(T15)CustomAttribute(客製化屬性)、Reflection(反射)  
CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92  
=======================================================================  
(T15)CustomAttribute(客製化屬性)、Reflection(反射)  
=======================================================================  
0. Summary

-----------

1. New Project

1.1. Create New Project

-----------

2. Program.cs  
=======================================================================

0. Summary

\*這個tutorial討論客製化attribute，應用方面是，搭配Reflection 和 XML 後，可以讓你寫的code可以用客製化，比如說你的XML明確規定  指讀取啥啥attribute的class，透過reflection動態讀取。

1.

Attribute

1.1.

Syntax:

//[AttributeUsage(AttributeTargets.Class | AttributeTargets.Property | AttributeTargets.Method | ...etc.)]

//public class ClassNameAttribute : System.Attribute

1.2.

Attribute is a Class which extend System.Attribute and

provide declarative information which is queried at runtime using reflection.

The suffix of Attribute is "Attribute".

[AttributeUsage(AttributeTargets.All)] is default usage setting

that means it can apply to every where.

//[AttributeUsage(AttributeTargets.Class | AttributeTargets.Property | AttributeTargets.Method)]

it means this attribute can only apply to Class, Property, Method

2.

Pre-defined attributes in the .NET framework.

2.1.

//[Obsolete]

Marks types and type members outdated.

2.1.1.

The compiler issues a warning to types or type members with [Obsolete].

2.1.2.

The compiler issues a warning with message to types or type members with [Obsolete("Message")]

2.1.3.

The compiler issues a compiler error with message to types or type members with [Obsolete("Message", true)]

2.2.

//[WebMethod]

expose a method as an XML Web service method

2.3.

//[Serializable]

Indicates that a class can be serialized

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

1. New Project

1.1. Create New Project

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

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

Name: **Sample**







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

2. Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using OnLineGame;

namespace Sample

{

    class Program

    {

        static void Main(string[] args)

        {

            //3. ---------------------------------------

            Console.WriteLine("AttributeLinqSample(); ===========================");

            AttributeLinqSample();

            //4. ---------------------------------------

            Console.WriteLine("ObsoleteSample(); ===========================");

            ObsoleteSample();

            Console.ReadLine();

        }

        //3. ---------------------------------------

        static void AttributeLinqSample()

        {

            // Get all the Types which apply GamerB1Attribute

            IEnumerable<Type> types = from t in Assembly.GetExecutingAssembly().GetTypes()

                                      where t.GetCustomAttributes<GamerB1Attribute>().Any()

                                      select t;

            foreach (Type t in types)

            {

                // TypeObject.FullName is NameSpace.ClassName

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

                Console.WriteLine(t.FullName);

                Console.WriteLine("properties --------------------------");

                foreach (PropertyInfo propertyInfo in t.GetProperties())

                {

                    // "PropertyType PropertyName"

                    Console.WriteLine($"{propertyInfo.PropertyType} {propertyInfo.Name}");

                }

                // Get all the PropertyInfo which apply GamerB1Attribute

                IEnumerable<PropertyInfo> gamerB1AttributePropertyInfo = from pInfo in t.GetProperties()

                                                                         where pInfo.GetCustomAttributes<GamerB1Attribute>().Any()

                                                                         select pInfo;

                Console.WriteLine("gamerB1Attribute Properties --------------------------");

                foreach (PropertyInfo propertyInfo in gamerB1AttributePropertyInfo)

                {

                    // "PropertyType PropertyName"

                    Console.WriteLine($"{propertyInfo.PropertyType} {propertyInfo.Name}");

                }

                Console.WriteLine("Methods --------------------------");

                foreach (MethodInfo methodInfo in t.GetMethods())

                {

                    // "ReturnType MethodName"

                    Console.WriteLine($"{methodInfo.ReturnType.Name} {methodInfo.Name}");

                }

                IEnumerable<MethodInfo> gamerB1AttributeMethodInfo = from mInfo in t.GetMethods()

                                                                     where mInfo.GetCustomAttributes<GamerB1Attribute>().Any()

                                                                     select mInfo;

                Console.WriteLine("gamerB1Attribute Methods --------------------------");

                foreach (MethodInfo methodInfo in gamerB1AttributeMethodInfo)

                {

                    // "ReturnType MethodName"

                    Console.WriteLine($"{methodInfo.ReturnType.Name} {methodInfo.Name}");

                }

            }

        }

        //4. ---------------------------------------

        static void ObsoleteSample()

        {

            Console.WriteLine($"GameScoreCaculator.Sum(2, 3)  :  {GameScoreCaculator.Sum(2, 3)}");

            Console.WriteLine($"GameScoreCaculator.Sum(1, 2, 3)  :  {GameScoreCaculator.Sum(1, 2, 3)}");

            List<int> intList = new List<int>{1,2,3,4};

            GameScoreCaculator.Sum(intList);

            Console.WriteLine($" GameScoreCaculator.Sum(intList)  :  { GameScoreCaculator.Sum(intList)}");

        }

    }

}

namespace OnLineGame

{

    //1. ---------------------------------------

    //[AttributeUsage(AttributeTargets.All)] is default usage setting

    //that means it can apply to every where.

    public class GamerA1Attribute : Attribute

    {

    }

    [GamerA1]

    public class GamerA

    {

        // Properties ----------------------------

        [GamerA1]

        public int GameScore { get; set; }

        [GamerA1]

        public string Name { get; set; }

        // Methods ----------------------------

        [GamerA1]

        public override string ToString()

        {

            return $"GameScore : {GameScore} ; Name : {Name}";

        }

        public void NoAttributeMethod()

        {

        }

    }

    //2. ---------------------------------------

    // it means this attribute can only apply to Class, Property, Method

    [AttributeUsage(AttributeTargets.Class | AttributeTargets.Property | AttributeTargets.Method)]

    public class GamerB1Attribute : Attribute

    {

        public string Name { get; set; }

        public double Version { get; set; }

    }

    [GamerB1(Name = "GamerB", Version = 1.0)]

    public class GamerB

    {

        // Properties ----------------------------

        [GamerB1(Name = "GamerBMethod", Version = 1.0)]

        public int GameScore { get; set; }

        [GamerA1]

        public string Name { get; set; }

        // Methods ----------------------------

        [GamerB1]

        public override string ToString()

        {

            return $"GameScore : {GameScore} ; Name : {Name}";

        }

        public void NoAttributeMethod()

        {

        }

    }

    [GamerB1(Name = "GamerB2", Version = 1.0)]

    public class GamerB2

    {

        // Properties ----------------------------

        [GamerB1(Name = "GamerB2Method", Version = 1.0)]

        public int GameScore { get; set; }

        [GamerA1]

        public string Name { get; set; }

        // Methods ----------------------------

        [GamerB1]

        public override string ToString()

        {

            return $"GameScore : {GameScore} ; Name : {Name}";

        }

    }

   //3. ---------------------------------------

    public class GamerCNoAttribute

    {

    }

    //4. ---------------------------------------

    public class GameScoreCaculator

    {

        [Obsolete]

        public static int Sum(int i1, int i2)

        {

            return i1 + i2;

        }

        [Obsolete("Use Sum(List<int> intList) instead.")]

        //[Obsolete("Use Sum(List<int> intList) instead.", true)]

        public static int Sum(int i1, int i2, int i3)

        {

            return i1 + i2 + i3;

        }

        public static int Sum(List<int> intList)

        {

            int Sum = 0;

            foreach (int i in intList)

            {

                Sum += i;

            }

            return Sum;

        }

        //2.1.

        ////[Obsolete]

        //Marks types and type members outdated.

        //2.1.1.

        //The compiler issues a warning to types or type members with[Obsolete].

        //2.1.2.

        //The compiler issues a warning with message to

        //types or type members with[Obsolete("Message")]

        //2.1.3.

        //The compiler issues a compiler error with message to

        //types or type members with [Obsolete("Message", true)]

    }

}

/\*

1.

Attribute

1.1.

Syntax:

//[AttributeUsage(AttributeTargets.Class | AttributeTargets.Property | AttributeTargets.Method | ...etc.)]

//public class ClassNameAttribute : System.Attribute

1.2.

Attribute is a Class which extend System.Attribute and

provide declarative information which is queried at runtime using reflection.

The suffix of Attribute is "Attribute".

[AttributeUsage(AttributeTargets.All)] is default usage setting

that means it can apply to every where.

//[AttributeUsage(AttributeTargets.Class | AttributeTargets.Property | AttributeTargets.Method)]

it means this attribute can only apply to Class, Property, Method

2.

Pre-defined attributes in the .NET framework.

2.1.

//[Obsolete]

Marks types and type members outdated.

2.1.1.

The compiler issues a warning to types or type members with [Obsolete].

2.1.2.

The compiler issues a warning with message to

types or type members with [Obsolete("Message")]

2.1.3.

The compiler issues a compiler error with message to

types or type members with [Obsolete("Message", true)]

2.2.

//[WebMethod]

expose a method as an XML Web service method

2.3.

//[Serializable]

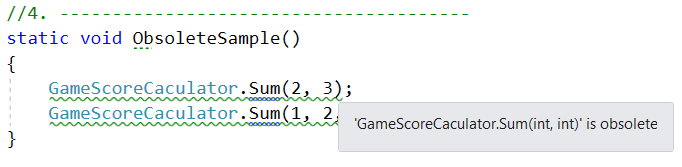
Indicates that a class can be serialized

\*/



Graphical user interface, application

Description automatically generated



Graphical user interface, text, application

Description automatically generated

