(T23)討論LinqToXml的CRUD(Create、Read、Update、Delete)  
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c  
=======================================================================  
(T23)討論LinqToXml的CRUD(Create、Read、Update、Delete)  
=======================================================================  
0. Summary

-----------

1. System.Xml.Linq : Linq to XML

-----------

2. Console App

2.1. Program.cs  
=======================================================================

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.

using "Let" keyword can declare xElement variable in linq query and store "Name" value

E.g.

//// 3.1. .Descendants(\"Gamer\") ----------------------------------------

//Console.WriteLine("3.1. .Descendants(\"Gamer\") ------------ ");

//IEnumerable<string> names =

//    from gamer in XDocument

//        .Load(@"C:\Xmls\Gamers1.xml")   //load xml

//        .Descendants("Gamer")   // find all "Gamer" Descendants

//    where (int)gamer.Element("Score") > 4900    // filter "Gamer" by "Score".

//    orderby (int)gamer.Element("Score") descending  // descending orderby "Score".

//    let xElement = gamer.Element("Name")    //declare xElement variable in linq  query and store "Name" value

//    where xElement != null  //string can be null, if "Name" is not null

//    select xElement.Value;  // project to the "Name" value.

//foreach (string name in names)

//{

//    Console.WriteLine(name);

//}

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

1. System.Xml.Linq : Linq to XML

Graphical user interface, text, application, email

Description automatically generated

2. Console App

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

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

Name: **Sample**

Graphical user interface, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, Excel

Description automatically generated

2.1. Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

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. ===========================

            //QueryXml();

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

            QueryXml();

            // 4. ===========================

            //XmlInsert();

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

            XmlInsert();

            // 5. ===========================

            //XmlUpdate();

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

            XmlUpdate();

            // 6. ===========================

            //XmlUpdateComment();

            Console.WriteLine("6. XmlUpdateComment() ==================== ");

            XmlUpdateComment();

            // 7. ===========================

            //XmlRemoveAllComment();

            Console.WriteLine("7. XmlRemoveAllComment() ==================== ");

            XmlRemoveAllComment();

            // 8. ===========================

            //XmlRemove();

            Console.WriteLine("8. XmlRemove() ==================== ");

            XmlRemove();

            // 9. ===========================

            //XmlRemoveAll();

            Console.WriteLine("9. XmlRemoveAll() ==================== ");

            XmlRemoveAll();

            Console.ReadLine();

        }

       // 1. ===========================

        //CreateXml();

        private static void CreateXml()

        {

            var xDocument = new XDocument(

                new XDeclaration("1.0", "utf-8", "yes"),

                new XComment("Linq to XML"),

                new XElement("Gamers",

                    new XElement("Gamer", new XAttribute("Id", 1),

                        new XElement("Name", "Name1 ABC"),

                        new XElement("Gender", "Male"),

                        new XElement("Score", 5000)),

                    new XElement("Gamer", new XAttribute("Id", 2),

                        new XElement("Name", "Name2 ABCDE"),

                        new XElement("Gender", "Female"),

                        new XElement("Score", 4500)),

                    new XElement("Gamer", new XAttribute("Id", 3),

                        new XElement("Name", "Name3 EFGH"),

                        new XElement("Gender", "Male"),

                        new XElement("Score", 6500)),

                    new XElement("Gamer", new XAttribute("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 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>

        //</Gamer>

       // 2. ===========================

        //CreateXml2();

        private static void CreateXml2()

        {

            List<Gamer> gamersList = GamerHelper.GetAllGamers();

            var 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("Id", 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 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>

        //</Gamer>

       // 3. ===========================

        //QueryXml();

        private static void QueryXml()

        {

            // 3.1. .Descendants(\"Gamer\") ----------------------------------------

           //.Descendants("Gamer") - Can search any element.

            Console.WriteLine("3.1. .Descendants(\"Gamer\") ------------ ");

            IEnumerable<string> names =

                from gamer in XDocument

                    .Load(@"C:\Xmls\Gamers1.xml")   //load xml

                    .Descendants("Gamer")   // find all "Gamer" Descendants

                where (int)gamer.Element("Score") > 4900    // filter "Gamer" by "Score".

                orderby (int)gamer.Element("Score") descending  // descending orderby "Score".

                let xElement = gamer.Element("Name")    //declare xElement variable in linq query and store "Name" value

                where xElement != null  //string can be null, if "Name" is not null

                select xElement.Value;  // project to the "Name" value.

            foreach (string name in names)

            {

                Console.WriteLine(name);

            }

            //Name3 EFGH

            //Name1 ABC

            // 3.2. .Descendants(\"Gamer\") ----------------------------------------

           //.Elements("Gamers").Elements("Gamer")... - "Elements" can only search from outside to inside sequentially.

            Console.WriteLine("3.2. .Descendants(\"Gamer\") ------------ ");

            IEnumerable<string> names2 =

                from gamer in XDocument

                    .Load(@"C:\Xmls\Gamers1.xml")   //load xml

                    .Elements("Gamers")   // find all "Gamers" elements

                    .Elements("Gamer")  // find all "Gamer" elements from the "Gamers" element

                where (int)gamer.Element("Score") > 4900    // filter "Gamer" by "Score".

                orderby (int)gamer.Element("Score") descending  // descending orderby "Score".

                let xElement = gamer.Element("Name")    //declare xElement variable and store "Name" value

                where xElement != null  //string can be null, if "Name" is not null

                select xElement.Value;  // project to the "Name" value.

            foreach (string name in names2)

            {

                Console.WriteLine(name);

            }

            //Name3 EFGH

            //Name1 ABC

        }

       // 4. ===========================

        //XmlInsert();

        static void XmlInsert()

        {

            // 1,2,3,4

            XDocument xDocument = XDocument.Load(@"C:\Xmls\Gamers1.xml");

            XElement xElement = xDocument.Element("Gamers");

            // 4.1. Add ----------------------------------------------

            Console.WriteLine("4.1. Add --------------------- ");

            //if (xElement != null)

            //    xElement.Add(

            xElement?.Add(

                new XElement("Gamer", new XAttribute("Id", 5),

                    new XElement("Name", "Name5 NOP"),

                    new XElement("Gender", "Male"),

                    new XElement("Score", 3000)

                ));

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

            // 1,2,3,4,5

            // 4.2. AddFirst ----------------------------------------------

            Console.WriteLine("4.2. AddFirst --------------------- ");

            //if (xElement != null)

            //    xElement.AddFirst(

            xElement?.AddFirst(

                new XElement("Gamer", new XAttribute("Id", 6),

                    new XElement("Name", "Name6 PQRSTUVW"),

                    new XElement("Gender", "Male"),

                    new XElement("Score", 4000)

                ));

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

            // 6,1,2,3,4,5

            // 4.3. AddBeforeSelf ----------------------------------------------

            Console.WriteLine("4.3. AddBeforeSelf --------------------- ");

            //XElement xElement = xDocument.Element("Gamers");

            if (xElement != null)

            {

                XElement firstOrDefault = xElement.Elements("Gamer").FirstOrDefault(x =>

                {

                    // FirstOrDefault take Func<XElement, bool> predicate as parameter.

                    //This anonymous method need to return a bool.

                    XAttribute xAttribute = x.Attribute("Id");

                    return xAttribute != null && xAttribute.Value == "3"; //it is a string, not int

                });

                //if (firstOrDefault != null)

                //    firstOrDefault.AddBeforeSelf(

                firstOrDefault?.AddBeforeSelf(

                    new XElement("Gamer", new XAttribute("Id", 7),

                        new XElement("Name", "Name7 XYZ"),

                        new XElement("Gender", "Male"),

                        new XElement("Score", 4500)));

            }

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

            // 6,1,2,7,3,4,5

            // 4.4. AddAfterSelf ----------------------------------------------

            Console.WriteLine("4.4. AddAfterSelf --------------------- ");

            //XElement xElement = xDocument.Element("Gamers");

            if (xElement != null)

            {

                XElement firstOrDefault = xElement.Elements("Gamer").FirstOrDefault(x =>

                {

                    // FirstOrDefault take Func<XElement, bool> predicate as parameter.

                    //This anonymous method need to return a bool.

                    XAttribute xAttribute = x.Attribute("Id");

                    return xAttribute != null && xAttribute.Value == "3"; //it is a string, not int

                });

                //if (firstOrDefault != null)

                //    firstOrDefault.AddAfterSelf(

                firstOrDefault?.AddAfterSelf(

                    new XElement("Gamer", new XAttribute("Id", 8),

                        new XElement("Name", "Name8 ZABD"),

                        new XElement("Gender", "Male"),

                        new XElement("Score", 4500)));

            }

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

            // 6,1,2,7,3,8,4,5

        }

       // 5. ===========================

        //XmlUpdate();

        static void XmlUpdate()

        {

            XDocument xDocument = XDocument.Load(@"C:\Xmls\Gamers1.xml");

            XElement xElement = xDocument.Element("Gamers");

            // 5.1. SetElementValue ----------------------------------------------

            Console.WriteLine("5.1. SetElementValue -------------------------- ");

            // if (xElement != null){

            //    XElement firstOrDefault = xElement.Elements("Gamer")....

            XElement firstOrDefault = xElement?.Elements("Gamer")

                .Where(x =>

                {

                    XAttribute xAttribute = x.Attribute("Id");

                    return xAttribute != null && xAttribute.Value == "5";   //it is a string, not int

                }).FirstOrDefault();

            //if (firstOrDefault != null)

            //    firstOrDefault.SetElementValue("Score", 5555);

            firstOrDefault?.SetElementValue("Score", 5555);

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

            //Get Id==5 Gamer, and update its Score to 5555.

            // 5.2. SetValue ----------------------------------------------

            Console.WriteLine("5.2. SetValue -------------------------- ");

            // if (xElement != null){

            //    XElement firstOrDefault = xElement.Elements("Gamer")....

            XElement firstOrDefault2 = xElement?.Elements("Gamer")

                .Where(x =>

                {

                    XAttribute xAttribute = x.Attribute("Id");

                    return xAttribute != null && xAttribute.Value == "4";   //it is a string, not int

                })

                .Select(x => x.Element("Score")).FirstOrDefault();

            //if (firstOrDefault2 != null)

            //    firstOrDefault2?.SetValue(4444);

            firstOrDefault2?.SetValue(4444);

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

            //Get Id==4 Gamer, and update its Score to 4444.

        }

        // 6. ===========================

        //XmlUpdateComment();

        static void XmlUpdateComment()

        {

            XDocument xDocument = XDocument.Load(@"C:\Xmls\Gamers1.xml");

            //Update the first comment

            //Nodes() returns the collections of child nodes

            //OfType<XComment> filters the collection and returns only XComment type object.

            XComment firstOrDefault =

                xDocument.Nodes().OfType<XComment>().FirstOrDefault();

            if (firstOrDefault != null)

                firstOrDefault.Value += "\_New";

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

        }

        // 7. ===========================

        //XmlRemoveAllComment();

        static void XmlRemoveAllComment()

        {

            XDocument xDocument = XDocument.Load(@"C:\Xmls\Gamers1.xml");

            //Remove all xml comments

            //Nodes() returns the collections of child nodes

            //OfType<XComment> filters the collection and returns only XComment type object.

            xDocument.Nodes().OfType<XComment>().Remove();

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

        }

       // 8. ===========================

        //XmlRemove();

        static void XmlRemove()

        {

            XDocument xDocument = XDocument.Load(@"C:\Xmls\Gamers1.xml");

            //Get the Id==5 Gamer and then remove it.

            //if (xDocument.Root != null)

            //    xDocument.Root.Elements().Where(x =>

            xDocument.Root?.Elements().Where(x =>

            {

                XAttribute xAttribute = x.Attribute("Id");

                return xAttribute != null && xAttribute.Value == "5";   //it is a string, not int

            }).Remove();

            xDocument.Save(@"C:\Xmls\Gamers1.xml");

        }

        // 9. ===========================

        //XmlRemoveAll();

        static void XmlRemoveAll()

        {

            XDocument xDocument = XDocument.Load(@"C:\Xmls\Gamers1.xml");

            //Get the all Gamers and then remove them.

            //Remove All elements

            //if (xDocument.Root != null)

            //    xDocument.Root.Elements().Remove();

            xDocument.Root?.Elements().Remove();

            xDocument.Save(@"C:\Xmls\Gamers1.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 override string ToString()

        {

            return $"Id=={Id},Name=={Name},Gender=={Gender},Score=={Score}";

        }

    }

    public class GamerHelper

    {

        public static List<Gamer> GetAllGamers()

        {

            return new List<Gamer>

            {

                new Gamer {Id = 1, Name = "Name1 ABC", Gender = "Male", Score = 5000},

                new Gamer {Id = 2, Name = "Name2 ABCDE", Gender = "Female", Score = 4500},

                new Gamer {Id = 3, Name = "Name3 EFGH", Gender = "Male", Score = 6500},

                new Gamer {Id = 4, Name = "Name4 HIJKLMN", Gender = "Female", Score = 4500}

            };

        }

    }

}