(T9)討論 AccessModifiers。比較 Public、Protected、Private、Internal、ProtectedInternal
CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92

==

(T9)討論 AccessModifiers。比較 Public、Protected、Private、Internal、ProtectedInternal

==

==

0. Summary

1.
Access modifiers
Reference:
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/accessibility-levels
1.1.
Access modifiers are keywords used to specify the declared accessibility of a member or a type.
In this tutorial, we only discuss the following Accessibility Levels.

- **private** : Access is limited to the containing type. **(Default to Type Members)**
- **public** : Access is not restricted.
- **protected** : Access is limited to the containing class or types derived from the containing class.
- **internal** : Access is limited to the current assembly. **(Default to Types)**
- **protected internal** : Access is limited to the current assembly or types derived from the containing class.

1.2.
In general,
Types can use **public** and **internal** ,
and Types includes Class, Struct, Enums, Interface, Dlegate are belonged.
1.3.
Type Members can use **private**, **public**, **protected**, **internal**, **protected internal**
and Type Members includes **fields**, **properties**, **constructors**, and **methods**.

==

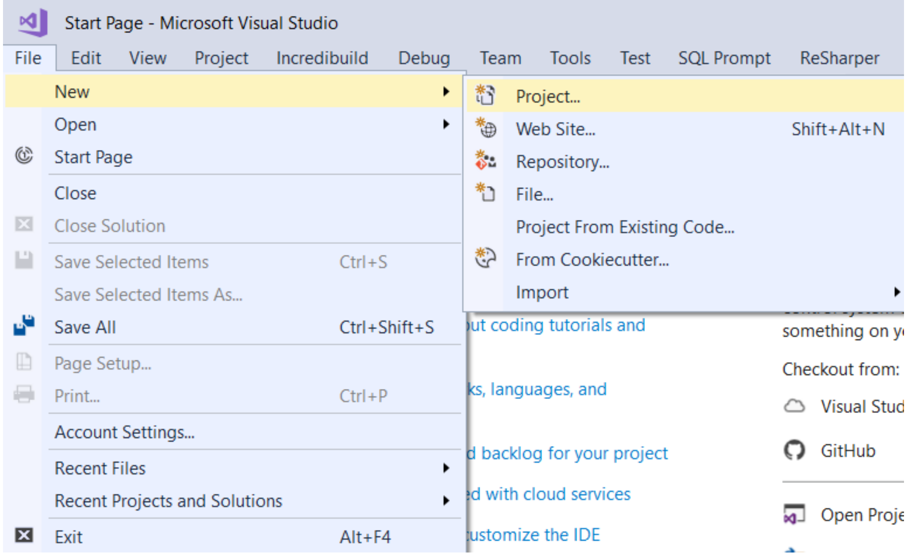
1. New Project

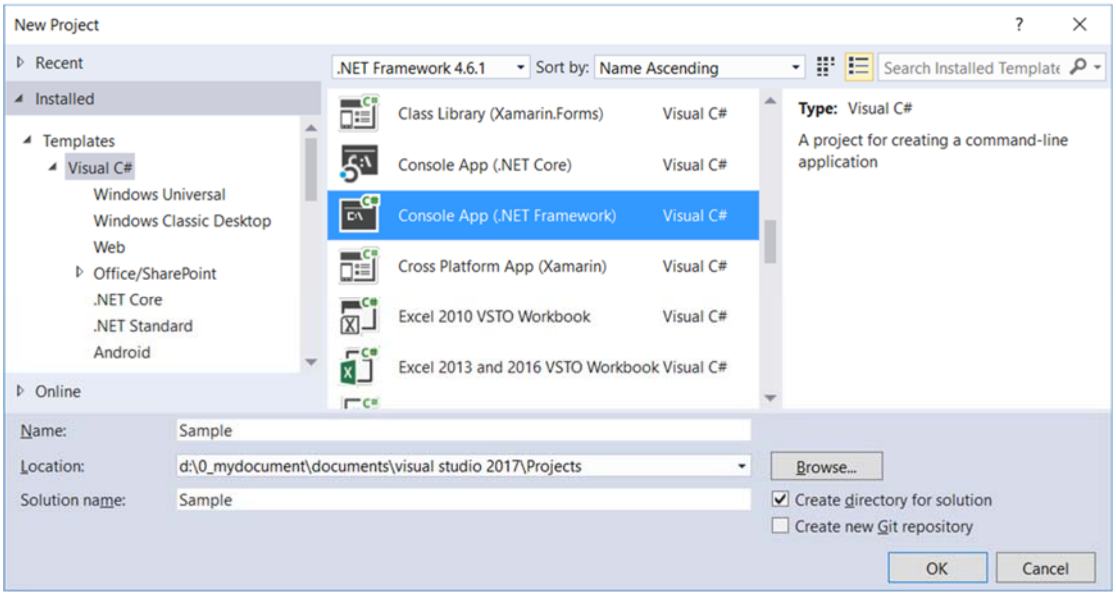
1.1. Create New Project

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

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

Name: **Sample**




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 Template
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#
Browse...
OK
Cancel

Type: Visual C#

A project for creating a command-line application

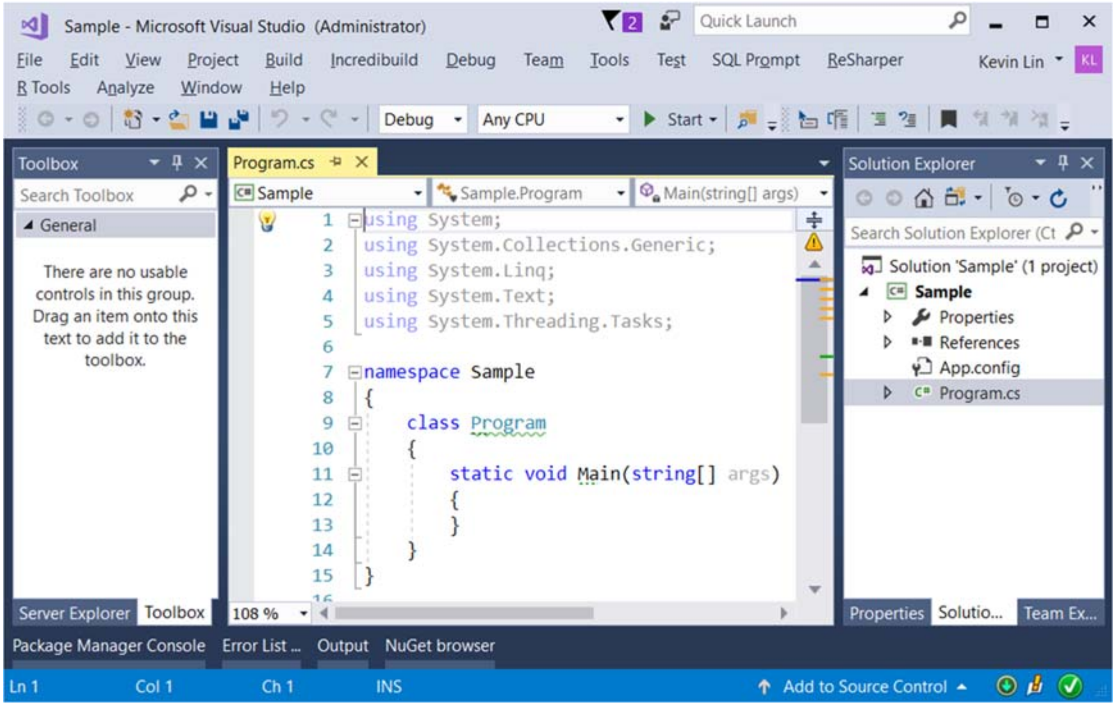
Name: Sample

Location: d:\0_mydocument\documents\visual studio 2017\Projects

Solution name: Sample

☑ Create directory for solution

☐ Create new Git repository



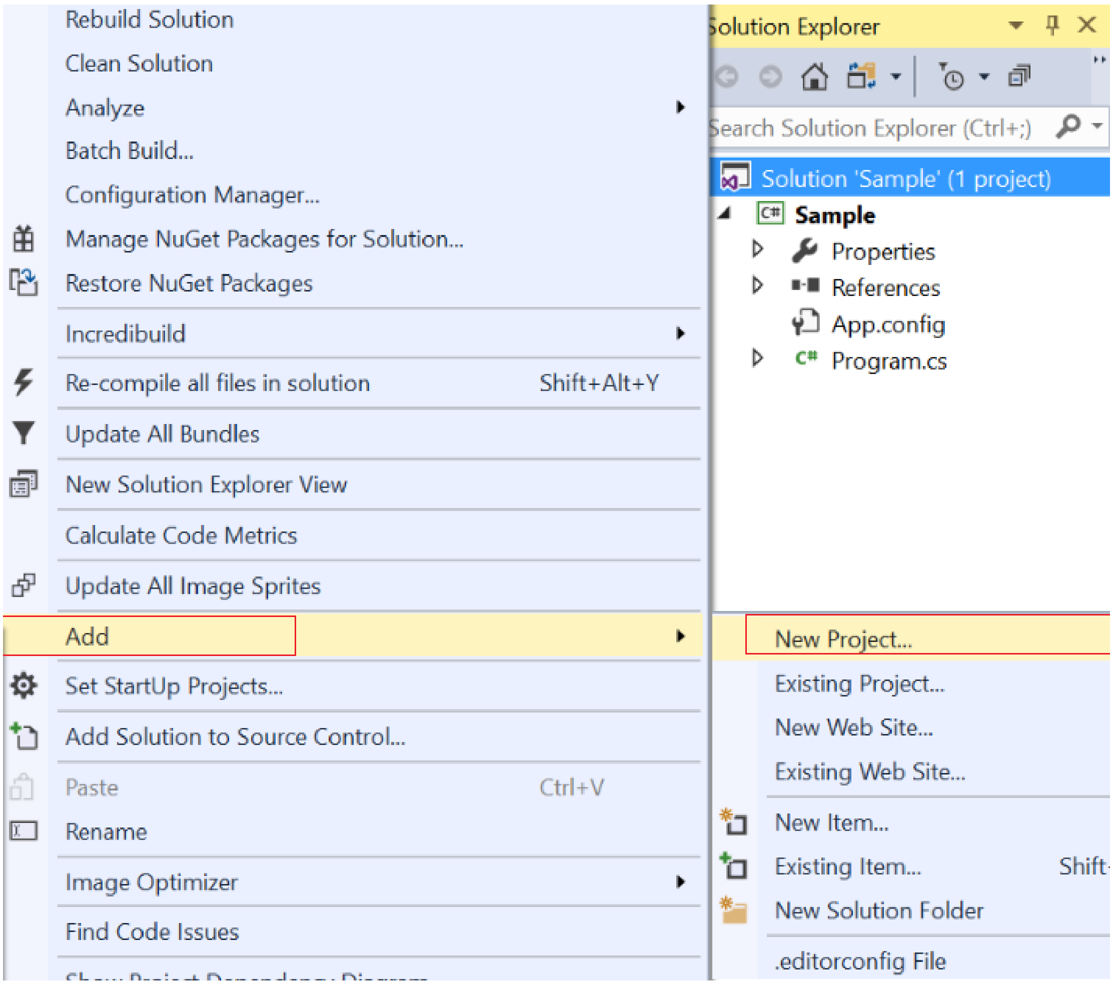
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

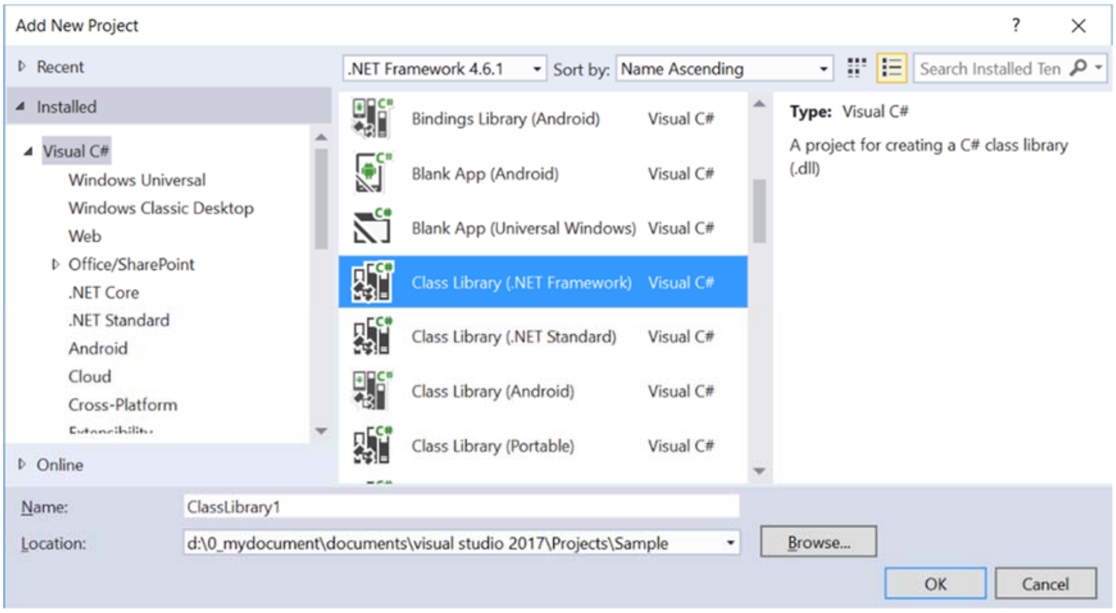
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

1.2. Add New Project

Solution Name --> Right Click --> Add --> New Project --> Class Library (.Net Framework)
-->
Project Name :
ClassLibrary1




Add New Project
Recent
Installed
Visual C#
Windows Universal
Windows Classic Desktop
Web
Office/SharePoint
.NET Core
.NET Standard
Android
Cloud
Cross-Platform
Online
Search Installed Ten
Bindings Library (Android) Visual C#
Blank App (Android) Visual C#
Blank App (Universal Windows) Visual C#
Class Library (.NET Framework) Visual C#
Class Library (.NET Standard) Visual C#
Class Library (Android) Visual C#
Class Library (Portable) Visual C#
Browse...
OK
Cancel

.NET Framework 4.6.1

Sort by: Name Ascending

Type: Visual C#

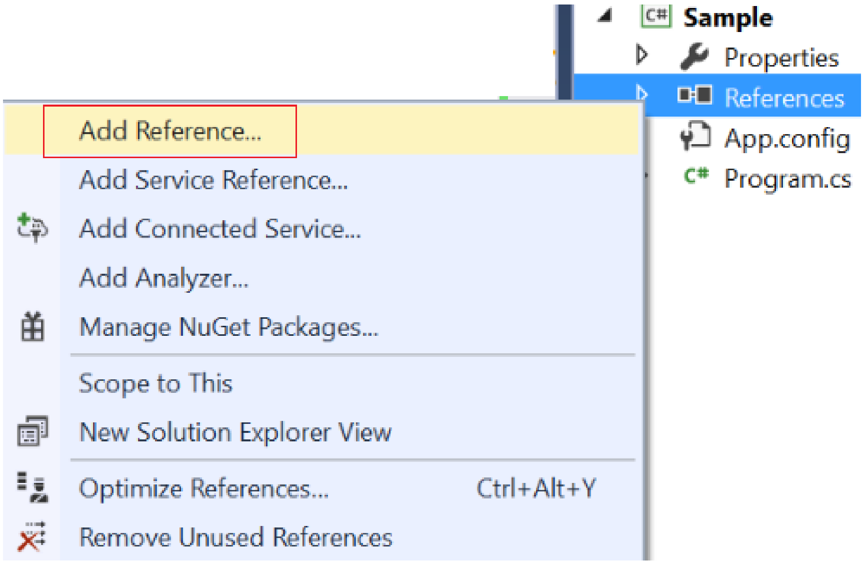
A project for creating a C# class library (.dll)

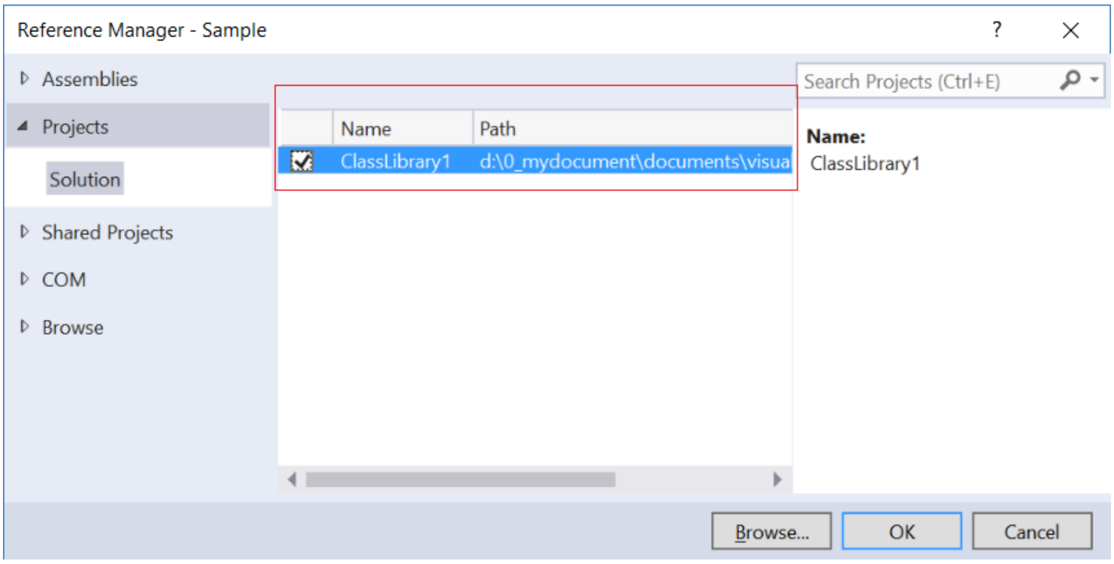
Name: ClassLibrary1

Location: d:\0_mydocument\documents\visual studio 2017\Projects\Sample

1.3. Add Reference

Project Name --> References --> Add Reference
--> Select the reference you want to add.





	Name	Path
☑	ClassLibrary1	d:\0_mydocument\documents\visua

Name:
ClassLibrary1

===

2. Access modifiers

1.
Access modifiers
Reference:
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/accessibility-levels
1.1.
Access modifiers are keywords used to specify the declared accessibility of a member or a type.
In this tutorial, we only discuss the following Accessibility Levels.

- **private** : Access is limited to the containing type. **(Default to Type Members)**
- **public** : Access is not restricted.
- **protected** : Access is limited to the containing class or types derived from the containing class.
- **internal** : Access is limited to the current assembly. **(Default to Types)**
- **protected internal** : Access is limited to the current assembly or types derived from the containing class.

1.2.
In general,
Types can use **public** and **internal** ,

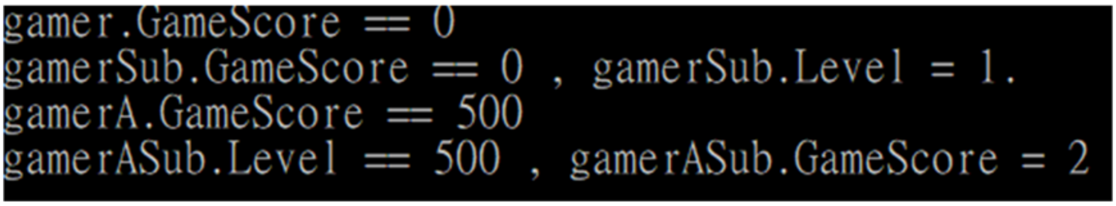
and **Types** includes **Class**, **Struct**, **Enums**, **Interface**, **Dlegate**.
1.3.
Type Members can use **private**, **public**, **protected**, **internal**, **protected internal**
and **Type Members** includes **fields**, **properties**, **constructors**, and **methods**.

2.1. ClassLibrary1/GamerA.cs

```
namespace OnLineGameA
{
    public class GamerA
    {
        // private field means only available in current class.
        private int _gameScore = 500;
        // protected field means available in current class and its sub class.
        protected internal int _level = 2;
        // public property means available every where.
        public int GameScore
        {
            get
            {
                return _gameScore;
            }
            set
            {
                _gameScore = value;
            }
        }
    }
    //Internal means only available in current assembly.
    public class GamerASub : GamerA
    {
        // public property means available every where.
        public int Level
        {
            get
            {
                return _level;
                // Sub Class can access the protected field from base class.
            }
            set
            {
                _level = value;
            }
        }
        // Protected internal method means only available in current assembly, and its sub class.
        protected internal int GetGameScore()
        {
            ////return base._gameScore;
            //  base._gameScore is private, thus, not available in its sub class.
            return GameScore;
        }
    }
}
```

2.2. Sample/Program.cs

```
using System;
using OnlineGame;
using OnLineGameA;
namespace Sample
{
    class Program
    {
        static void Main(string[] args)
        {
            Gamer gamer = new Gamer();
            //int gamer_gameScore = gamer._gameScore; // Error, Not available.
            //int gamer_Level = gamer._level; // Error, Not available.
            Console.WriteLine("gamer.GameScore == {0}", gamer.GameScore);
            GamerSub gamerSub = new GamerSub();
             gamerSub.GetGameScore();
            Console.WriteLine("gamerSub.GameScore == {0} , gamerSub.Level = {1}.", gamerSub.GameScore, 
gamerSub.Level);
            GamerA gamerA = new GamerA();
            //int gamerA_gameScore = gamerA._gameScore; // Error, Not available.
            //int gamerA_Level = gamerA._level; // Error, Not available.
            Console.WriteLine("gamerA.GameScore == {0}", gamerA.GameScore);
            GamerASub gamerASub = new GamerASub();
            Console.WriteLine("gamerASub.Level == {0} , gamerASub.GameScore = {1}", gamerASub.GameScore, 
gamerASub.Level);
            // gamerASub.GetGameScore();   // Error, Not available.
            Console.ReadLine();
        }
    }
}
namespace OnlineGame
{
    public class Gamer
    {
        // private field means only available in current class.
        private int _gameScore = 0;
        // protected field means available in current class and its sub class.
        protected int _level = 1;
        // public property means available every where.
        public int GameScore
        {
            get
            {
                return _gameScore;
            }
            set
            {
                 _gameScore = value;
            }
        }
    }
    //Internal means only available in current assembly.
    internal class GamerSub : Gamer
    {
        // public property means available every where.
        public int Level
        {
            get
            {
                return _level;
                // Sub Class can access the protected field from base class.
            }
            set
            {
                _level = value;
            }
        }
        public int GetGameScore()
        {
            ////return base._gameScore; //Error, Not available.
            //  base._gameScore is private, thus, not available in its sub class.
            return GameScore;
        }
    }
}
/*
1.
Access modifiers
Reference:
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/accessibility-levels
1.1.
Access modifiers are keywords used to specify the declared accessibility of a member or a type.
In this tutorial, we only discuss the following Accessibility Levels.
* private : Access is limited to the containing type. (Default to Type Members)
* public : Access is not restricted.
* protected : Access is limited to the containing class or types derived from the containing class.
* internal : Access is limited to the current assembly. (Default to Types)
* protected internal : Access is limited to the current assembly or types derived from the containing
class.
1.2.
In general,
Types can use public and internal ,
and Types includes Class, Struct, Enums, Interface, Dlegate.
1.3.
Type Members can use private, public, protected, internal, protected internal
and Type Members includes fields, properties, constructors, and methods.
*/
```



```
gamer.GameScore == 0
gamerSub.GameScore == 0 , gamerSub.Level = 1.
gamerA.GameScore == 500
gamerASub.Level == 500 , gamerASub.GameScore = 2
```