(T8)比較 LazyLoading 延遲執行(Select、Where、Take、Skip)、EagerLoading 立刻執行(aggregate、ToList)
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c
==========
(T8)比較 LazyLoading 延遲執行(Select、Where、Take、Skip)、EagerLoading 立刻執行(aggregate、ToList)
==========

==========

0. Summary

Based on the behavior of query execution, Linq can be classified into 2 categories.
1. Deferred Operators/Lazy Operators/Lazy Loading use deferred execution.
E.g. select, where, Take, Skip ...
2. Immediate Operators/Greedy Operators/Eager Loading use immediate execution.
E.g. count, average, min, max, ToList ...

==========

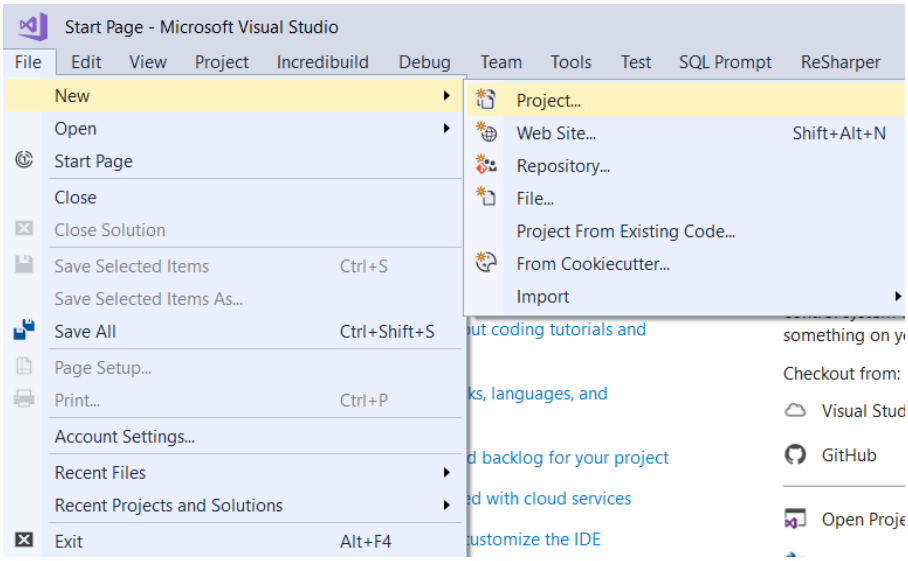
1. New Project

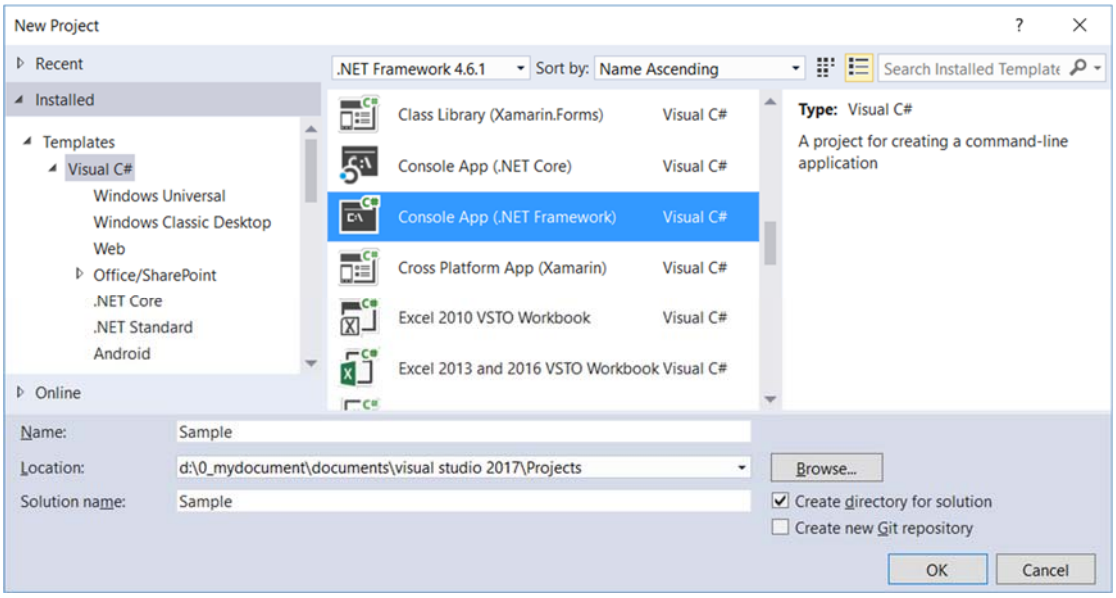
1.1. Create New Project : Sample

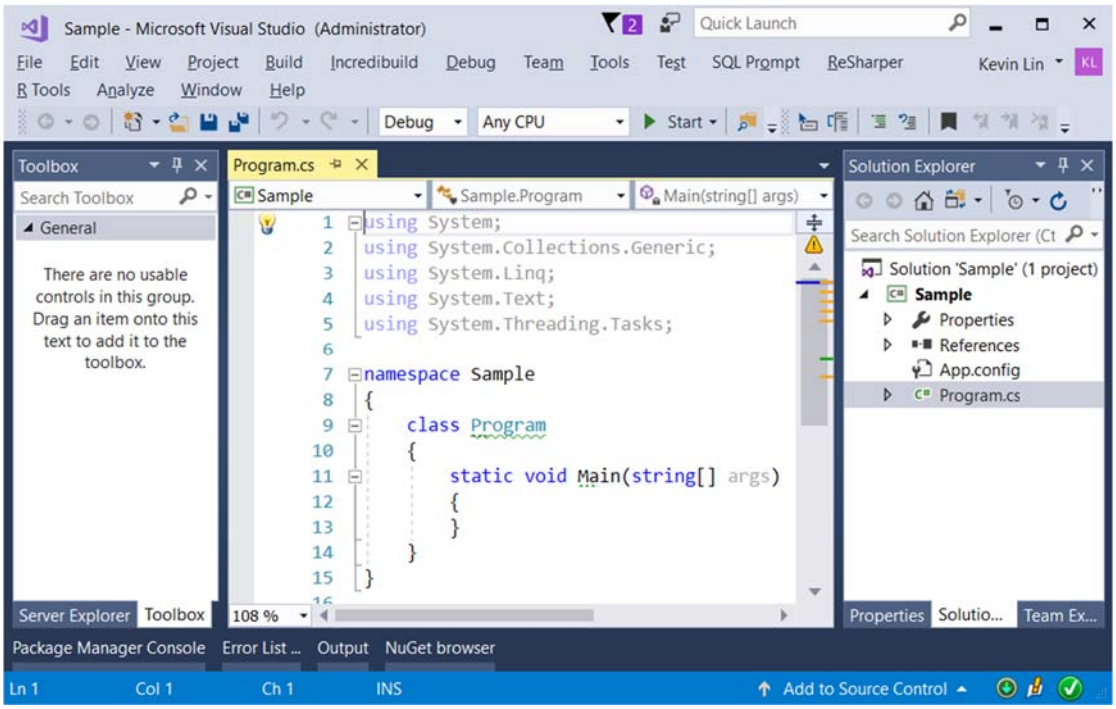
File --> New --> Project... -->

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

Name: **Sample**


Start Page - Microsoft Visual Studio
File Edit View Project Incredibuild Debug Team Tools Test SQL Prompt ReSharper
New
Open
Start Page
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project...
Web Site... Shift+Alt+N
Repository...
File...
Project From Existing Code...
From Cookiecutter...
Import
Checkout from:
Visual Stud
GitHub
Open Proje


New Project
Recent
Installed
Templates
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Online
.NET Framework 4.6.1
Sort by: Name Ascending
Search Installed Templat
Class Library (Xamarin.Forms) Visual C#
Console App (.NET Core) Visual C#
Console App (.NET Framework) Visual C#
Cross Platform App (Xamarin) Visual C#
Excel 2010 VSTO Workbook Visual C#
Excel 2013 and 2016 VSTO Workbook Visual C#
Type: Visual C#
A project for creating a command-line application
Name: Sample
Location: d:\0_mydocument\documents\visual studio 2017\Projects
Solution name: Sample
Browse...
Create directory for solution
Create new Git repository
OK
Cancel


Sample - Microsoft Visual Studio (Administrator)
Quick Launch
File Edit View Project Build Incredibuild Debug Team Tools Test SQL Prompt ReSharper
R Tools Analyze Window Help
Kevin Lin
Debug
Any CPU
Start
Toolbox
Search Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Program.cs
Sample
Sample.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Sample
{
class Program
{
static void Main(string[] args)
{
}
}
}
Solution Explorer
Search Solution Explorer (Ct
Solution 'Sample' (1 project)
Sample
Properties
References
App.config
Program.cs
Server Explorer
Toolbox
108 %
Properties
Solutio...
Team Ex...
Package Manager Console
Error List ...
Output
NuGet browser
Ln 1
Col 1
Ch 1
INS
Add to Source Control

==========

2. Sample : Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using OnLineGame;
namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            //1. ============================================
            //LinqDeferredExecutionExample();
            Console.WriteLine("1. LinqDeferredExecutionExample(); ============= ");
             LinqDeferredExecutionExample();
            //2. ============================================
            //LinqImmediateExecutionExample();
            Console.WriteLine("2. LinqImmediateExecutionExample(); ============= ");
             LinqImmediateExecutionExample();
            //3. ============================================
            //LinqImmediateExecutionExample2();
            Console.WriteLine("3. LinqImmediateExecutionExample2(); ============= ");
             LinqImmediateExecutionExample2();
            Console.ReadLine();
        }

        //1. ============================================
        //LinqDeferredExecutionExample();
        static void LinqDeferredExecutionExample()
        {
            List<Gamer> gamersList = GamerHelper.GetSampleGamers();
            //1.
            //Deferred /Lazy Operators use deferred execution.
            //E.g.select, where, Take, Skip...
            //LINQ Query has been defined but not executed yet at this point.
            //If the Linq query has been executed at this point,
            //the result should not include
            //new Gamer { Id = 4, Name = "Name4", Score = 100 }
            IEnumerable<Gamer> gamerScoreEqualTo100 =
              from gamer in gamersList
              where gamer.Score == 100
              select gamer;
            // Add a new Gamer object with Score=100 to the source list.
             gamersList.Add(new Gamer { Id = 4, Name = "Name4", Score = 100 });
            //The above LINQ Query has been actually executed here
            //when using foreach loop.
            //the result includes
            //new Gamer { Id = 4, Name = "Name4", Score = 100 }
            foreach (var gamerScoreEqualTo100Item in gamerScoreEqualTo100)
            {
                Console.WriteLine(gamerScoreEqualTo100Item);
```

```
        }
    }
    //Id==1,Name==Name1,Score==100
    //Id==2,Name==Name2,Score==100
    //Id==4,Name==Name4,Score==100


//2. ============================================
    //LinqImmediateExecutionExample();
    private static void LinqImmediateExecutionExample()
    {
        List<Gamer> gamersList = GamerHelper.GetSampleGamers();
        //2.
        //Immediate/Greedy Operators use immediate execution.
        //E.g.count, average, min, max, ToList...
        //ToList() which is a Immediate/Greedy Operator,
        //so LINQ Query has been executed immediately at this point.
        //the LINQ Query is executed immediately at this point.
        //the result does not include
        //new Gamer { Id = 4, Name = "Name4", Score = 100 }
        List<Gamer> gamerScoreEqualTo100 =
            (from gamer in gamersList
             where gamer.Score == 100
             select gamer).ToList();
        //Add a new Gamer object with Score=100 to the source list.
        //This will not affect on the result
        //because the Linq query has been already executed.
        gamersList.Add(new Gamer { Id = 4, Name = "Name4", Score = 100 });
        //The above LINQ Query has been actually executed
        //when using .ToList()
        //the result will not include
        //new Gamer { Id = 4, Name = "Name4", Score = 100 }
        foreach (var gamerScoreEqualTo100Item in gamerScoreEqualTo100)
        {
            Console.WriteLine(gamerScoreEqualTo100Item);
        }
    }
    //Id==1,Name==Name1,Score==100
    //Id==2,Name==Name2,Score==100


//3. ============================================
    //LinqImmediateExecutionExample2();
    static void LinqImmediateExecutionExample2()
    {
        List<Gamer> gamersList = GamerHelper.GetSampleGamers();
        //2.
        //Immediate/Greedy Operators use immediate execution.
        //E.g.count, average, min, max, ToList...
        //Count() which is a Immediate/Greedy Operator,
        //so LINQ Query has been executed immediately at this point.
        //the LINQ Query is executed immediately at this point.
        //the result does not include
        //new Gamer { Id = 4, Name = "Name4", Score = 100 }
```

```
            int gamerScoreEqualTo100Count =     //2
                 (from gamer in gamersList
                  where gamer.Score == 100
                  select gamer).Count();
            //Add a new Gamer object with Score=100 to the source list.
            //This will not affect on the result
            //because the Linq query has been already executed.
             gamersList.Add(new Gamer { Id = 4, Name = "Name4", Score = 100 });
            //The above LINQ Query has been actually executed
            //when using .Count()
            //the result will not include
            //new Gamer { Id = 4, Name = "Name4", Score = 100 }
            Console.WriteLine($"gamerScoreEqualTo100Count=={gamerScoreEqualTo100Count}");
        }
        //gamerScoreEqualTo100Count==2
    }
}
namespace OnLineGame
{
    public class Gamer
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public int Score { get; set; }
        public override string ToString()
        {
            return $"Id=={Id},Name=={Name},Score=={Score}";
        }
    }
    public class GamerHelper
    {
        public static List<Gamer> GetSampleGamers()
        {
            return new List<Gamer>
            {
                new Gamer { Id = 1, Name = "Name1", Score =100 },
                new Gamer { Id = 2, Name = "Name2", Score =100 },
                new Gamer { Id = 3, Name = "Name3", Score =200 }
            };
        }
        // Create a List<Gamer> which contains numberOfGamers gamers.
        public static List<Gamer> GetSampleGamers(int numberOfGamers)
        {
            //int numberOfGamers = 10;
            List<Gamer> gamerList = new List<Gamer>();
            for (int i = 1; i <= numberOfGamers; i++)
            {
                Random rnd = new Random();
                int rndScore = rnd.Next(1000, 6000); // creates a number between 1000 and 6000
                 gamerList.Add(new Gamer { Id = i, Name = $"Name{i}", Score = rndScore });
            }
            return gamerList;
        }
    }
}
```

```
1. LinqDeferredExecutionExample(); ============
Id==1,Name==Name1,Score==100
Id==2,Name==Name2,Score==100
Id==4,Name==Name4,Score==100
2. LinqImmediateExecutionExample(); ============
Id==1,Name==Name1,Score==100
Id==2,Name==Name2,Score==100
3. LinqImmediateExecutionExample2(); ============
gamerScoreEqualTo100Count==2
```