. Introduction

1.
Yield Return
1.1.
Reference:
http://limitedcode.blogspot.com/2014/07/c-yeild.html
https://www.kenneth-truyers.net/2016/05/12/yield-return-in-c/
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/yield
https://docs.microsoft.com/zh-tw/dotnet/csharp/language-reference/keywords/yield
https://youtu.be/4fju3xcm21M
https://youtu.be/F7L9seU_mak

1.2.
"yield return" can do custom stateful iteration over the collection.
"yield return" will return a collection. E.g. IEnumerable<T>
-->
我們針對某個 collection 可以做客製化的 stateful iteration
通常回傳一個 collection。E.g. IEnumerable<T>

1.3.
Normal iteration WITHOUT yield return :
when I traverse each element in the loop,
if I encounter an element that meets the criteria,
We usually create a temp collection
and then add the matching elements to that temp collection.
Then we will return that temp collection.
-->
如果沒有 yield return，我們通常如下作法。

我們先做一個空的 temp collection
當我們 iterate 每個 element 的時候，如果哪個 element 有符合條件，
就加入我們的 temp collection。
當 loop 結束後，就直接 return 這個 temp collection。

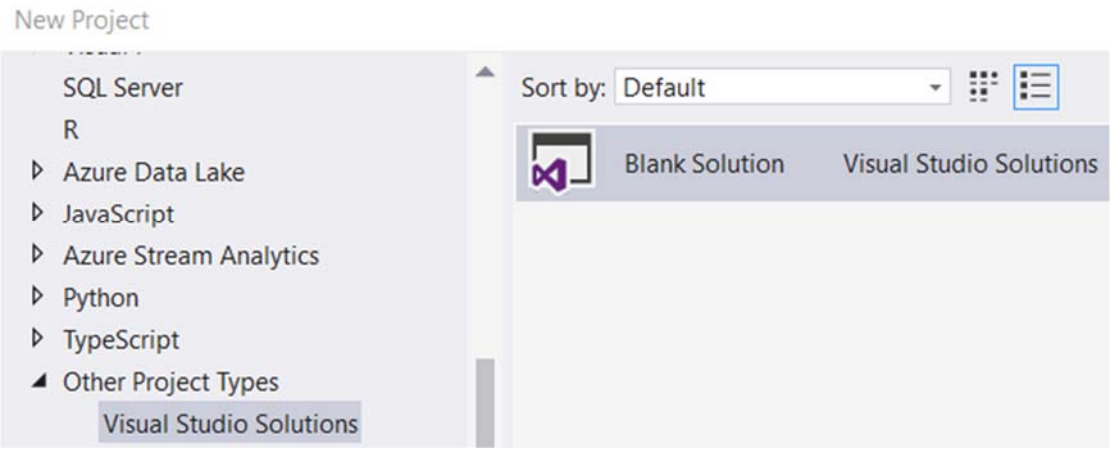
1.4.
iteration WITH yield return :
when I traverse each element in the loop,
if I encounter an element that meets the criteria,
we return that ONE element back to the previous layer which is its caller.
when previous layer has done what it needs to do,
then jump back the loop and then get the next element.
有 yield return 的時候，我們可以如下做法。
我們"不必"做一個空的 temp collection
當我們 iterate 每個 element 的時候，如果哪個 element 有符合條件，
我們直接"yield return"到上一層，也就是他的 caller。
然後它的 caller 解決他該做的事情的時候，
它又 jump 回原本的 loop 然後再去看看下一個 element。
-->
We repeatedly to check if the next element that meets the criteria,
then repeatedly return that ONE element back to the previous layer which is its caller.
and then repeatedly jump back to the loop to get the next element
over and over again until the loop ends.
我們"重複地"去找下一個有符合條件的 element，
然後"重複地"回傳有符合條件的 element 到上一層也就是它的 caller。
然後"重複的"jump 回原本的 loop 再繼續看下一個 element 直到該 loop 結束。

2. OnlineGame Solution

2.1. OnlineGame Solution

Open Visual Studio, I am currently using VS2017
If you don't have it, you may follow the instruction here to download.
http://ithandyguytutorial.blogspot.com/2017/10/ch00install-visual-studio-2017-offline.html

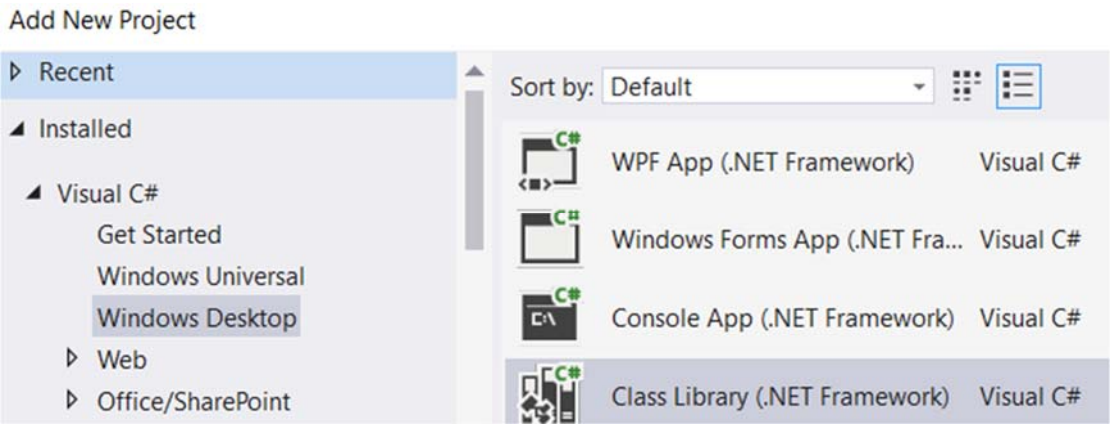
In **VS2017Community**
New Project --> Other Project Type --> Visual Studio Solutions
--> **Blank Solution**
Name:
OnlineGame

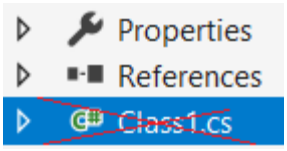


2.2. OnlineGame.Library - Class Library (.Net Framework)

In **VS2017Community**
Solution Name --> Add --> New Project
--> Visual C# --> **Class Library (.Net Framework)**
-->
Name: **OnlineGame.Library**

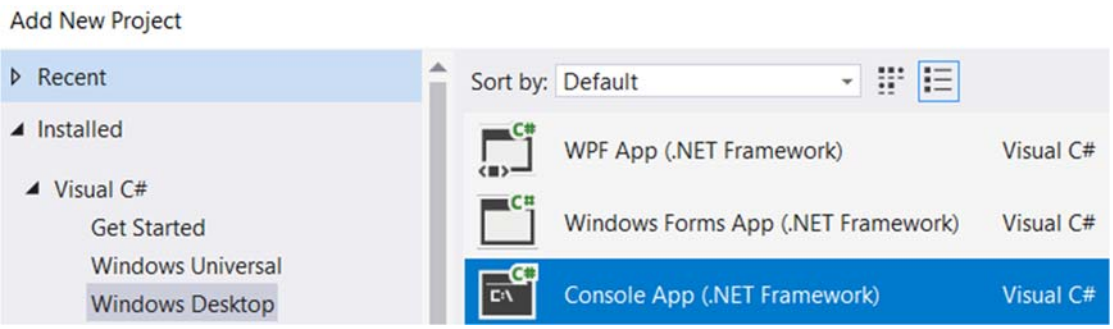
-->
Delete **Class1.cs**



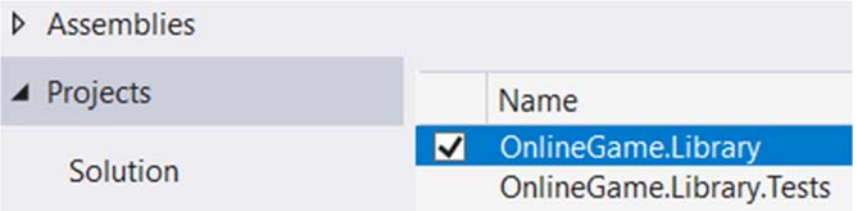


2.3. OnlineGame.ConsoleApp - ConsoleApp (.NET Framework)

In **VS2017Community**
Solution Name --> Add --> New Project
--> Visual C# --> **ConsoleApp (.NET Framework)**
-->
Name: **OnlineGame.ConsoleApp**



Please add the following as the reference
-->
OnlineGame.Library



3. OnlineGame Solution

3.1. OnlineGame.Library/S01IntCollection.cs

```
//Please set the break point at the line which contains "return" or "yield return".

using System.Collections.Generic;
namespace OnlineGame.Library
{
    public class S01IntCollection
    {
        public static List<int> intList = new List<int> { 100, 99, 98, 97, 96, 95 };
        //=======================================
        //1. GetMoreThanOrEqual
        public static IEnumerable<int> GetMoreThanOrEqual(int number)
        {
            var tempList = new List<int>();
```

```
            foreach (int i in intList)
            {
                if (i >= number)
                    tempList.Add(i);
            }
            return tempList;
        }
        public static IEnumerable<int> GetMoreThanOrEqualYied(int number)
        {
            foreach (int i in intList)
            {
                if (i >= number)
                    yield return i;
            }
        }
        //=======================================
        //2. Total
        public static int GetTotal()
        {
            int total = 0;
            foreach (var i in intList)
                total += i;
            return total;
        }
        public static IEnumerable<int> GetRunningTotal()
        {
            var tempList = new List<int>();
            int total = 0;
            foreach (var i in intList)
            {
                total += i;
                tempList.Add(total);
            }
            return tempList;
        }
        public static IEnumerable<int> GetRunningTotalYield()
        {
            //Inspect "total" variable and find out
            //the value is always preserved from the last run.
            int total = 0;
            foreach (var i in intList)
            {
                total += i;
                yield return total;
            }
        }
    }
}
/*
1.
Yield Return
1.1.
Reference:
http://limitedcode.blogspot.com/2014/07/c-yeild.html
https://www.kenneth-truyers.net/2016/05/12/yield-return-in-c/
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/yield
https://docs.microsoft.com/zh-tw/dotnet/csharp/language-reference/keywords/yield
https://youtu.be/4fju3xcm21M
https://youtu.be/F7L9seU_mak
------------------------
1.2.
```

```
"yield return" can do custom stateful iteration over the collection.
"yield return" will return a collection. E.g. IEnumerable<T>
-->
我們針對某個 collection 可以做客製化的 stateful iteration
通常回傳一個 collection。E.g. IEnumerable<T>
------------------------
1.3.
Normal iteration WITHOUT yield return :
when I traverse each element in the loop,
if I encounter an element that meets the criteria,
We usually create a temp collection
and then add the matching elements to that temp collection.
Then we will return that temp collection.
-->
如果沒有 yield return，我們通常如下作法。
我們先做一個空的 temp collection
當我們 iterate 每個 element 的時候，如果哪個 element 有符合條件，
就加入我們的 temp collection。
當 loop 結束後，就直接 return 這個 temp collection。
------------------------
1.4.
iteration WITH yield return :
when I traverse each element in the loop,
if I encounter an element that meets the criteria,
we return that ONE element back to the previous layer which is its caller.
when previous layer has done what it needs to do,
then jump back the loop and then get the next element.
有 yield return 的時候，我們可以如下做法。
我們"不必"做一個空的 temp collection
當我們 iterate 每個 element 的時候，如果哪個 element 有符合條件，
我們直接"yield return"到上一層，也就是他的 caller。
然後它的 caller 解決他該做的事情的時候，
它又 jump 回原本的 loop 然後再去看看下一個 element。
-->
We repeatedly to check if the next element that meets the criteria,
then repeatedly return that ONE element back to the previous layer which is its caller.
and then repeatedly jump back to the loop to get the next element
over and over again until the loop ends.
我們"重複地"去找下一個有符合條件的 element，
然後"重複地"回傳有符合條件的 element 到上一層也就是它的 caller。
然後"重複的"jump 回原本的 loop 再繼續看下一個 element 直到該 loop 結束。
*/

-->
```

```
14      public static IEnumerable<int> GetMoreThanOrEqual(int number)
15      {
16          var tempList = new List<int>();
17          foreach (int i in intList)
18          {
19              if (i >= number)
20                  tempList.Add(i);
21          }
22          return tempList;
23      }
```

```
26  public static IEnumerable<int> GetMoreThanOrEqualYied(int number)
27  {
28      foreach (int i in intList)
29      {
30          if (i >= number)
31              yield return i;
32      }
33  }
```

```
36  //=========================================
37  //2. Total

38

39  public static int GetTotal()
40  {
41      int total = 0;
42      foreach (var i in intList)
43          total += i;
44      return total;
45  }
46

47

48  public static IEnumerable<int> GetRunningTotal()
49  {
50      var tempList = new List<int>();
51      int total = 0;
52      foreach (var i in intList)
53      {
54          total += i;
55          tempList.Add(total);
56      }
57      return tempList;
58  }
```

```
62  public static IEnumerable<int> GetRunningTotalYield()
63  {
64      //Inspect "total" variable and find out
65      //the value is always preserved from the last run.
66      int total = 0;
67      foreach (var i in intList)
68      {
69          total += i;
70          yield return total;
71      }
72  }
```

3.2. OnlineGame.ConsoleApp/Program.cs

```
//Please set the break point at the line which contains "Console.WriteLine" or
"S01IntCollection.GetTotal()".

using System;
using OnlineGame.Library;
namespace OnlineGame.ConsoleApp
{
    class Program
    {
```

```
        static void Main(string[] args)
        {
            //=====================================
            //1. GetMoreThanOrEqual
            //1.1.
            //Normal IEnumerable<int>
            Console.WriteLine("1.1. Normal IEnumerable<int> ============");
            foreach (int intA in S01IntCollection.GetMoreThanOrEqual(98))
            {
                Console.WriteLine(intA);
            }
            //1.2.
            //IEnumerable<int> with "yield return"
            Console.WriteLine("1.2. IEnumerable<int> with yield return ===========");
            foreach (int intB in S01IntCollection.GetMoreThanOrEqualYied(98))
            {
                Console.WriteLine(intB);
            }
            //=====================================
            //2. Total
            //2.1.
            //Normal IEnumerable<int>
            Console.WriteLine("2.1. GetTotal ============");
            int total = S01IntCollection.GetTotal();
            Console.WriteLine(total);
            Console.WriteLine("2.2. GetRunningTotal ============");
            foreach (int intA in S01IntCollection.GetRunningTotal())
            {
                Console.WriteLine(intA);
            }
            Console.WriteLine("2.3. GetRunningTotalYield ============");
            foreach (int intB in S01IntCollection.GetRunningTotalYield())
            {
                Console.WriteLine(intB);
            }
            Console.ReadLine();
        }
    }
}
```

-->

```
12  {
13      //=====================================
14      //1. GetMoreThanOrEqual
15
16      //1.1.
17      //Normal IEnumerable<int>
18      Console.WriteLine("1.1. Normal IEnumerable<int> ============");
19      foreach (int intA in S01IntCollection.GetMoreThanOrEqual(98))
20      {
21          Console.WriteLine(intA);
22      }
23
24      //1.2.
25      //IEnumerable<int> with "yield return"
26      Console.WriteLine("1.2. IEnumerable<int> with yield return ===========");
27      foreach (int intB in S01IntCollection.GetMoreThanOrEqualYied(98))
28      {
29          Console.WriteLine(intB);
30      }
```

-->

```
32          //=====================================
33          //2. Total
34
35          //2.1.
36          //Normal IEnumerable<int>
37          Console.WriteLine("2.1. GetTotal ============");
38          int total = S01IntCollection.GetTotal();
39          Console.WriteLine(total);
40
41          Console.WriteLine("2.2. GetRunningTotal ============");
42          foreach (int intA in S01IntCollection.GetRunningTotal())
43          {
44              Console.WriteLine(intA);
45          }
46
47          Console.WriteLine("2.3. GetRunningTotalYield ============");
48          foreach (int intB in S01IntCollection.GetRunningTotalYield())
49          {
50              Console.WriteLine(intB);
51          }
```

-->

1.1. Normal IEnumerable<int> ============
100
99
98
1.2. IEnumerable<int> with yield return ===========
100
99
98
2.1. GetTotal ============
585
2.2. GetRunningTotal ============
100
199
297
394
490
585
2.3. GetRunningTotalYield ============
100
199
297
394
490
585