(T17)比較Indexer、Properties。比較Dictionary、List  
CourseGUID: 29f1196a-1950-41a4-b9c1-dd13a9e92d92  
=======================================================================  
(T17)比較Indexer、Properties。比較Dictionary、List  
=======================================================================  
0. Summary

-----------

1. New Project

1.1. Create New Project : Sample

1.2. Create New Project : SampleWebForm

-----------

2. Sample : Program.cs

-----------

3. SampleWebForm : WebForm1.aspx.cs  
=======================================================================

0. Summary

1.

Indexers V.S. Properties

Reference:

<http://www.c-sharpcorner.com/uploadfile/puranindia/indexers-in-C-Sharp/>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/indexers/>

<https://social.msdn.microsoft.com/Forums/vstudio/en-US/c02abcfd-3e3e-484e-9a7c-d2dcb32123e5/how-to-invoke-multi-parameter-indexer-in-c-?forum=netfxbcl>

Indexers are used on group of instances of a class or struct just like arrays.

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

1.1.

Indexer is Parameterized property

which is created by "this" keyword and

identified by signature.

However,

properties don't need "this" keyword and

are identified by their names.

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

1.2.

Indexers are accessed using indexes and

are instance member, so can't be static.

However,

Properties are accessed by their names and

can be static or instance members.

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

1.3.

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

1.3.1.

Indexer is similar to property and it has get and set accessor for the [ ] operator.

Indexer can not use ref and out parameter and

it must have at least one parameter in [ ] operator.

Mmulti-parameter indexers are considered poor design style.

A indexer "get" accessor use the parameter list in [ ] operator as the indexer,

and return value of this indexer.

A indexer "set" accessor use the parameter list in [ ] operator as the indexer,

set the value to the indexer.

Indexer can be overloaded by using different signatures.

E.g.

//public class F

//{

//    public object Indexer[string a, string b] { ... }

//}

//

//var f = new F();

//f["hello", "world"] = new object();

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

1.3.2.

Property has "get" and "set" accessor.

A property "get" accessor has no parameters and

return value of the property.

A "set" accessor of a property contains the implicit value parameter and

set the value to the property.

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

1. New Project

1.1. Create New Project : Sample

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

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

Name: **Sample**

Graphical user interface, application, email

Description automatically generated



Graphical user interface, text, application, Excel

Description automatically generated

1.2. Create New Project : SampleWebForm

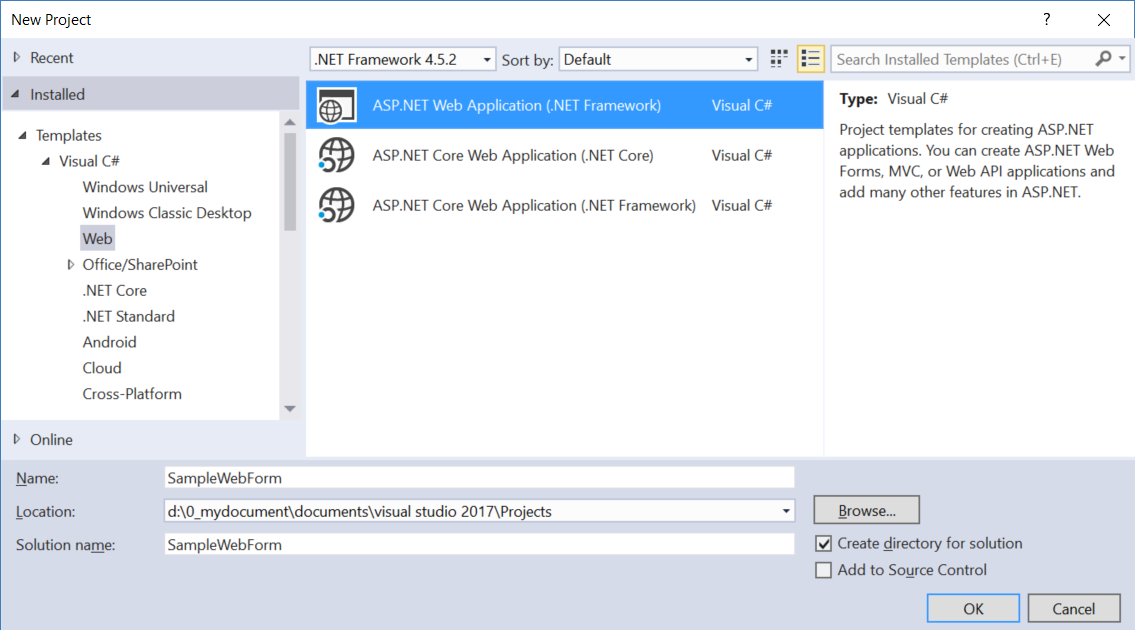
File --> New --> Project... -->

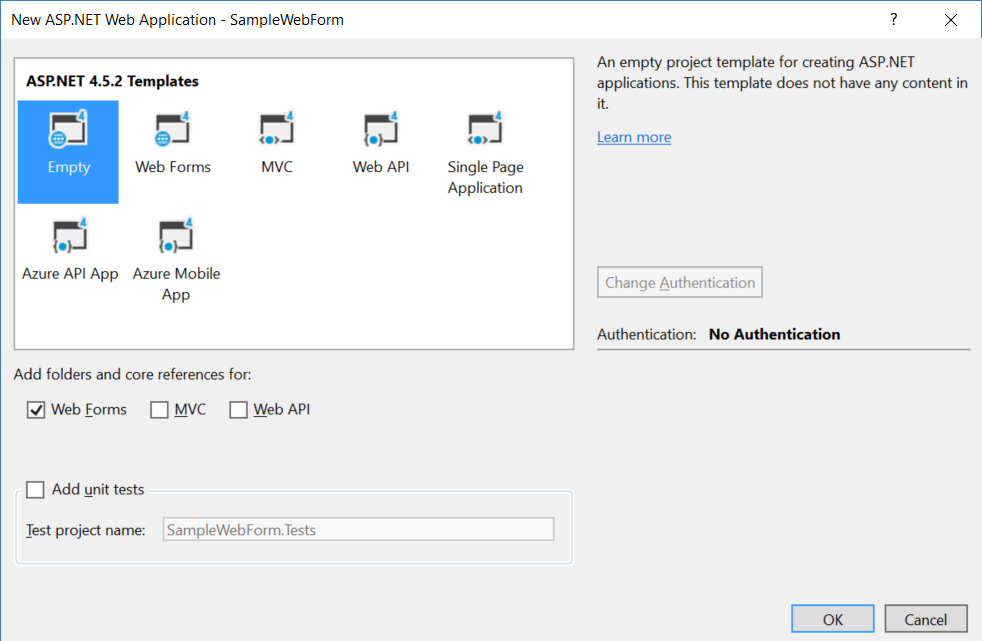
Visual C# -->  Web  -->  **ASP.NET Web Application** **(.Net Framework)** -->

Name: **SampleWebForm**

--> Select "Empty" --> Select "Web Forms"







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

2. Sample : Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using OnLineGame;

namespace Sample

{

    class Program

    {

        static void Main(string[] args)

        {

            // 0. ----------------------------------------------------

            Console.WriteLine("0. DefaultValue() =============================");

            DefaultValue();

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

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

            ListSample();

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

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

            DictionarySample();

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

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

            ListOfStringIndexerSample();

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

            ListOfGamerIndexerSample();

            Console.WriteLine("5. ListOfGamerIndexerSample2() =============================");

            ListOfGamerIndexerSample2();

            Console.WriteLine("6. Dictionary\_String\_Gamer\_IndexerSample() =============================");

            Dictionary\_String\_Gamer\_IndexerSample();

            Console.ReadLine();

        }

        // 0. ---------------------------------------------

        static void DefaultValue()

        {

            int intDefault = default(int);

            Console.WriteLine($"default(int) == {intDefault}");

            string stringDefault = default(string) == null ?

                "NULL" : default(string);

            Console.WriteLine($"default(string) == {stringDefault}");

        }

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

        static void ListSample()

        {

            List<string> listMagicSpell = new List<string>

            {

                "Spell01", "Spell02", "Spell03", "Spell04"

            };

            Console.WriteLine("1.1. listMagicSpell ------------------ ");

            Console.Write("listMagicSpell == {");

            foreach (string magicSpell in listMagicSpell)

            {

                if (magicSpell == listMagicSpell.Last())

                {

                    Console.WriteLine($"\"{magicSpell}\"" + " }");

                }

                else

                {

                    Console.Write($"\"{magicSpell}\", ");

                }

            }

            //1.1.listMagicSpell------------------

            //listMagicSpell == { "Spell01", "Spell02", "Spell03", "Spell04" }

            Console.WriteLine("1.2. ListObject.FindIndex() ---------------------- ");

            int indexOfSpell03 = listMagicSpell.FindIndex(a => a == "Spell03");

            Console.WriteLine($"listMagicSpell.FindIndex(a => a == \"Spell03\")  :  " +

                             $"{indexOfSpell03}");

            //1.2.ListObject.FindIndex()----------------------

            //listMagicSpell.FindIndex(a => a == "Spell03")  :  2

            int indexOfSpell0 = listMagicSpell.FindIndex(a => a == "Spell0");

            Console.WriteLine($"listMagicSpell.FindIndex(a => a == \"Spell0\")  :  " +

                             $"{indexOfSpell0}");

            //listMagicSpell.FindIndex(a => a == "Spell0")  :  -1

            //FindIndex() return -1 if the index can not be found.

            Console.WriteLine("1.3. ListObject.Contains() ---------------------- ");

            bool containsSpell3 = listMagicSpell.Contains("Spell03");

            Console.WriteLine($"listMagicSpell.Contains(\"Spell03\")  :  " +

                              $"{containsSpell3}");

            //1.3.ListObject.Contains()----------------------

            //listMagicSpell.Contains("Spell03")  :  True

            string firstContainSpell03 = listMagicSpell.FirstOrDefault(x => x.Contains("Spell03"));

            Console.WriteLine($"listMagicSpell.FirstOrDefault(x => x.Contains(\"Spell03\"))  :  " +

                              $"{firstContainSpell03}");

            //listMagicSpell.FirstOrDefault(x => x.Contains("Spell03"))  :  Spell03

            string firstContainSpell0 = listMagicSpell.FirstOrDefault(x => x.Contains("Spell0"));

            Console.WriteLine($"listMagicSpell.FirstOrDefault(x => x.Contains(\"Spell0\"))  :  " +

                              $"{firstContainSpell0}");

            //listMagicSpell.FirstOrDefault(x => x.Contains("Spell0"))  :  Spell01

            Console.WriteLine("1.4. ListObject.Equals() ---------------------- ");

            string firstEqualsSpell02 = listMagicSpell.FirstOrDefault(x => x.Equals("Spell02"));

            Console.WriteLine($"listMagicSpell.FirstOrDefault(x => x.Equals(\"Spell02\"))  :  " +

                              $"{firstEqualsSpell02}");

            //1.4.ListObject.Equals()----------------------

            //listMagicSpell.FirstOrDefault(x => x.Equals("Spell02"))  :  Spell02

            string firstEqualsSpell0 =

                listMagicSpell.FirstOrDefault(x => x.Equals("Spell0")) == null ?

                "Not Found" :

                listMagicSpell.FirstOrDefault(x => x.Equals("Spell0"));

            Console.WriteLine($"listMagicSpell.FirstOrDefault(x => x.Equals(\"Spell0\"))  :  " +

                              $"{firstEqualsSpell0}");

            //listMagicSpell.FirstOrDefault(x => x.Equals("Spell0"))  :  Not Found

            Console.WriteLine("1.5. ListObject[index] ---------------------- ");

            string magicSpell0 = listMagicSpell[0]; // get index 0 value string.

            Console.WriteLine($"listMagicSpell[0]  ==  {listMagicSpell[0]}");

            Console.WriteLine($"listMagicSpell[1]  ==  {listMagicSpell[1]}");

            Console.WriteLine($"listMagicSpell[2]  ==  {listMagicSpell[2]}");

            Console.WriteLine($"listMagicSpell[3]  ==  {listMagicSpell[3]}");

            //Console.WriteLine($"listMagicSpell[4]  ==  {listMagicSpell[4]}");   // Error! Throw Example.

            //1.5.ListObject[index]----------------------

            //listMagicSpell[0] == Spell01

            //listMagicSpell[1] == Spell02

            //listMagicSpell[2] == Spell03

            //listMagicSpell[3] == Spell04

        }

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

        static void DictionarySample()

        {

            Dictionary<string, Gamer> dictionaryGamers = new Dictionary<string, Gamer>();

            dictionaryGamers.Add("Name01", new Gamer { Id = 1, Name = "Name01", Email = "[1@1.com](mailto:1@1.com)" });

            dictionaryGamers.Add("Name02", new Gamer { Id = 2, Name = "Name02", Email = "[2@2.com](mailto:2@2.com)" });

            dictionaryGamers.Add("Name03", new Gamer { Id = 3, Name = "Name03", Email = "[3@3.com](mailto:3@3.com)" });

            dictionaryGamers.Add("Name04", new Gamer { Id = 4, Name = "Name04", Email = "[4@4.com](mailto:4@4.com)" });

            Console.WriteLine("2.1. dictionaryGamers --------------------");

            KeyValuePair<string, Gamer> name01KeyValue =

                dictionaryGamers.FirstOrDefault(x => x.Key == "Name01");

            Gamer gamer1 = dictionaryGamers.FirstOrDefault(x => x.Key == "Name01").Value;

            KeyValuePair<string, Gamer> name02KeyValue =

                dictionaryGamers.FirstOrDefault(x => x.Key == "Name02");

            KeyValuePair<string, Gamer> name03KeyValue =

                dictionaryGamers.FirstOrDefault(x => x.Key == "Name03");

            KeyValuePair<string, Gamer> name04KeyValue =

                dictionaryGamers.FirstOrDefault(x => x.Key == "Name04");

            Console.WriteLine($"name01KeyValue.Key == {name01KeyValue.Key}  ;  " +

                              $"gamer1.ToString() == {name01KeyValue.Value.ToString()}");

            Console.WriteLine($"name02KeyValue.Key == {name02KeyValue.Key}  ;  " +

                              $"gamer2.ToString() == {name02KeyValue.Value.ToString()}");

            Console.WriteLine($"name03KeyValue.Key == {name03KeyValue.Key}  ;  " +

                              $"gamer3.ToString() == {name03KeyValue.Value}");

            Console.WriteLine($"name04KeyValue.Key == {name04KeyValue.Key}  ;  " +

                              $"gamer4.ToString() == {name04KeyValue.Value}");

            //2.1.dictionaryGamers--------------------

            //name01KeyValue.Key == Name01; gamer1.ToString() == Id == 1; Name == Name01; Email: [1@1.com](mailto:1@1.com)

            //name02KeyValue.Key == Name02; gamer2.ToString() == Id == 2; Name == Name02; Email: [2@2.com](mailto:2@2.com)

            //name03KeyValue.Key == Name03; gamer3.ToString() == Id == 3; Name == Name03; Email: [3@3.com](mailto:3@3.com)

            //name04KeyValue.Key == Name04; gamer4.ToString() == Id == 4; Name == Name04; Email: [4@4.com](mailto:4@4.com)

            Console.WriteLine("2.2. dictionaryGamers input key output value ----------------");

            Gamer g1 = dictionaryGamers["Name01"];

            Console.WriteLine("dictionaryGamers[\"Name01\"]  ==  {0}", g1);

            //2.2.dictionaryGamers input key output value ----------------

            //dictionaryGamers["Name01"] == Id == 1; Name == Name01; Email: [1@1.com](mailto:1@1.com)

            Console.WriteLine("2.3. dictionaryGamers input Index output value ----------------");

            string lastItem = dictionaryGamers.Keys.ElementAt(dictionaryGamers.Count - 1);

            Console.WriteLine($"dictionaryGamers.Keys.ElementAt(dictionaryGamers.Count - 1)  ==  {lastItem}");  // Key, "Name04"

            Console.WriteLine($"dictionaryGamers[dictionaryGamers.Keys.ElementAt(dictionaryGamers.Count - 1)]  ==  " +

                              $"{dictionaryGamers[lastItem]}"); // Value, (Id == 4 ; Name == Name04 ; Email : [4@4.com](mailto:4@4.com))

            //2.3.dictionaryGamers input Index output value ----------------

            //dictionaryGamers.Keys.ElementAt(dictionaryGamers.Count - 1) == Name04

            //dictionaryGamers[dictionaryGamers.Keys.ElementAt(dictionaryGamers.Count - 1)] == Id == 4; Name == Name04; Email: [4@4.com](mailto:4@4.com)

            string fitstItem = dictionaryGamers.Keys.ElementAt(0);

            Console.WriteLine($"dictionaryGamers.Keys.ElementAt(0)  ==  {fitstItem}");  // Key, "Name01"

            Console.WriteLine($"dictionaryGamers[dictionaryGamers.Keys.ElementAt(0)]  ==  " +

                              $"{dictionaryGamers[fitstItem]}");    // Value, (Id == 1 ; Name == Name01 ; Email : [1@1.com](mailto:1@1.com))

            //dictionaryGamers.Keys.ElementAt(0) == Name01

            //dictionaryGamers[dictionaryGamers.Keys.ElementAt(0)] == Id == 1; Name == Name01; Email: [1@1.com](mailto:1@1.com)

        }

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

        static void ListOfStringIndexerSample()

        {

            Console.WriteLine("3.1. input index integer, output string magicSpell ---------------");

            MagicSpell magicSpell = new MagicSpell();

            string magicSpell0 = magicSpell[0]; // input is index integer of List, output is magicSpell string.

            Console.WriteLine($"magicSpell[0]  ==  {magicSpell[0]}");   //magicSpell[0]  ==  Spell01

            Console.WriteLine($"magicSpell[1]  ==  {magicSpell[1]}");   //magicSpell[1]  ==  Spell02

            Console.WriteLine($"magicSpell[2]  ==  {magicSpell[2]}");   //magicSpell[2]  ==  Spell03

            Console.WriteLine($"magicSpell[3]  ==  {magicSpell[3]}");   //magicSpell[3]  ==  Spell04

            //Console.WriteLine($"magicSpell[4]  ==  {magicSpell[4]}");   // Error! Throw Example

            //3.1.input index integer, output string magicSpell ---------------

            //magicSpell[0] == Spell01

            //magicSpell[1] == Spell02

            //magicSpell[2] == Spell03

            //magicSpell[3] == Spell04

            Console.WriteLine("3.2. magic spell changed ---------------");

            magicSpell[0] += "New";

            magicSpell[1] += "New";

            magicSpell[2] += "New";

            magicSpell[3] += "New";

            Console.WriteLine($"magicSpell[0]  ==  {magicSpell[0]}");   //magicSpell[0]  ==  Spell01New

            Console.WriteLine($"magicSpell[1]  ==  {magicSpell[1]}");   //magicSpell[1]  ==  Spell02New

            Console.WriteLine($"magicSpell[2]  ==  {magicSpell[2]}");   //magicSpell[2]  ==  Spell03New

            Console.WriteLine($"magicSpell[3]  ==  {magicSpell[3]}");   //magicSpell[3]  ==  Spell04New

            //3.2.magic spell changed ---------------

            //magicSpell[0] == Spell01New

            //magicSpell[1] == Spell02New

            //magicSpell[2] == Spell03New

            //magicSpell[3] == Spell04New

            Console.WriteLine("3.3. Add magic spell if index is not found. ---------------");

            magicSpell[4] = "Spell05";

            Console.WriteLine($"magicSpell[4]  ==  {magicSpell[4]}");   //magicSpell[4]  ==  Spell05

            //3.3.Add magic spell if index is not found. -------------- -

            //magicSpell[4] == Spell05

            Console.WriteLine("3.4. Input string magicSpell, output index integer. ---------------");

            Console.WriteLine($"magicSpell[\"Spell05\"]  ==  " +

                              $"{magicSpell["Spell05"]}");   //magicSpell["Spell05"]  ==  4

            Console.WriteLine($"magicSpell[\"Spell03New\"]  ==  " +

                              $"{magicSpell["Spell03New"]}");   //magicSpell["Spell03New"] == 2

            //Console.WriteLine($"magicSpell[\"Spell0\"]  ==  " +

            //                  $"{magicSpell["Spell0"]}");   //Error! Throw Example

            //3.4.Input string magicSpell, output index integer. ---------------

            //magicSpell["Spell05"] == 4

            //magicSpell["Spell03New"] == 2

        }

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

        static void ListOfGamerIndexerSample()

        {

            Console.WriteLine("4.1. input Id output name ---------------------------------");

            Team team = new Team();

            Console.WriteLine($"Id 1 Gamer Name == team[1] == {team[1]}");

            Console.WriteLine($"Id 2 Gamer Name == team[2] == {team[2]}");

            Console.WriteLine($"Id 3 Gamer Name == team[3] == {team[3]}");

            Console.WriteLine($"Id 4 Gamer Name == team[4] == {team[4]}");

            //4.1.input Id output name-------------------------------- -

            //Id 1 Gamer Name == team[1] == Name01

            //Id 2 Gamer Name == team[2] == Name02

            //Id 3 Gamer Name == team[3] == Name03

            //Id 4 Gamer Name == team[4] == Name04

            Console.WriteLine("4.2. input Id output name : Change Names -----------------------");

            team[1] += "New";

            team[2] += "New";

            team[3] += "New";

            team[4] += "New";

            Console.WriteLine($"Id 1 Gamer Name == team[1] == {team[1]}");

            Console.WriteLine($"Id 2 Gamer Name == team[2] == {team[2]}");

            Console.WriteLine($"Id 3 Gamer Name == team[3] == {team[3]}");

            Console.WriteLine($"Id 4 Gamer Name == team[4] == {team[4]}");

            //4.2.input Id output name: Change Names -----------------------

            //Id 1 Gamer Name == team[1] == Name01New

            //Id 2 Gamer Name == team[2] == Name02New

            //Id 3 Gamer Name == team[3] == Name03New

            //Id 4 Gamer Name == team[4] == Name04New

            Console.WriteLine("4.3. if input Id does not exist, then add new gamer -----------------");

            team[5] = "Name05";

            Console.WriteLine($"Id 5 Gamer Name == team[5] == {team[5]}");

            //4.3. if input Id does not exist, then add new gamer ---------------- -

            //Id 5 Gamer Name == team[5] == Name05

            Console.WriteLine("4.4. input Email output name -----------------");

            Console.WriteLine($"[1@1.com](mailto:1@1.com) Gamer Name == team[\"[1@1.com](mailto:1@1.com)\"] == {team["[1@1.com](mailto:1@1.com)"]}");

            Console.WriteLine($"[2@2.com](mailto:2@2.com) Gamer Name == team[\"[2@2.com](mailto:2@2.com)\"] == {team["[2@2.com](mailto:2@2.com)"]}");

            Console.WriteLine($"[3@3.com](mailto:3@3.com) Gamer Name == team[\"[3@3.com](mailto:3@3.com)\"] == {team["[3@3.com](mailto:3@3.com)"]}");

            Console.WriteLine($"[4@4.com](mailto:4@4.com) Gamer Name == team[\"[4@4.com](mailto:4@4.com)\"] == {team["[4@4.com](mailto:4@4.com)"]}");

            Console.WriteLine("Changing names---------------");

            team["[1@1.com](mailto:1@1.com)"] += "New2";

            team["[2@2.com](mailto:2@2.com)"] += "New2";

            team["[3@3.com](mailto:3@3.com)"] += "New2";

            team["[4@4.com](mailto:4@4.com)"] += "New2";

            Console.WriteLine($"[1@1.com](mailto:1@1.com) Gamer Name == team[\"[1@1.com](mailto:1@1.com)\"] == {team["[1@1.com](mailto:1@1.com)"]}");

            Console.WriteLine($"[2@2.com](mailto:2@2.com) Gamer Name == team[\"[2@2.com](mailto:2@2.com)\"] == {team["[2@2.com](mailto:2@2.com)"]}");

            Console.WriteLine($"[3@3.com](mailto:3@3.com) Gamer Name == team[\"[3@3.com](mailto:3@3.com)\"] == {team["[3@3.com](mailto:3@3.com)"]}");

            Console.WriteLine($"[4@4.com](mailto:4@4.com) Gamer Name == team[\"[4@4.com](mailto:4@4.com)\"] == {team["[4@4.com](mailto:4@4.com)"]}");

            //4.4.input Email output name---------------- -

            //[1@1.com](mailto:1@1.com) Gamer Name == team["[1@1.com](mailto:1@1.com)"] == Name01New

            //[2@2.com](mailto:2@2.com) Gamer Name == team["[2@2.com](mailto:2@2.com)"] == Name02New

            //[3@3.com](mailto:3@3.com) Gamer Name == team["[3@3.com](mailto:3@3.com)"] == Name03New

            //[4@4.com](mailto:4@4.com) Gamer Name == team["[4@4.com](mailto:4@4.com)"] == Name04New

            //Changing names---------------

            //[1@1.com](mailto:1@1.com) Gamer Name == team["[1@1.com](mailto:1@1.com)"] == Name01NewNew2

            //[2@2.com](mailto:2@2.com) Gamer Name == team["[2@2.com](mailto:2@2.com)"] == Name02NewNew2

            //[3@3.com](mailto:3@3.com) Gamer Name == team["[3@3.com](mailto:3@3.com)"] == Name03NewNew2

            //[4@4.com](mailto:4@4.com) Gamer Name == team["[4@4.com](mailto:4@4.com)"] == Name04NewNew2

        }

       // 5. ---------------------------------------------

        static void ListOfGamerIndexerSample2()

        {

            Console.WriteLine("5.1. input index output gamer ---------------------------------");

            TeamA teamA = new TeamA();

            Gamer gamer0 = teamA[0];

            Console.WriteLine($"Index==0 Gamer Name == team[0] == {gamer0.ToString()}");

            Console.WriteLine($"Index==1 Gamer Name == team[1] == {teamA[1]}");

            Console.WriteLine($"Index==2 Gamer Name == team[2] == {teamA[2]}");

            Console.WriteLine($"Index==3 Gamer Name == team[3] == {teamA[3]}");

            //5.1.input index output gamer-------------------------------- -

            //Index == 0 Gamer Name == team[0] == Id == 1; Name == Name01; Email: [1@1.com](mailto:1@1.com)

            //Index == 1 Gamer Name == team[1] == Id == 2; Name == Name02; Email: [2@2.com](mailto:2@2.com)

            //Index == 2 Gamer Name == team[2] == Id == 3; Name == Name03; Email: [3@3.com](mailto:3@3.com)

            //Index == 3 Gamer Name == team[3] == Id == 4; Name == Name04; Email: [4@4.com](mailto:4@4.com)

            Console.WriteLine("5.2. input index output gamer : Change Names -----------------------");

            teamA[0].Name += "New"; //Call Get Method.

            teamA[1] = new Gamer { Name = "Name02New2" };  //Call Set Method.

            teamA[2].Name += "New"; //Call Get Method.

            teamA[3].Name += "New"; //Call Get Method.

            Console.WriteLine($"Index==0 Gamer Name == team[0] == {teamA[0]}");

            Console.WriteLine($"Index==1 Gamer Name == team[1] == {teamA[1]}");

            Console.WriteLine($"Index==2 Gamer Name == team[2] == {teamA[2]}");

            Console.WriteLine($"Index==3 Gamer Name == team[3] == {teamA[3]}");

            //5.2.input index output gamer: Change Names -----------------------

            //Index == 0 Gamer Name == team[0] == Id == 1; Name == Name01New; Email: [1@1.com](mailto:1@1.com)

            //Index == 1 Gamer Name == team[1] == Id == 2; Name == Name02New2; Email: [2@2.com](mailto:2@2.com)

            //Index == 2 Gamer Name == team[2] == Id == 3; Name == Name03New; Email: [3@3.com](mailto:3@3.com)

            //Index == 3 Gamer Name == team[3] == Id == 4; Name == Name04New; Email: [4@4.com](mailto:4@4.com)

            Console.WriteLine("5.3. if input index does not exist, then add new gamer -----------------");

            //teamA[5].Name = "Name05";   // Call Get method and get Error!

            teamA[5] = new Gamer

            {

                Id = 5,

                Name = "Name05",

                Email = "[5@5.com](mailto:5@5.com)"

            };

            Console.WriteLine($"Index==4 Gamer Name == team[4] == {teamA[4]}");

            //5.3. if input index does not exist, then add new gamer ---------------- -

            //Index == 4 Gamer Name == team[4] == Id == 5; Name == Name05; Email: [5@5.com](mailto:5@5.com)

            Console.WriteLine("5.4. input Email output name -----------------");

            Console.WriteLine($"[1@1.com](mailto:1@1.com) Gamer Index == team[\"[1@1.com](mailto:1@1.com)\"] == {teamA["[1@1.com](mailto:1@1.com)"]}");

            Console.WriteLine($"[2@2.com](mailto:2@2.com) Gamer Index == team[\"[2@2.com](mailto:2@2.com)\"] == {teamA["[2@2.com](mailto:2@2.com)"]}");

            Console.WriteLine($"[3@3.com](mailto:3@3.com) Gamer Index == team[\"[3@3.com](mailto:3@3.com)\"] == {teamA["[3@3.com](mailto:3@3.com)"]}");

            Console.WriteLine($"[4@4.com](mailto:4@4.com) Gamer Index == team[\"[4@4.com](mailto:4@4.com)\"] == {teamA["[4@4.com](mailto:4@4.com)"]}");

            //5.4.input Email output name---------------- -

            //[1@1.com](mailto:1@1.com) Gamer Index == team["[1@1.com](mailto:1@1.com)"] == 0

            //[2@2.com](mailto:2@2.com) Gamer Index == team["[2@2.com](mailto:2@2.com)"] == 1

            //[3@3.com](mailto:3@3.com) Gamer Index == team["[3@3.com](mailto:3@3.com)"] == 2

            //[4@4.com](mailto:4@4.com) Gamer Index == team["[4@4.com](mailto:4@4.com)"] == 3

        }

       // 6. ---------------------------------------------

        static void Dictionary\_String\_Gamer\_IndexerSample()

        {

            TeamB teamB = new TeamB();

            Console.WriteLine("6.1. Dictionary of Gamer by string key ---------------");

            //Gamer gamerKey1 = teamB["Key1"];

            Console.WriteLine($"teamB[\"Key1\"] == {teamB["Key1"]}");

            Console.WriteLine($"teamB[\"Key2\"] == {teamB["Key2"]}");

            Console.WriteLine($"teamB[\"Key3\"] == {teamB["Key3"]}");

            Console.WriteLine($"teamB[\"Key4\"] == {teamB["Key4"]}");

            //6.1.Dictionary of Gamer by string key ---------------

            //teamB["Key1"] == Id == 1; Name == Name01; Email: [1@1.com](mailto:1@1.com)

            //teamB["Key2"] == Id == 2; Name == Name02; Email: [2@2.com](mailto:2@2.com)

            //teamB["Key3"] == Id == 3; Name == Name03; Email: [3@3.com](mailto:3@3.com)

            //teamB["Key4"] == Id == 4; Name == Name04; Email: [4@4.com](mailto:4@4.com)

            Console.WriteLine("6.2. Add new gamer by string key ---------------");

            teamB["Key5"] = new Gamer

            {

                Id = 5,

                Name = "Name05",

                Email = "[5@5.com](mailto:5@5.com)"

            };

            Console.WriteLine($"teamB[\"Key5\"] == {teamB["Key5"]}");

            //6.2.Add new gamer by string key -------------- -

            //teamB["Key5"] == Id == 5; Name == Name05; Email: [5@5.com](mailto:5@5.com)

            Console.WriteLine("6.3. Modify Gamer by string key ---------------");

            Gamer teamAGamer01 = teamB["Key1"];

            teamAGamer01.Name += "New3";

            Console.WriteLine($"teamB[\"Key1\"] == {teamB["Key1"]}");

            //6.3.Modify Gamer by by string key -------------- -

            //teamB["Key1"] == Id == 1; Name == Name01New3; Email: [1@1.com](mailto:1@1.com)

            Console.WriteLine("6.4. Modify Gamer by string key ---------------");

            teamB["Key1"] = new Gamer();

            Console.WriteLine($"teamB[\"Key1\"] == {teamB["Key1"]}");

            //6.4.Modify Gamer by string key -------------- -

            //teamB["Key1"] == Id == 1; Name == Name01New3; Email: [1@1.com](mailto:1@1.com)

            Console.WriteLine("6.5. Modify Gamer by string key ---------------");

            teamB["Key1"] = new Gamer

            {

                Id = 101,

                Name = "Name101",

                Email = "[101@101.com](mailto:101@101.com)"

            };

            Console.WriteLine($"teamB[\"Key1\"] == {teamB["Key1"]}");

            //6.5.Modify Gamer by string key -------------- -

            //teamB["Key1"] == Id == 101; Name == Name101; Email: [101@101.com](mailto:101@101.com)

            Console.WriteLine("6.6. Dictionary of Gamer by int index ---------------");

            //Gamer gamerKey1 = teamB["Key1"];

            Console.WriteLine($"teamB[0] == {teamB[0]}");

            Console.WriteLine($"teamB[1] == {teamB[1]}");

            Console.WriteLine($"teamB[2] == {teamB[2]}");

            Console.WriteLine($"teamB[3] == {teamB[3]}");

            Console.WriteLine($"teamB[4] == {teamB[4]}");

            //6.6.Dictionary of Gamer by int index ---------------

            //teamB[0] == Id == 101 ; Name == Name101; Email: [101@101.com](mailto:101@101.com)

            //teamB[1] == Id == 2 ; Name == Name02; Email: [2@2.com](mailto:2@2.com)

            //teamB[2] == Id == 3 ; Name == Name03; Email: [3@3.com](mailto:3@3.com)

            //teamB[3] == Id == 4 ; Name == Name04; Email: [4@4.com](mailto:4@4.com)

            //teamB[4] == Id == 5 ; Name == Name05 ; Email : [5@5.com](mailto:5@5.com)

            Console.WriteLine("6.7. Add new gamer by int index ---------------");

            teamB[5] = new Gamer

            {

                Id = 6,

                Name = "Name06",

                Email = "[6@6.com](mailto:6@6.com)"

            };

            Console.WriteLine($"teamB[5] == {teamB[5]}");

            //6.7.Add new gamer by int index ---------------

            //teamB[5] == Id == 6; Name == Name06; Email: [6@6.com](mailto:6@6.com)

            Console.WriteLine("6.8. Modify Gamer by int index ---------------");

            Gamer teamAGamer02 = teamB[1];

            teamAGamer02.Name += "New4";

            Console.WriteLine($"teamB[1] == {teamB[1]}");

            //6.8.Modify Gamer by int index ---------------

            //teamB[1] == Id == 2; Name == Name02New4; Email: [2@2.com](mailto:2@2.com)

            Console.WriteLine("6.9. Modify Gamer by int index ---------------");

            teamB[1] = new Gamer();

            Console.WriteLine($"teamB[1] == {teamB[1]}");

            //6.9.Modify Gamer by int index ---------------

            //teamB[1] == Id == 2; Name == Name02New4; Email: [2@2.com](mailto:2@2.com)

            Console.WriteLine("6.10. Modify Gamer by int index ---------------");

            teamB[1] = new Gamer

            {

                Id = 102,

                Name = "Name102",

                Email = "[102@102.com](mailto:102@102.com)"

            };

            Console.WriteLine($"teamB[1] == {teamB[1]}");

            //6.10.Modify Gamer by int index ---------------

            //teamB[1] == Id == 102; Name == Name102; Email: [102@102.com](mailto:102@102.com)

        }

    }

}

namespace OnLineGame

{

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

    public class MagicSpell

    {

        private List<string> \_listMagicSpells;

        public MagicSpell()

        {

            \_listMagicSpells = new List<string>

            {

                "Spell01", "Spell02", "Spell03", "Spell04"

            };

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "int index" as parameter

        /// Return the magicSpell from the List by the "int index".

        /// Or set the magicSpell in the List by the "int index".

        /// Or Add new magicSpell to the List

        /// if the input index does not exist.

        /// </summary>

        /// <param name="index">The index integer of the List</param>

        /// <returns>The magicSpell from the List by the "int index".</returns>

        public string this[int index]

        {

            // Just like properties indexers have get and set accessors

            get

            {

                string magicSpell = \_listMagicSpells[index];

                if (magicSpell != null)

                {

                    return magicSpell;

                }

                throw new IndexOutOfRangeException($"The index=={index} magic spell can not be found.");

            }

            set

            {

                if (index <= \_listMagicSpells.Count-1)  //means index integer will be found.

                {

                    // string magicSpell = \_listMagicSpells[index];

                    \_listMagicSpells[index] = value;

                }

                else

                {

                    \_listMagicSpells.Add(value);

                }

            }

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "string magicSpell" as parameter

        /// Return the index of List by the input magicSpell.

        /// </summary>

        /// <param name="magicSpell">The string value of magicSpell.</param>

        /// <returns>The index of List by the input magicSpell</returns>

        public int this[string magicSpell]

        {

            // Just like properties indexers have get and set accessors

            get

            {

                int index = \_listMagicSpells.FindIndex(ms => ms == magicSpell);

                if (index != -1)

                {

                    return index;

                }

                throw new Exception("The magicSpell spell can not be found in the list.");

            }

            // You can not change the index, so no "set".

        }

    }

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

    public class Gamer

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public string Email { get; set; }

        public override string ToString()

        {

            return $"Id == {Id} ; Name == {Name} ; Email : {Email}";

        }

    }

    public class Team

    {

        private List<Gamer> \_listGamers;

        public Team()

        {

            \_listGamers = new List<Gamer>();

            \_listGamers.Add(new Gamer

            { Id = 1, Name = "Name01", Email = "[1@1.com](mailto:1@1.com)" });

            \_listGamers.Add(new Gamer

            { Id = 2, Name = "Name02", Email = "[2@2.com](mailto:2@2.com)" });

            \_listGamers.Add(new Gamer

            { Id = 3, Name = "Name03", Email = "[3@3.com](mailto:3@3.com)" });

            \_listGamers.Add(new Gamer

            { Id = 4, Name = "Name04", Email = "[4@4.com](mailto:4@4.com)" });

            //Gamer aa = \_listGamers[0];

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "int index" as parameter

        /// Return the gamer name from the List by the "int index".

        /// Or set the gamer name in the List by the "int index".

        /// Or Add new gamer to the List if the index does not exist.

        /// </summary>

        /// <param name="id">The gamer Id</param>

        /// <returns>the gamer name from the List by the "int index"</returns>

        public string this[int id]

        {

            // Just like properties indexers have get and set accessors

            get

            {

                Gamer gamer = \_listGamers.

                    FirstOrDefault(g => g.Id == id);

                if (gamer != null)

                    return gamer.Name;

                throw new Exception("The Id can not be found in the list.");

            }

            set

            {

                Gamer gamer = \_listGamers.

                    FirstOrDefault(x => x.Id == id);

                if (gamer != null)

                {

                    gamer.Name = value;

                }

                else

                {

                    \_listGamers.Add(new Gamer { Id = id, Name = value, Email = String.Empty });

                }

            }

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "string email" as parameter

        /// Return the gamer name from the List by the "string email".

        /// Or set the gamer name in the List by the "string email".

        /// </summary>

        /// <param name="email">The gamer Email</param>

        /// <returns>the gamer name from the List by the "string email"</returns>

        public string this[string email]

        {

            // Just like properties indexers have get and set accessors

            get

            {

                Gamer gamer = \_listGamers.

                    FirstOrDefault(g => g.Email == email);

                if (gamer != null)

                    return gamer.Name;

                throw new Exception("The email can not be found in the list.");

            }

            set

            {

                Gamer gamer = \_listGamers.

                    FirstOrDefault(g => g.Email == email);

                if (gamer != null)

                {

                    gamer.Name = value;

                }

                else

                {

                    throw new Exception("The email can not be found in the list.");

                }

            }

        }

    }

   // 5. ---------------------------------------------

    public class TeamA

    {

        private List<Gamer> \_listGamers;

        public TeamA()

        {

            \_listGamers = new List<Gamer>();

            \_listGamers.Add(new Gamer

            { Id = 1, Name = "Name01", Email = "[1@1.com](mailto:1@1.com)" });

            \_listGamers.Add(new Gamer

            { Id = 2, Name = "Name02", Email = "[2@2.com](mailto:2@2.com)" });

            \_listGamers.Add(new Gamer

            { Id = 3, Name = "Name03", Email = "[3@3.com](mailto:3@3.com)" });

            \_listGamers.Add(new Gamer

            { Id = 4, Name = "Name04", Email = "[4@4.com](mailto:4@4.com)" });

            //Gamer aa = \_listGamers[0];

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "int index" as parameter

        /// Return the gamer from the List by the "int index".

        /// Or set the gamer in the List by the "int index".

        /// Or Add new gamer to the List

        /// if the input index does not exist.

        /// </summary>

        /// <param name="index">The index integer of the List</param>

        /// <returns>The gamer from the List by the "int index".</returns>

        public Gamer this[int index]

        {

            // Just like properties indexers have get and set accessors

            get

            {

                Gamer gamer = \_listGamers[index];

                if (gamer != null)

                {

                    return gamer;

                }

                throw new IndexOutOfRangeException($"The index=={index} Gamer can not be found.");

            }

            set

            {

                if (index <= \_listGamers.Count - 1)  //means index integer will be found.

                {

                    //\_listGamers[index] = value; // Not good enough

                    Gamer gamer = \_listGamers[index];

                    gamer.Id = value.Id == default(int) ? gamer.Id : value.Id;

                    gamer.Name = value.Name == default(string) ? gamer.Name : value.Name;

                    gamer.Email = value.Email == default(string) ? gamer.Email : value.Email;

                }

                else

                {

                    \_listGamers.Add(value);

                }

            }

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "string email" as parameter

        /// Return the index of List by the input email.

        /// </summary>

        /// <param name="email">The string value of email.</param>

        /// <returns>The index of List by the input email</returns>

        public int this[string email]

        {

            // Just like properties indexers have get and set accessors

            get

            {

                int index = \_listGamers.FindIndex(g => g.Email == email);

                if (index != -1)

                {

                    return index;

                }

                throw new Exception("The email can not be found in the list.");

            }

            // You can not change the index, so no "set".

        }

    }

   // 6. ---------------------------------------------

    public class TeamB

    {

        private Dictionary<string, Gamer> \_dictionaryGamers;

        public TeamB()

        {

            \_dictionaryGamers = new Dictionary<string, Gamer>();

            \_dictionaryGamers.Add("Key1", new Gamer

            { Id = 1, Name = "Name01", Email = "[1@1.com](mailto:1@1.com)" });

            \_dictionaryGamers.Add("Key2", new Gamer

            { Id = 2, Name = "Name02", Email = "[2@2.com](mailto:2@2.com)" });

            \_dictionaryGamers.Add("Key3", new Gamer

            { Id = 3, Name = "Name03", Email = "[3@3.com](mailto:3@3.com)" });

            \_dictionaryGamers.Add("Key4", new Gamer

            { Id = 4, Name = "Name04", Email = "[4@4.com](mailto:4@4.com)" });

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "string key" as parameter

        /// Return the Gamer from the Dictionary by the "string key".

        /// Or set the Gamer in the Dictionary by the "string key".

        /// Or Add new Gamer to the Dictionary

        /// if the input "string key" does not exist.

        /// </summary>

        /// <param name="index">The string key of the Dictionary</param>

        /// <returns>The Gamer from the Dictionary by the "string key".</returns>

        public Gamer this[string key]

        {

            get

            {

                KeyValuePair<string, Gamer> gamerKeyValue = \_dictionaryGamers.

                    FirstOrDefault(x => x.Key == key);

                Gamer gamer = \_dictionaryGamers.

                    FirstOrDefault(x => x.Key == key).Value;

                if (gamer != null)

                {

                    return gamer;

                }

                throw new KeyNotFoundException("The key can not be found.");

            }

            set

            {

                KeyValuePair<string, Gamer> gamerKeyValue =

                    \_dictionaryGamers.FirstOrDefault(x => x.Key == key);

                Gamer gamer =

                    \_dictionaryGamers.FirstOrDefault(x => x.Key == key).Value;

                if (gamer != null)

                {

                    gamer.Id = value.Id == default(int) ?

                        gamer.Id : value.Id;

                    // if new index does not exist, then use old index.

                    gamer.Name = value.Name == default(string) ?

                        gamer.Name : value.Name;

                    gamer.Email = value.Email == default(string) ?

                        gamer.Email : value.Email;

                }

                else

                {

                    \_dictionaryGamers.Add(key, value);

                }

            }

        }

        /// <summary>

        /// Use "this" keyword to create an indexer.

        /// This indexer takes "int index" as parameter

        /// Return the Gamer from the Dictionary by the "int index".

        /// Or set the Gamer in the Dictionary by the "int index".

        /// Or Add new Gamer to the Dictionary

        /// if the input index does not exist.

        /// </summary>

        /// <param name="index">The index integer of the Dictionary</param>

        /// <returns>The Gamer from the Dictionary by the "int index".</returns>

        public Gamer this[int index]

        {

            // Just like properties indexers have get and set accessors

            get

            {

                string keyOfTheIndex = \_dictionaryGamers.Keys.ElementAt(index); // get the key of the index.

                Gamer gamerOfTheIndex = \_dictionaryGamers[keyOfTheIndex];

                if (gamerOfTheIndex != null)

                {

                    return gamerOfTheIndex;

                }

                else

                {

                    throw new IndexOutOfRangeException("Gamer can not be found by the index.");

                }

            }

            set

            {

                if (index <= \_dictionaryGamers.Count - 1)  //means index integer will be found.

                {

                    //\_dictionaryGamers[\_dictionaryGamers.Keys.ElementAt(index)] = value; // Not good enough

                    string keyOfTheIndex = \_dictionaryGamers.Keys.ElementAt(index); // get the key of the index.

                    Gamer gamerOfTheIndex = \_dictionaryGamers[keyOfTheIndex];

                    gamerOfTheIndex.Id = value.Id == default(int) ?

                        gamerOfTheIndex.Id : value.Id;

                    gamerOfTheIndex.Name = value.Name == default(string) ?

                        gamerOfTheIndex.Name : value.Name;

                    gamerOfTheIndex.Email = value.Email == default(string) ?

                        gamerOfTheIndex.Email : value.Email;

                }

                else

                {

                    \_dictionaryGamers.Add("Key"+ (\_dictionaryGamers.Count + 1), value);

                }

            }

        }

    }

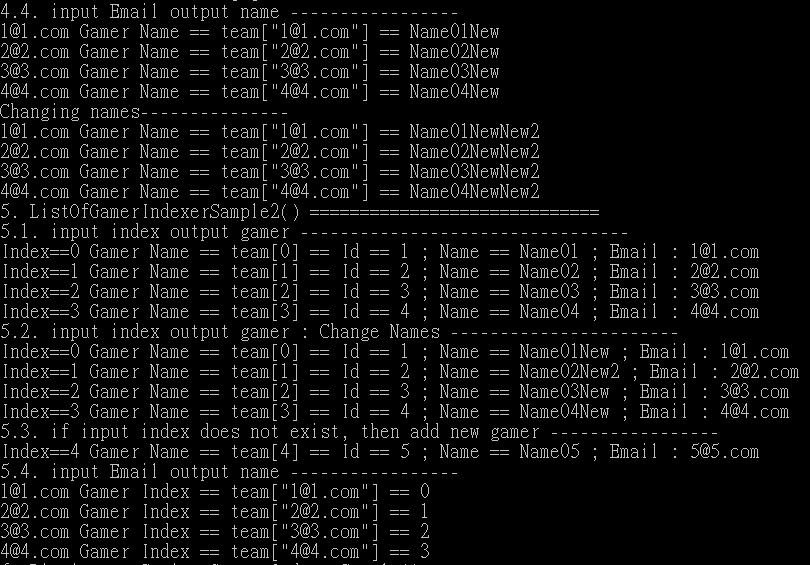
}

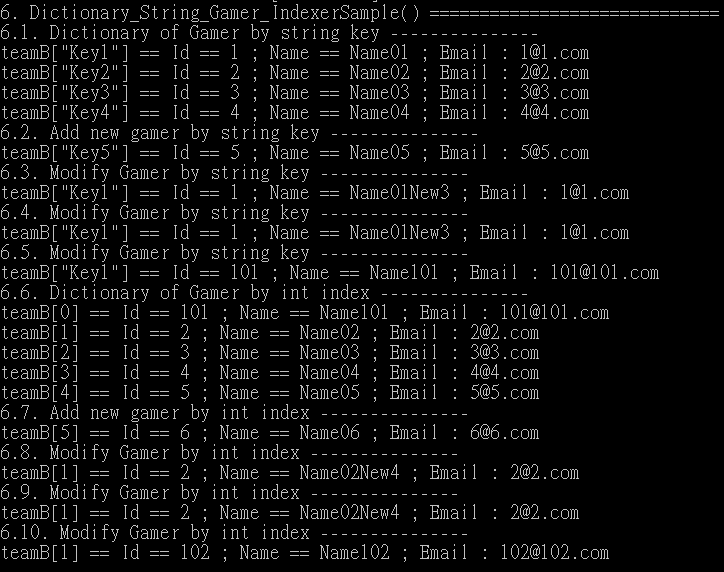
Text

Description automatically generated

Text

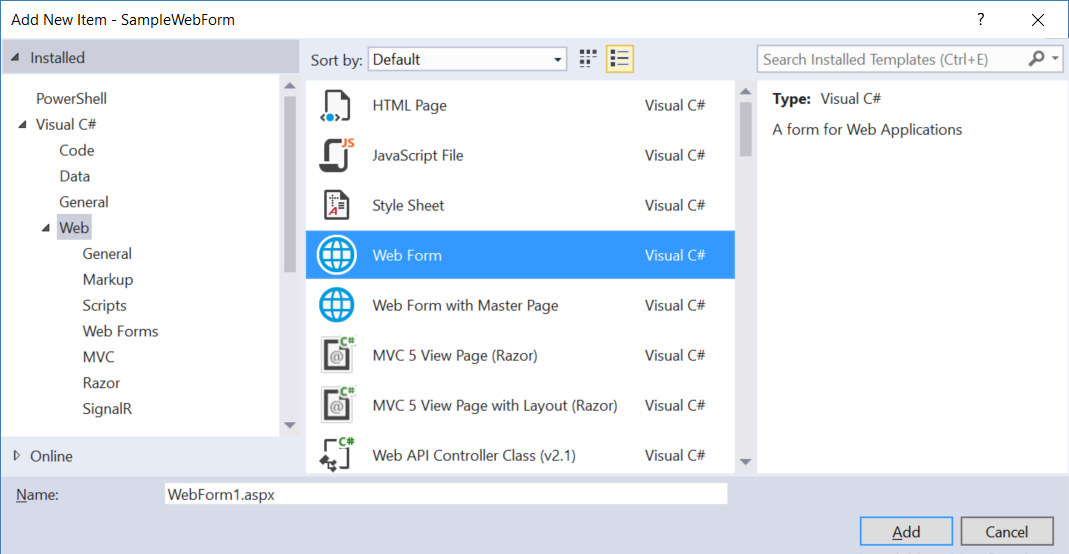
Description automatically generated





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

3. SampleWebForm : WebForm1.aspx.cs



using System;

namespace SampleWebForm

{

    public partial class WebForm1 : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

            // Using the string indexer to store session data

            Session["Session1"] = "Session1 Data";

            Session["Session2"] = "Session2 Data";

            Session["Session2"] = "Session2New Data";   // Change data

            //Session[2] = "Session3 Data";   // Error! can not use int index to set session data

            Session["Session3"] = new Gamer

            {

                Id = 1,

                Name = "Name01",

                Email = "[1@1.com](mailto:1@1.com)"

            };

            // input int index indexer, output object

            Response.Write($"Session[0].ToString() ==  " +

                           $"{Session[0].ToString()} <br/>");

            // input string indexer, output object

            Response.Write($"Session[\"Session2\"].ToString()  ==  " +

                           $"{Session["Session2"].ToString()} <br/>");

            // input int index indexer, output object

            Response.Write($"Session[2].ToString() ==  " +

                           $"{Session[2].ToString()} <br/>");

            // input string indexer, output object

            Response.Write($"Session[\"Session3\"].ToString()  ==  " +

                           $"{Session["Session3"].ToString()} <br/>");

            //Session[0].ToString() == Session1 Data

            //Session["Session2"].ToString() == Session2New Data

            //Session[2].ToString() == Id == 1; Name == Name01; Email: [1@1.com](mailto:1@1.com)

            //Session["Session3"].ToString() == Id == 1; Name == Name01; Email: [1@1.com](mailto:1@1.com)

        }

    }

    public class Gamer

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public string Email { get; set; }

        public override string ToString()

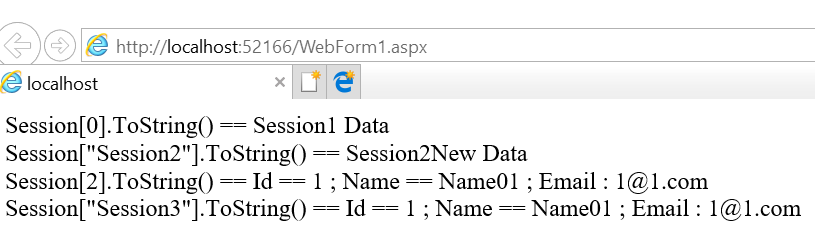
        {

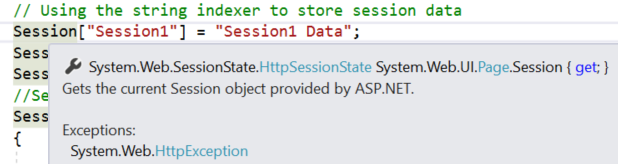
            return $"Id == {Id} ; Name == {Name} ; Email : {Email}";

        }

    }

}





Session is HttpSessionState

Now you may open dotPeek and check HttpSessionState

dotPeek is free software which allows you see the code from dll.

<https://www.jetbrains.com/decompiler/>

Here is the link you may download it.

