(T24)討論 LinqToXml 的 XmlToXml、XmlToHtml、XmlToCsv
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c
==
(T24)討論 LinqToXml 的 XmlToXml、XmlToHtml、XmlToCsv
==

==

0. Summary

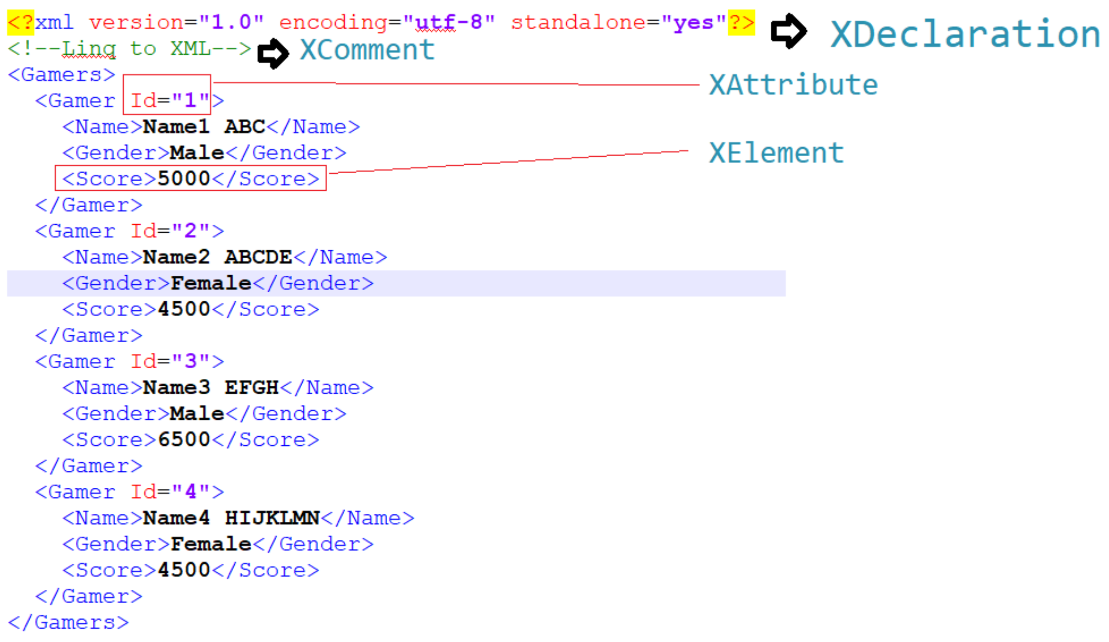
* XML and Reflection。
   * 通常軟體 會把 使用者的設定，儲存在 XML，
XML 通常會包括要讀取的 DLL 名稱，要使用的 class 名稱，要使用的 property 名稱...etc，
然後軟體讀取 XML 裡面的設定，使用 Reflection 將 XML 裡面的字串，
動態讀取 DLL 並且動態去執行一些 method。
   * 要做到這點，首先你必須要對 Linq to XML 非常的了解，
T023_LinqToXml_LinqQueryLet_CreateXml_QueryXml_XmlAdd_XmlUpdate_XmlRemove，
這個 tutorial 是討論 要如何 使用 C#產生 XML，
並且要如何使用 linq 語法 query XML 或是 update/delete/insert xml element
   * T024_XmlToXml_XmlToHtml_XmlToCsv，
這個 tutorial 更絕了，假設某客戶給你一個 XML，
你要如何轉成你公司使用的格式呢?該 tutorial 討論了如何用 C#的 linq to xml，
把 XML 轉成 CSV，或是轉成 HTML 或是轉成另一個格式的 XML。
   * T025_XMLValidation_XSD，這個 tutorial 也很猛，
假設你客戶要求你給他 XML，在上一個 tutoral 你已經學會如何 把 XML 轉換成 另一個格式的 XML，
但是你之後想要寫 test code，所以你要驗證你的 XML 的格式有沒有符合客戶要求，
於是你需要客製化 XSD 來規定 XML 的格式。XSD 上面就是一堆 XML 格式定義，
只要 XML 有符合該定義，validation 之後就會 pass。就代表有符合客戶需求的 XML。
   * C# 課程，T014_ReflectionAndLateBinding，
該 tutorial 介紹了 Reflection 的用法，應用方面的話是，通常你的軟體 讀取 XML 裡面的設定，
使用 Reflection 將 XML 裡面的字串，動態讀取 DLL 並且動態去執行一些 method。
   * C# 課程 ，T015_CustomizedAttributesAndReflection，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. System.Xml.Linq : Linq to XML

This tutorial will convert a xml to other xml, and xml to Html, and Xml to csv

1.1. Linq to XML



```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!--Linq to XML-->
<Gamers>
  <Gamer Id="1">
    <Name>Name1 ABC</Name>
    <Gender>Male</Gender>
    <Score>5000</Score>
  </Gamer>
  <Gamer Id="2">
    <Name>Name2 ABCDE</Name>
    <Gender>Female</Gender>
    <Score>4500</Score>
  </Gamer>
  <Gamer Id="3">
    <Name>Name3 EFGH</Name>
    <Gender>Male</Gender>
    <Score>6500</Score>
  </Gamer>
  <Gamer Id="4">
    <Name>Name4 HIJKLMN</Name>
    <Gender>Female</Gender>
    <Score>4500</Score>
  </Gamer>
</Gamers>
```

1.2. XML for this tutorial

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!--Linq to XML-->
<Gamers>
  <Gamer Type="Magician">
    <Id>1</Id>
    <Name>Name1 ABC</Name>
    <Gender>Male</Gender>
    <Score>5000</Score>
  </Gamer>
  <Gamer Type="Warrior">
    <Id>2</Id>
    <Name>Name2 ABCDE</Name>
    <Gender>Female</Gender>
    <Score>4500</Score>
  </Gamer>
  <Gamer Type="Magician">
    <Id>3</Id>
    <Name>Name3 EFGH</Name>
    <Gender>Male</Gender>
    <Score>6500</Score>
  </Gamer>
```

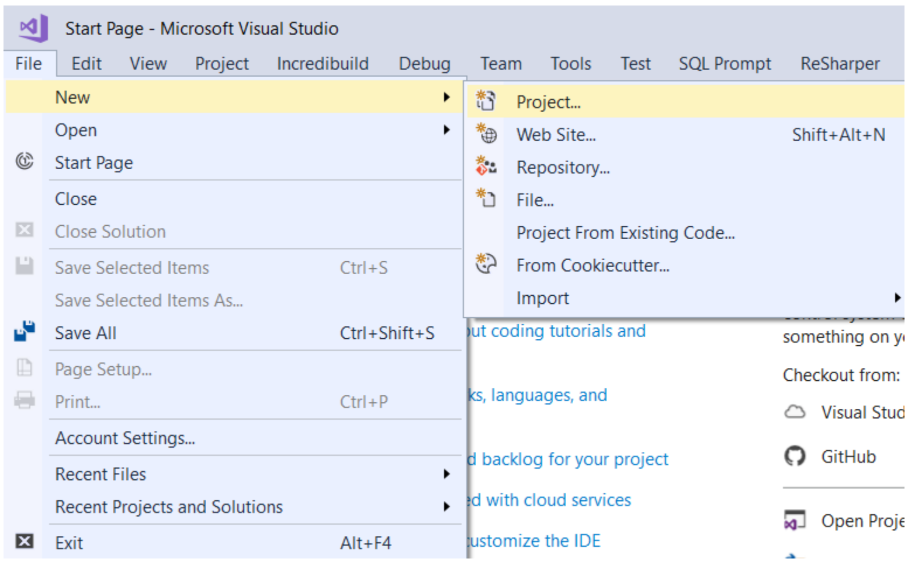
```xml
    <Gamer Type="Warrior">
      <Id>4</Id>
      <Name>Name4 HIJKLMN</Name>
      <Gender>Female</Gender>
      <Score>4500</Score>
    </Gamer>
  </Gamers>
```

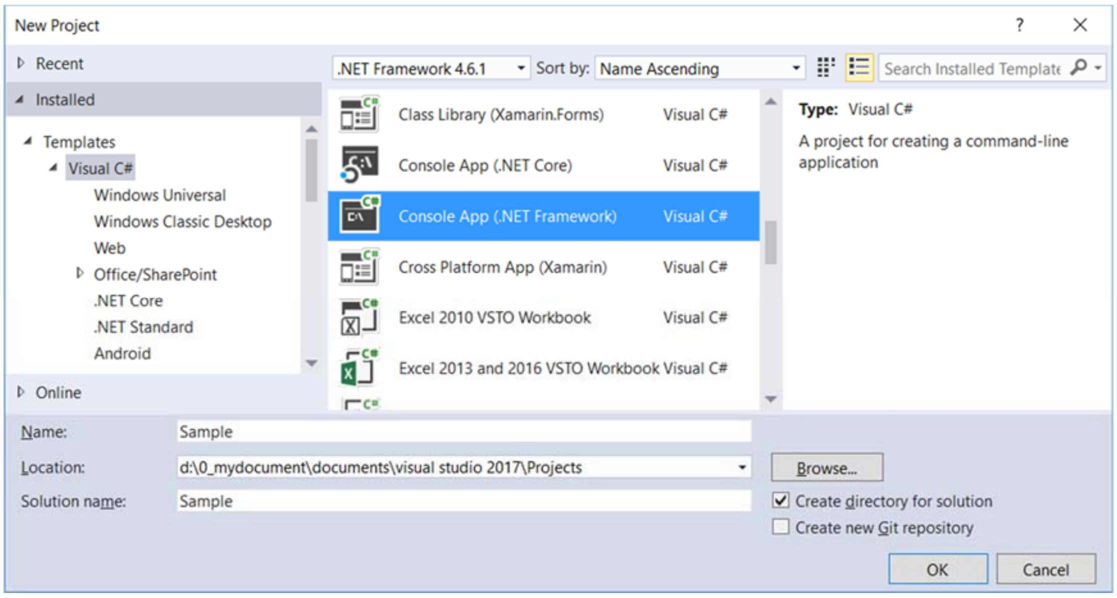
2. Console App

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 Templates
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

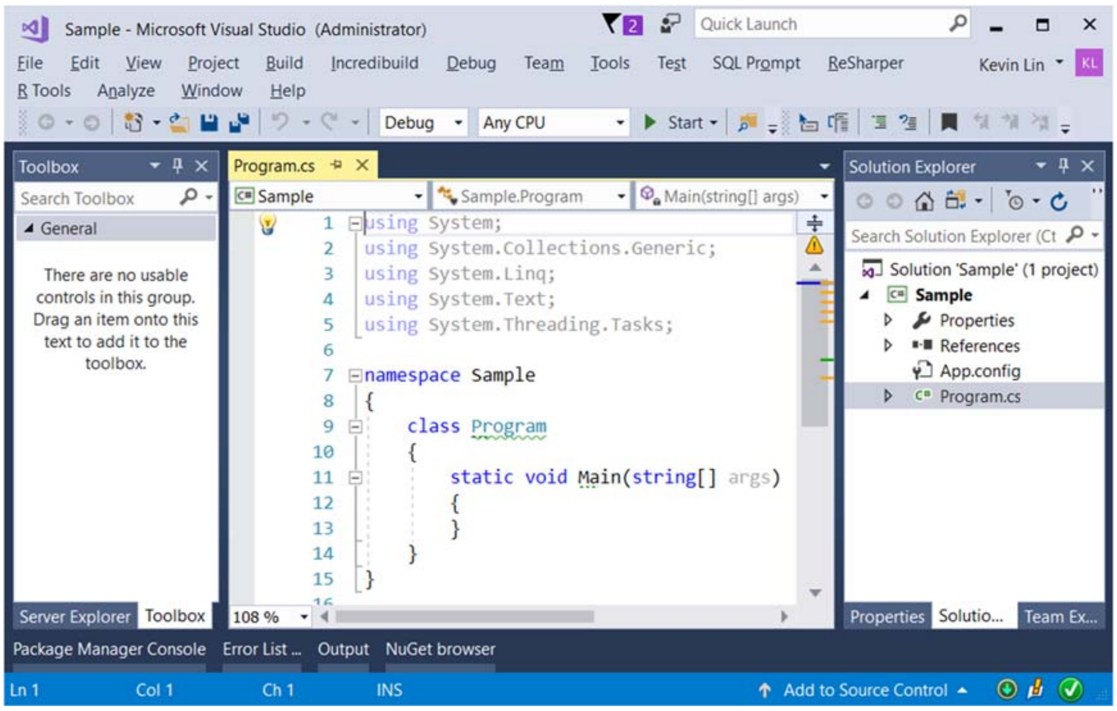
Name: Sample

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

Solution name: Sample

☑ Create directory for solution

☐ Create new Git repository


```
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)
        {
        }
    }
}
```

2.1. Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Xml.Linq;
using OnlineGamer;
namespace Sample
{
    internal class Program
     {
        private static void Main(string[] args)
         {
            // 1. ===========================
            //CreateXml();
            Console.WriteLine("1. CreateXml() ==================== ");
             CreateXml();
            // 2. ===========================
            //CreateXml2();
            Console.WriteLine("2. CreateXml2() ==================== ");
             CreateXml2();
            // 3. ===========================
            //XmlToCsv();
            Console.WriteLine("3. XmlToCsv() ==================== ");
             XmlToCsv();
            // 4. ===========================
            //XmlToHtml();
            Console.WriteLine("4. XmlToHtml() ==================== ");
             XmlToHtml();
            // 5. ===========================
            //XmlToXml();
        Console.WriteLine("5. XmlToXml() ==================== ");
         XmlToXml();
        Console.ReadLine();
    }


  // 1. ===========================
   //CreateXml();
   private static void CreateXml()
    {
        XDocument xDocument = new XDocument(
            new XDeclaration("1.0", "utf-8", "yes"),
            new XComment("Linq to XML"),
            new XElement("Gamers",
                new XElement("Gamer", new XAttribute("Type", "Magician"),
                    new XElement("Id", 1),
                    new XElement("Name", "Name1 ABC"),
                    new XElement("Gender", "Male"),
                    new XElement("Score", 5000)),
                new XElement("Gamer", new XAttribute("Type", "Warrior"),
                    new XElement("Id", 2),
                    new XElement("Name", "Name2 ABCDE"),
                    new XElement("Gender", "Female"),
                    new XElement("Score", 4500)),
                new XElement("Gamer", new XAttribute("Type", "Magician"),
                    new XElement("Id", 3),
                    new XElement("Name", "Name3 EFGH"),
                    new XElement("Gender", "Male"),
                    new XElement("Score", 6500)),
                new XElement("Gamer", new XAttribute("Type", "Warrior"),
                    new XElement("Id", 4),
                    new XElement("Name", "Name4 HIJKLMN"),
                    new XElement("Gender", "Female"),
                    new XElement("Score", 4500))));
         xDocument.Save(@"C:\Xmls\Gamers1.xml");
    }
   //<?xml version = "1.0" encoding="utf-8" standalone="yes"?>
   //<!--Linq to XML-->
   //<Gamers>
   //  <Gamer Type = "Magician" >
   //    < Id > 1 </ Id >
   //    < Name > Name1 ABC</Name>
   //    <Gender>Male</Gender>
   //    <Score>5000</Score>
   //  </Gamer>
   //  <Gamer Type = "Warrior" >
   //    < Id > 2 </ Id >
   //    < Name > Name2 ABCDE</Name>
   //    <Gender>Female</Gender>
   //    <Score>4500</Score>
   //  </Gamer>
   //  <Gamer Type = "Magician" >
   //    < Id > 3 </ Id >
   //    < Name > Name3 EFGH</Name>
```

```
 //    <Gender>Male</Gender>
 //    <Score>6500</Score>
 //  </Gamer>
 //  <Gamer Type = "Warrior" >
 //    < Id > 4 </ Id >
 //    < Name > Name4 HIJKLMN</Name>
 //    <Gender>Female</Gender>
 //    <Score>4500</Score>
 //  </Gamer>
 //</Gamers>


// 2. ===========================
 //CreateXml2();
 private static void CreateXml2()
 {
     List<Gamer> gamersList = GamerHelper.GetAllGamers();
     XDocument xDocument = new XDocument(
         new XDeclaration("1.0", "utf-8", "yes"),
         new XComment("Linq to XML"),
         new XElement("Gamers",
             from gamer in gamersList
             select new XElement("Gamer", new XAttribute("Type", gamer.Type),
                 new XElement("Name", gamer.Id),
                 new XElement("Name", gamer.Name),
                 new XElement("Gender", gamer.Gender),
                 new XElement("Score", gamer.Score))
         ));
     //SaveOptions.DisableFormatting will disable formatting the XML document
     //The following xml will be stored in one single line.
      xDocument.Save(@"C:\Xmls\Gamers2.xml", SaveOptions.DisableFormatting);
 }
 //<?xml version = "1.0" encoding="utf-8" standalone="yes"?>
 //<!--Linq to XML-->
 //<Gamers>
 //  <Gamer Type = "Magician" >
 //    < Id > 1 </ Id >
 //    < Name > Name1 ABC</Name>
 //    <Gender>Male</Gender>
 //    <Score>5000</Score>
 //  </Gamer>
 //  <Gamer Type = "Warrior" >
 //    < Id > 2 </ Id >
 //    < Name > Name2 ABCDE</Name>
 //    <Gender>Female</Gender>
 //    <Score>4500</Score>
 //  </Gamer>
 //  <Gamer Type = "Magician" >
 //    < Id > 3 </ Id >
 //    < Name > Name3 EFGH</Name>
 //    <Gender>Male</Gender>
 //    <Score>6500</Score>
 //  </Gamer>
 //   <Gamer Type = "Warrior" >
 //     < Id > 4 </ Id >
 //     < Name > Name4 HIJKLMN</Name>
 //     <Gender>Female</Gender>
 //     <Score>4500</Score>
 //   </Gamer>
 //</Gamers>
 // 3. ===========================
 //XmlToCsv();
 private static void XmlToCsv()
  {
     var sb = new StringBuilder();
     var delimiter = ",";
     XDocument.Load(@"C:\Xmls\Gamers1.xml").Descendants("Gamer")
         .ToList().ForEach(gamerElement =>
         {
             XAttribute xAttributeType = gamerElement.Attribute("Type");
             if (xAttributeType == null) return;
             XElement xElementId = gamerElement.Element("Id");
             if (xElementId == null) return;
             XElement xElementName = gamerElement.Element("Name");
             if (xElementName == null) return;
             XElement xElementGender = gamerElement.Element("Gender");
             if (xElementGender == null) return;
             XElement xElementScore = gamerElement.Element("Score");
             if (xElementScore != null)
                 sb.Append($"{xAttributeType.Value}{delimiter}" +
                           $"{xElementId.Value}{delimiter}" +
                           $"{xElementName.Value}{delimiter}" +
                           $"{xElementGender.Value}{delimiter}" +
                           $"{xElementScore.Value}{delimiter}\r\n");
         });
     var sw = new StreamWriter(@"C:\Xmls\Gamers1.csv");
      sw.WriteLine(sb.ToString());
      sw.Close();
  }
 // Magician,1,Name1 ABC,Male,5000,
 // Warrior,2,Name2 ABCDE,Female,4500,
 // Magician,3,Name3 EFGH,Male,6500,
 // Warrior,4,Name4 HIJKLMN,Female,4500,



 // 4. ===========================
 //XmlToHtml();
 private static void XmlToHtml()
  {
     XDocument xDocument = XDocument.Load(@"C:\Xmls\Gamers1.xml");
     var xDocumentHtml =
         new XDocument
         (new XElement("table", new XAttribute("border", 1),
             new XElement("thead",
                 new XElement("tr",
                     new XElement("th", "Type"),
                         new XElement("th", "Id"),
                         new XElement("th", "Name"),
                         new XElement("th", "Gender"),
                         new XElement("th", "Score"))),
                 new XElement("tbody",
                     from gamer in xDocument.Descendants("Gamer")
                     let xAttributeType = gamer.Attribute("Type")
                     where xAttributeType != null
                     let xElementId = gamer.Element("Id")
                     where xElementId != null
                     let xElementName = gamer.Element("Name")
                     where xElementName != null
                     let xElementGender = gamer.Element("Gender")
                     where xElementGender != null
                     let xElementScore = gamer.Element("Score")
                     where xElementScore != null
                     select new XElement("tr",
                         new XElement("td", xAttributeType.Value),
                         new XElement("td", xElementId.Value),
                         new XElement("td", xElementName.Value),
                         new XElement("td", xElementGender.Value),
                         new XElement("td", xElementScore.Value)))));
     xDocumentHtml.Save(@"C:\Xmls\Gamers1.html");
 }
 //<?xml version = "1.0" encoding="utf-8"?>
 //<table border = "1" >
 //  < thead >
 //    < tr >
 //      < th > Type </ th >
 //      < th > Id </ th >
 //      < th > Name </ th >
 //      < th > Gender </ th >
 //      < th > Score </ th >
 //    </ tr >
 //  </ thead >
 //  < tbody >
 //    < tr >
 //      < td > Magician </ td >
 //      < td > 1 </ td >
 //      < td > Name1 ABC</td>
 //      <td>Male</td>
 //      <td>5000</td>
 //    </tr>
 //    <tr>
 //      <td>Warrior</td>
 //      <td>2</td>
 //      <td>Name2 ABCDE</td>
 //      <td>Female</td>
 //      <td>4500</td>
 //    </tr>
 //    <tr>
 //      <td>Magician</td>
 //      <td>3</td>
 //      <td>Name3 EFGH</td>
 //      <td>Male</td>
 //      <td>6500</td>
 //    </tr>
 //    <tr>
 //      <td>Warrior</td>
 //      <td>4</td>
 //      <td>Name4 HIJKLMN</td>
 //      <td>Female</td>
 //      <td>4500</td>
 //    </tr>
 //  </tbody>
 //</table>
 // 5. ===========================
 //XmlToXml();
 private static void XmlToXml()
 {
     XDocument xDocumentGamer = XDocument.Load(@"C:\Xmls\Gamers1.xml");
     var xDocumentXml = new XDocument(
         new XElement("Gamers",
             new XElement("Magician",
                 from gamer in xDocumentGamer.Descendants("Gamer")
                 let xAttributeType = gamer.Attribute("Type")
                 where xAttributeType != null && xAttributeType.Value == "Magician"
                 let xElementId = gamer.Element("Id")
                 where xElementId != null
                 let xElementName = gamer.Element("Name")
                 where xElementName != null
                 let xElementGender = gamer.Element("Gender")
                 where xElementGender != null
                 let xElementScore = gamer.Element("Score")
                 where xElementScore != null
                 select new XElement("Gamer",
                     new XElement("Id", xElementId.Value),
                     new XElement("Name", xElementName.Value),
                     new XElement("Gender", xElementGender.Value),
                     new XElement("Score", xElementScore.Value))),
             new XElement("Warrior",
                 from gamer in xDocumentGamer.Descendants("Gamer")
                 let xAttributeWarriorType = gamer.Attribute("Type")
                 where xAttributeWarriorType != null && xAttributeWarriorType.Value == "Warrior"
                 let xElementWarriorId = gamer.Element("Id")
                 where xElementWarriorId != null
                 let xElementWarriorName = gamer.Element("Name")
                 where xElementWarriorName != null
                 let xElementWarriorGender = gamer.Element("Gender")
                 where xElementWarriorGender != null
                 let xElementWarriorScore = gamer.Element("Score")
                 where xElementWarriorScore != null
                 select new XElement("Gamer",
                     new XElement("Id", xElementWarriorId.Value),
                     new XElement("Name", xElementWarriorName.Value),
                     new XElement("Gender", xElementWarriorGender.Value),
                               new XElement("Score", xElementWarriorScore.Value)))));
             xDocumentXml.Save(@"C:\Xmls\Gamers1A.xml");
         }
     }
 }


 namespace OnlineGamer
 {
     // 2. ===========================
     public class Gamer
     {
         public int Id { get; set; }
         public string Name { get; set; }
         public string Gender { get; set; }
         public int Score { get; set; }
         public string Type { get; set; }
         public override string ToString()
         {
             return $"Id=={Id},Name=={Name},Gender=={Gender},Score=={Score},Type=={Type}";
         }
     }
     public class GamerHelper
     {
         public static List<Gamer> GetAllGamers()
         {
             return new List<Gamer>
             {
                 new Gamer {Id = 1, Name = "Name1 ABC", Gender = "Male", Score = 5000, Type = "Magician"},
                 new Gamer {Id = 2, Name = "Name2 ABCDE", Gender = "Female", Score = 4500, Type = "Warrior"},
                 new Gamer {Id = 3, Name = "Name3 EFGH", Gender = "Male", Score = 6500, Type = "Magician"},
                 new Gamer {Id = 4, Name = "Name4 HIJKLMN", Gender = "Female", Score = 4500, Type
 = "Warrior"}
             };
         }
     }
 }
```