(T25)討論LinqToXml的Validation，用Xsd來驗證XmlFormat  
CourseGUID: 5ba9a6fe-7475-4b0c-8b99-bbcf7f5e2e1c  
=======================================================================  
(T25)討論LinqToXml的Validation，用Xsd來驗證XmlFormat  
=======================================================================  
0. Summary

-----------

1. XML Schema Definition Language (XSD)

-----------

2. Console App

2.1. Gamers XSD

2.2. Gamers XML

2.2.1. Gamers.xml

2.2.2. Gamers2.xml

2.2.3. Gamers3.xml

2.2.4. Gamers4.xml

2.3. BookStore XSD

2.4. BookStore XML

2.4.1. Bookstore.xml

2.4.2. Bookstore2.xml

2.4.3. Bookstore3.xml

2.4.4. Bookstore4.xml

2.4.5. Bookstore5.xml

2.4.6. Bookstore6.xml

2.5. 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.

Gamers.xsd

Complex Type

Reference:

<https://www.w3schools.com/xml/schema_complex.asp>

[https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset%28v=vs.110%29.aspx)

1.1.

A complex element is an XML element that contains other elements and/or attributes.

We can define a complex element in an XML Schema two different ways.

1.2.

In this case, Gamer is a complex type.

If you use the method described above.

//<xsd:element name="Gamers">

the root element must be "**Gamers**".

The "**Gamers**" contains several "**Gamer**",

and only the "**Gamers**" and "**Gamer**" element can use the specified **complex** type.

1.3.

Note that the child elements, "**Id**", "**Name**", "**Gender**", and "**Score**",

are **surrounded** by the <**sequence**> indicator.

This means that the child elements must appear

in the **same order** as they are declared.

1.4.

//<xsd:element name="Gamer" minOccurs="2" maxOccurs="4">

It means "Gamers" must contain at least 2 "Gamer" and

must not contain more than 4 "Gamer" elements.

1.5.

//<xsd:element name="Id" minOccurs="1" maxOccurs="1"/>

It means "Gamer" can only contain 1 "Id" element.

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

2.

Bookstore.xsd

Complex Type

Reference:

<https://www.w3schools.com/xml/schema_complex.asp>

<https://www.w3schools.com/xml/schema_simple_attributes.asp>

[https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset%28v=vs.110%29.aspx)

2.1.

A complex element is an XML element that contains other elements and/or attributes.

We can define a complex element in an XML Schema two different ways.

2.2.

//<xsd:element name="title" type="xsd:string"/>

//<xsd:element name="bookstore" type="bookstoreType"/>

XML Schema has a lot of built-in data types. The most common types are:

xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, and xs:time

In addition, you may use self define type.

In this case, self define type is "**bookstoreType**"

2.3.

//<xsd:element name="bookstore" type="bookstoreType"/>

...

//<xsd:complexType name="bookstoreType">

//  <xsd:sequence minOccurs="2" maxOccurs="unbounded">

//    <xsd:element name="book"  type="bookType"/>

//  </xsd:sequence>

//</xsd:complexType>

In this case, "**bookstore**" is a complex type.

The "**bookstore**" element can have a self define "**type**" attribute

that refers to the **name** of the **complex type**

//<xsd:sequence minOccurs="2" maxOccurs="unbounded">

Note that the child elements, "**book**"

are **surrounded** by the <**sequence**> indicator.

This means that the child elements must appear

in the **same order** as they are declared.

//minOccurs="2" maxOccurs="unbounded"

means this "**bookstoreType**" can contain

**minim** 2  "**book**" elements,

and the maximum number of elements is "**unbounded**"

2.4.

//<xsd:element name="book"  type="bookType"/>

...

//<xsd:complexType name="bookType">

//     <xsd:sequence>

//       <xsd:element name="title" type="xsd:string"/>

//       <xsd:element name="author" type="authorName"/>

//       <xsd:element name="price"  type="xsd:decimal"/>

//     </xsd:sequence>

//     <xsd:attribute name="genre" type="xsd:string" use="optional"/>

//     <xsd:attribute name="lang" use="optional">

//       <xsd:simpleType>

//         <xsd:restriction base="xsd:string">

//           <xsd:pattern value="EnglishUK|EnglishUSA"/>

//         </xsd:restriction>

//       </xsd:simpleType>

//     </xsd:attribute>

//</xsd:complexType>

In this case, "**bookType**" is a complex type.

The "**bookType**" element can have a self-define "**type**" attribute

that refers to the **name** of the **complex type**

//<xsd:sequence>....

Note that the child elements, "**title**", "**author**"and "**price**",

are **surrounded** by the <**sequence**> indicator.

This means that the child elements must appear

in the **same order** as they are declared.

//<xsd:attribute name="genre" type="xsd:string" use="optional"/>

The "**attribute**" is after "**sequence**" and

surrounded by the **<xsd:complexType** **name="bookType">**  indicator.

That means to declare the **bookType** can contain these **attributes**.

In this case, "**genre**" is an attribute of "**bookType**",

and the datatype must be a **string**.

"**optional**" means this attribute is optional.

//     <xsd:attribute name="lang" use="optional">

//       <xsd:simpleType>

//         <xsd:restriction base="xsd:string">

//           <xsd:pattern value="EnglishUK|EnglishUSA"/>

//         </xsd:restriction>

//       </xsd:simpleType>

//     </xsd:attribute>

In this case, "**lang**" is an attribute of "**bookType**",

and the datatype must be **string**.

In addition, the value must be **EnglishUK** or **EnglishUSA**

"**optional**" means this attribute is optional.

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

1. XML Schema Definition Language (XSD)

An XSD ( XML Schema Definition Language) file defines the structure of the XML file.

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. Gamers XSD

Project Name --> Right Click --> Add --> New Item

--> **Xml Schema**

--> Name:

**Gamers.xsd**

-->

**Use the XML Editor to view and edit the underlying XML schema file**

-->

Modify the properties

**Copy to Output Directory :  Copy Always**

it means copy this xml to bin/debug

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated with medium confidence

Text

Description automatically generated

<?xml version="1.0" encoding="utf-8"?>

<xsd:schema xmlns:xsd="<http://www.w3.org/2001/XMLSchema>">

  <xsd:element name="Gamers">

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="Gamer" minOccurs="2" maxOccurs="4">

          <xsd:complexType>

            <xsd:sequence>

              <xsd:element name="Id" minOccurs="1" maxOccurs="1"/>

              <xsd:element name="Name" minOccurs="1" maxOccurs="1"/>

              <xsd:element name="Gender" minOccurs="1" maxOccurs="1"/>

              <xsd:element name="Score" minOccurs="1" maxOccurs="1"/>

            </xsd:sequence>

          </xsd:complexType>

        </xsd:element>

      </xsd:sequence>

    </xsd:complexType>

  </xsd:element>

</xsd:schema>

1.

Gamers.xsd

Complex Type

Reference:

<https://www.w3schools.com/xml/schema_complex.asp>

[https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset%28v=vs.110%29.aspx)

1.1.

A complex element is an XML element that contains other elements and/or attributes.

We can define a complex element in an XML Schema two different ways.

1.2.

In this case, Gamer is a complex type.

If you use the method described above.

//<xsd:element name="Gamers">

the root element must be "**Gamers**".

The "**Gamers**" contains several "**Gamer**",

and only the "**Gamers**" and "**Gamer**" element can use the specified **complex** type.

1.3.

Note that the child elements, "**Id**", "**Name**", "**Gender**", and "**Score**",

are **surrounded** by the <**sequence**> indicator.

This means that the child elements must appear

in the **same order** as they are declared.

1.4.

//<xsd:element name="Gamer" minOccurs="2" maxOccurs="4">

It means "Gamers" must contain at least 2 "Gamer" and

must not contain more than 4 "Gamer" elements.

1.5.

//<xsd:element name="Id" minOccurs="1" maxOccurs="1"/>

It means "Gamer" can only contain 1 "Id" element.

2.2. Gamers XML

2.2.1. Gamers.xml

Project Name --> Right Click --> Add --> New Item

--> **XML File**

--> Name:

**Gamer.xml**

-->

Modify the properties

**Copy to Output Directory :  Copy Always**

it means copy this xml to bin/debug

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, application, table

Description automatically generated

<?xml version="1.0" encoding="utf-8" ?>

<Gamers>

  <Gamer>

    <Id>1</Id>

    <Name>Name1 ABC</Name>

    <Gender>Male</Gender>

    <Score>5000</Score>

  </Gamer>

  <Gamer>

    <Id>2</Id>

    <Name>Name2 ABCDE</Name>

    <Gender>Female</Gender>

    <Score>4500</Score>

  </Gamer>

  <Gamer>

    <Id>3</Id>

    <Name>Name3 EFGH</Name>

    <Gender>Male</Gender>

    <Score>6500</Score>

  </Gamer>

  <Gamer>

    <Id>4</Id>

    <Name>Name4 HIJKLMN</Name>

    <Gender>Female</Gender>

    <Score>4500</Score>

  </Gamer>

</Gamers>

2.2.2. Gamers2.xml

More than 4 Gamers

<?xml version="1.0" encoding="utf-8" ?>

<Gamers>

  <Gamer>

    <Id>1</Id>

    <Name>Name1 ABC</Name>

    <Gender>Male</Gender>

    <Score>5000</Score>

  </Gamer>

  <Gamer>

    <Id>2</Id>

    <Name>Name2 ABCDE</Name>

    <Gender>Female</Gender>

    <Score>4500</Score>

  </Gamer>

  <Gamer>

    <Id>3</Id>

    <Name>Name3 EFGH</Name>

    <Gender>Male</Gender>

    <Score>6500</Score>

  </Gamer>

  <Gamer>

    <Id>4</Id>

    <Name>Name4 HIJKLMN</Name>

    <Gender>Female</Gender>

    <Score>4500</Score>

  </Gamer>

  <Gamer>

    <Id>5</Id>

    <Name>Name5 OPQ</Name>

    <Gender>Male</Gender>

    <Score>6500</Score>

  </Gamer>

</Gamers>

2.2.3. Gamers3.xml

Less than 2 Gamers

<?xml version="1.0" encoding="utf-8" ?>

<Gamers>

  <Gamer>

    <Id>1</Id>

    <Name>Name1 ABC</Name>

    <Gender>Male</Gender>

    <Score>5000</Score>

  </Gamer>

</Gamers>

2.2.4. Gamers4.xml

The Score and Id is not in the right position.

<?xml version="1.0" encoding="utf-8" ?>

<Gamers>

  <Gamer>

    <Id>1</Id>

    <Name>Name1 ABC</Name>

    <Gender>Male</Gender>

    <Score>5000</Score>

  </Gamer>

  <Gamer>

    <Id>2</Id>

    <Name>Name2 ABCDE</Name>

    <Gender>Female</Gender>

    <Score>4500</Score>

  </Gamer>

  <Gamer>

    <Score>6500</Score>

    <Id>3</Id>

    <Name>Name3 EFGH</Name>

    <Gender>Male</Gender>

  </Gamer>

</Gamers>

2.3. BookStore XSD

<xsd:schema xmlns:xsd="<http://www.w3.org/2001/XMLSchema>"

    xmlns="urn:bookstore-schema"

    elementFormDefault="qualified"

    targetNamespace="urn:bookstore-schema">

  <xsd:element name="bookstore" type="bookstoreType"/>

  <xsd:complexType name="bookstoreType">

    <xsd:sequence minOccurs="2" maxOccurs="unbounded">

      <xsd:element name="book"  type="bookType"/>

    </xsd:sequence>

  </xsd:complexType>

  <xsd:complexType name="bookType">

    <xsd:sequence>

      <xsd:element name="title" type="xsd:string"/>

      <xsd:element name="author" type="authorName"/>

      <xsd:element name="price"  type="xsd:decimal"/>

    </xsd:sequence>

    <xsd:attribute name="genre" type="xsd:string" use="optional"/>

    <xsd:attribute name="lang" use="optional">

      <xsd:simpleType>

        <xsd:restriction base="xsd:string">

          <xsd:pattern value="EnglishUK|EnglishUSA"/>

        </xsd:restriction>

      </xsd:simpleType>

    </xsd:attribute>

  </xsd:complexType>

  <xsd:complexType name="authorName">

    <xsd:sequence>

      <xsd:element name="first-name"  type="xsd:string"/>

      <xsd:element name="last-name" type="xsd:string"/>

    </xsd:sequence>

  </xsd:complexType>

</xsd:schema>

2.

Bookstore.xsd

Complex Type

Reference:

<https://www.w3schools.com/xml/schema_complex.asp>

<https://www.w3schools.com/xml/schema_simple_attributes.asp>

[https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset(v=vs.110).aspx](https://msdn.microsoft.com/en-us/library/system.xml.schema.xmlschemaset%28v=vs.110%29.aspx)

2.1.

A complex element is an XML element that contains other elements and/or attributes.

We can define a complex element in an XML Schema two different ways.

2.2.

//<xsd:element name="title" type="xsd:string"/>

//<xsd:element name="bookstore" type="bookstoreType"/>

XML Schema has a lot of built-in data types. The most common types are:

xs:string, xs:decimal, xs:integer, xs:boolean, xs:date, and xs:time

In addition, you may use self define type.

In this case, self define type is "**bookstoreType**"

2.3.

//<xsd:element name="bookstore" type="bookstoreType"/>

...

//<xsd:complexType name="bookstoreType">

//  <xsd:sequence minOccurs="2" maxOccurs="unbounded">

//    <xsd:element name="book"  type="bookType"/>

//  </xsd:sequence>

//</xsd:complexType>

In this case, "**bookstore**" is a complex type.

The "**bookstore**" element can have a self define "**type**" attribute

that refers to the **name** of the **complex type**

//<xsd:sequence minOccurs="2" maxOccurs="unbounded">

Note that the child elements, "**book**"

are **surrounded** by the <**sequence**> indicator.

This means that the child elements must appear

in the **same order** as they are declared.

//minOccurs="2" maxOccurs="unbounded"

means this "**bookstoreType**" can contain

**minim** 2  "**book**" elements,

and the maximum number of elements is "**unbounded**"

2.4.

//<xsd:element name="book"  type="bookType"/>

...

//<xsd:complexType name="bookType">

//     <xsd:sequence>

//       <xsd:element name="title" type="xsd:string"/>

//       <xsd:element name="author" type="authorName"/>

//       <xsd:element name="price"  type="xsd:decimal"/>

//     </xsd:sequence>

//     <xsd:attribute name="genre" type="xsd:string" use="optional"/>

//     <xsd:attribute name="lang" use="optional">

//       <xsd:simpleType>

//         <xsd:restriction base="xsd:string">

//           <xsd:pattern value="EnglishUK|EnglishUSA"/>

//         </xsd:restriction>

//       </xsd:simpleType>

//     </xsd:attribute>

//</xsd:complexType>

In this case, "**bookType**" is a complex type.

The "**bookType**" element can have a self define "**type**" attribute

that refers to the **name** of the **complex type**

//<xsd:sequence>....

Note that the child elements, "**title**", "**author**"and "**price**",

are **surrounded** by the <**sequence**> indicator.

This means that the child elements must appear

in the **same order** as they are declared.

//<xsd:attribute name="genre" type="xsd:string" use="optional"/>

The "**attribute**" is after "**sequence**" and

surrounded by the **<xsd:complexType** **name="bookType">**  indicator.

That means declare the **bookType** can contain these **attributes**.

In this case, "**genre**" is an attribute of "**bookType**",

and the datatype must be **string**.

"**optional**" means this attribute is optional.

//     <xsd:attribute name="lang" use="optional">

//       <xsd:simpleType>

//         <xsd:restriction base="xsd:string">

//           <xsd:pattern value="EnglishUK|EnglishUSA"/>

//         </xsd:restriction>

//       </xsd:simpleType>

//     </xsd:attribute>

In this case, "**lang**" is an attribute of "**bookType**",

and the datatype must be **string**.

In addition, the value must be **EnglishUK** or **EnglishUSA**

"**optional**" means this attribute is optional.

2.4. BookStore XML

2.4.1. Bookstore.xml

genre and lang attributes can be in any order.

<?xml version='1.0'?>

<bookstore xmlns="urn:bookstore-schema">

  <book genre="novel" lang="EnglishUK">

    <title>Book1</title>

    <author>

      <first-name>Name1</first-name>

      <last-name>ABC</last-name>

    </author>

    <price>11.99</price>

  </book>

  <book lang="EnglishUSA" genre="novel">

    <title>Book2</title>

    <author>

      <first-name>Name2</first-name>

      <last-name>EFGHI</last-name>

    </author>

    <price>9.99</price>

  </book>

</bookstore>

2.4.2. Bookstore2.xml

The first book has no title and no price.

<?xml version='1.0'?>

<bookstore xmlns="urn:bookstore-schema">

  <book>

    <author>

      <first-name>Name1</first-name>

      <last-name>ABC</last-name>

    </author>

  </book>

  <book lang="EnglishUSA" genre="novel">

    <title>Book2</title>

    <author>

      <first-name>Name2</first-name>

      <last-name>EFGHI</last-name>

    </author>

    <price>9.99</price>

  </book>

</bookstore>

2.4.3. Bookstore3.xml

The last book author has no first name and last name.

<?xml version='1.0'?>

<bookstore xmlns="urn:bookstore-schema">

  <book genre="novel">

    <title>Book1</title>

    <author>

      <first-name>Name1</first-name>

      <last-name>ABC</last-name>

    </author>

    <price>11.99</price>

  </book>

  <book genre="philosophy">

    <title>Book2</title>

    <author>

      <name>Name2</name>

    </author>

    <price>9.99</price>

  </book>

</bookstore>

2.4.4. Bookstore4.xml

The bookStore must contain at least 2 books

<?xml version='1.0'?>

<bookstore xmlns="urn:bookstore-schema">

  <book genre="novel" lang="EnglishUK">

    <title>Book1</title>

    <author>

      <first-name>Name1</first-name>

      <last-name>ABC</last-name>

    </author>

    <price>11.99</price>

  </book>

</bookstore>

2.4.5. Bookstore5.xml

The lang must be EnglishUK or EnglishUSA

<?xml version='1.0'?>

<bookstore xmlns="urn:bookstore-schema">

  <book genre="novel" lang="EnglishUK">

    <title>Book1</title>

    <author>

      <first-name>Name1</first-name>

      <last-name>ABC</last-name>

    </author>

    <price>11.99</price>

  </book>

  <book lang="Japanese" genre="novel">

    <title>Book2</title>

    <author>

      <first-name>Name2</first-name>

      <last-name>EFGHI</last-name>

    </author>

    <price>9.99</price>

  </book>

</bookstore>

2.4.6. Bookstore6.xml

The last book sub-elements must be in right order.

<?xml version='1.0'?>

<bookstore xmlns="urn:bookstore-schema">

  <book genre="novel" lang="EnglishUK">

    <title>Book1</title>

    <author>

      <first-name>Name1</first-name>

      <last-name>ABC</last-name>

    </author>

    <price>11.99</price>

  </book>

  <book lang="EnglishUSA" genre="novel">

    <price>9.99</price>

    <title>Book2</title>

    <author>

      <first-name>Name2</first-name>

      <last-name>EFGHI</last-name>

    </author>

  </book>

</bookstore>

2.5. Program.cs

using System;

using System.Xml.Linq;

using System.Xml.Schema;

namespace Sample

{

    class Program

    {

        static void Main(string[] args)

        {

            string xsdPathGamers = @"Gamers.xsd"; //load from bin/debug

            string xsdPathBookstore = @"Bookstore.xsd"; //load from bin/debug

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

            // Test Gamers.xml by Gamers.xsd

            Console.WriteLine("1. Test Gamers.xml by Gamers.xsd ================= ");

            XmlValidateByXsd(xsdPathGamers, @"Gamers.xml");

            // xsdPath==Gamers.xsd,xmlPath==Gamers.xml,validation==Passed

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

            // Test Gamers2.xml by Gamers.xsd

            Console.WriteLine("2. Test Gamers2.xml by Gamers.xsd ================= ");

            XmlValidateByXsd(xsdPathGamers, @"Gamers2.xml");

            Console.WriteLine("Fail because more than 4 Gamers");

            // The element 'Gamers' has invalid child element 'Gamer'.

            // xsdPath==Gamers.xsd,xmlPath==Gamers2.xml,validation==Failed

            // Fail because more than 4 Gamers

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

            // Test Gamers3.xml by Gamers.xsd

            Console.WriteLine("3. Test Gamers3.xml by Gamers.xsd ================= ");

            XmlValidateByXsd(xsdPathGamers, @"Gamers3.xml");

            Console.WriteLine("Fail because less than 2 Gamers");

            // The element 'Gamers' has incomplete content. List of possible elements expected: 'Gamer'.

            // xsdPath==Gamers.xsd,xmlPath==Gamers3.xml,validation==Failed

            // Fail because less than 2 Gamers

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

            // Test Gamers4.xml by Gamers.xsd

            Console.WriteLine("4. Test Gamers4.xml by Gamers.xsd ================= ");

            XmlValidateByXsd(xsdPathGamers, @"Gamers4.xml");

            Console.WriteLine("The Score and Id is not in the right position.");

            // 4. Test Gamers4.xml by Gamers.xsd =================

            // The element 'Gamer' has invalid child element 'Score'. List of possible elements expected: 'Id'.

            // xsdPath==Gamers.xsd,xmlPath==Gamers4.xml,validation==Failed

            // The Score and Id is not in the right position.

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

            // Test Bookstore.xml by Bookstore.xsd

            Console.WriteLine("5. Test Bookstore.xml by Bookstore.xsd ================= ");

            XmlValidateByXsd(xsdPathBookstore, @"Bookstore.xml");

            Console.WriteLine("Genere and lang attributes can be in any order.");

            // xsdPath==Bookstore.xsd,xmlPath==Bookstore.xml,validation==Passed

            // Genere and lang attributes can be in any order.

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

            // Test Bookstore2.xml by Bookstore.xsd

            Console.WriteLine("6. Test Bookstore2.xml by Bookstore.xsd ================= ");

            XmlValidateByXsd(xsdPathBookstore, @"Bookstore2.xml");

            Console.WriteLine("The first book has no title and no price.");

            // The element 'book' in namespace 'urn:bookstore-schema' has invalid child element 'author' in namespace 'urn:bookstore-schema'. List of possible elements expected: 'title' in namespace 'urn:bookstore-schema'.

            // xsdPath==Bookstore.xsd,xmlPath==Bookstore2.xml,validation==Failed

            // The first book has no title and no price.

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

            // Test Bookstore3.xml by Bookstore.xsd

            Console.WriteLine("7. Test Bookstore3.xml by Bookstore.xsd ================= ");

            XmlValidateByXsd(xsdPathBookstore, @"Bookstore3.xml");

            Console.WriteLine("The last book author has no first name and last name.");

            // The element 'author' in namespace 'urn:bookstore-schema' has invalid child element 'name' in namespace 'urn:bookstore-schema'. List of possible elements expected: 'first-name' in namespace 'urn:bookstore-schema'.

            // xsdPath==Bookstore.xsd,xmlPath==Bookstore3.xml,validation==Failed

            // The last book author has no first name and last name.

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

            // Test Bookstore4.xml by Bookstore.xsd

            Console.WriteLine("8. Test Bookstore4.xml by Bookstore.xsd ================= ");

            XmlValidateByXsd(xsdPathBookstore, @"Bookstore4.xml");

            Console.WriteLine("The bookStore must contain at least 2 books");

            // The element 'bookstore' in namespace 'urn:bookstore-schema' has incomplete content. List of possible elements expected: 'book' in namespace 'urn:bookstore-schema'.

            // xsdPath==Bookstore.xsd,xmlPath==Bookstore4.xml,validation==Failed

            // The bookStore must contain at least 2 books

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

            // Test Bookstore5.xml by Bookstore.xsd

            Console.WriteLine("9. Test Bookstore5.xml by Bookstore.xsd ================= ");

            XmlValidateByXsd(xsdPathBookstore, @"Bookstore5.xml");

            Console.WriteLine("The lang must be EnglishUK or EnglishUSA");

            // The 'lang' attribute is invalid - The value 'Japanese' is invalid according to its datatype 'String' - The Pattern constraint failed.

            // xsdPath==Bookstore.xsd,xmlPath==Bookstore5.xml,validation==Failed

            // The lang must be EnglishUK or EnglishUSA

            // 10. ====================================

            // Test Bookstore6.xml by Bookstore.xsd

            Console.WriteLine("10. Test Bookstore6.xml by Bookstore.xsd ================= ");

            XmlValidateByXsd(xsdPathBookstore, @"Bookstore6.xml");

            Console.WriteLine("The last book sub-elements must be in right order.");

            // The element 'book' in namespace 'urn:bookstore-schema' has invalid child element 'price' in namespace 'urn:bookstore-schema'. List of possible elements expected: 'title' in namespace 'urn:bookstore-schema'.

            // xsdPath==Bookstore.xsd,xmlPath==Bookstore6.xml,validation==Failed

            // The last book sub-elements must be in right order.

            Console.ReadLine();

        }

        private static void XmlValidateByXsd(string xsdPath, string xmlPath)

        {

            //Load xsd

            XmlSchemaSet xmlSchemaSet = new XmlSchemaSet();

            xmlSchemaSet.Add(null, xsdPath);

            //Load xml

            XDocument xmlDocument = XDocument.Load(xmlPath);

            //Validate Error

            bool validationErrors = false;

            xmlDocument.Validate(xmlSchemaSet, (sender, eventArgs) =>

            {

                Console.WriteLine(eventArgs.Message);

                validationErrors = true;

            });

            // if xmlDocument does NOT pass the validation of xmlSchemaSet,

            // then it will run the anonymous methods.

            string validationStr = validationErrors? "Failed": "Passed";

            Console.WriteLine($"xsdPath=={xsdPath},xmlPath=={xmlPath},validation=={validationStr}");

        }

    }

}