Lecture 1: 使用檔名：myXml_0010.xml & mySchema_0010.xsd

(Using **Schema** to validate their XML documents.)

What is a schema? A schema is any type of model document that defines the structure of something.

一、使用 myXml_0010.xml 檔。

```
<?xml version="1.0" encoding="Big5"?>
<!--      my first Xml file    myXml_0010.xml -->
<book xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ntou.edu.tw/XML mySchema_0010.xsd" >
      <title>通訊原理與應用</title>
      <author>藍國桐</author>
      <page>288 頁</page>
      <price>NT$ 350</price>
</book>
```

先產生一個 namespace (xsi) 指向 W3C，用這個 xsi 的屬性：schemaLocation 指向一個 Schema file : mySchema_0010.xsd

屬性：schemaLocation 必須包括兩資料：“namespace file-location”。namespace 為本 xml (該 xsd 檔吧??)檔所形成的 namespace (default namespace) 而 file-location 則是描述本 xml 檔的 schema 檔的位置 URL。

說明一、注意本 xml 檔中並沒有設 name space，所以在本範例中的 tag name (如 book, title, author, page, price) 都是 non qualified (即 非 Qname).

說明二、The schema itself is actually a special kind of XML document- specifically, it’s an XML document that is written according to the rules given in the W3C Schema specification. These rules constitute a language, known as the XML Schema definition language. A particular XML document that conforms to the strictures of a schema is known as an instance document of that schema.

二、再 edit 一個 mySchema_0010.xsd 檔。

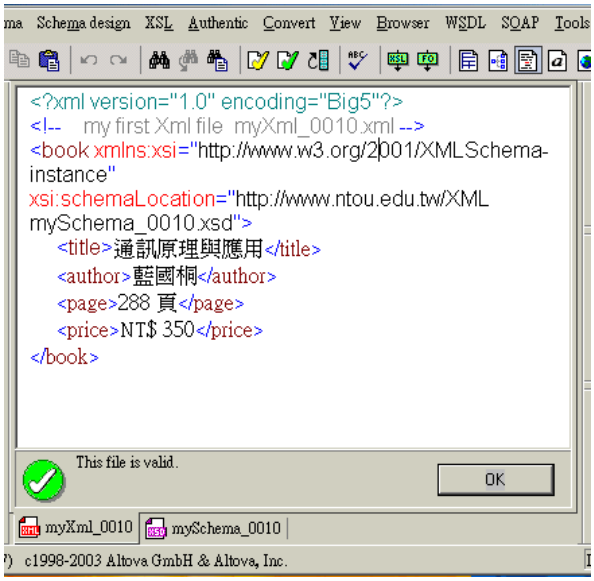
```
<?xml version="1.0" ?>
<!--   my first Schema file mySchema_0010.xsd    -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
    <xsd:element name="book">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="title" />
```

```
                    <xsd:element name="author" />
                    <xsd:element name="page" />
                    <xsd:element name="price" />
                </xsd:sequence>
            </xsd:complexType>
    </xsd:element>
</xsd:schema…….>
```

執行結果(Demo)：

(要驗證一個 xml 及其相對的 schema，必需要有一個 validator，在此我們採用 XMLSPY)

(一般的 browser 不會去作 validating, 所以 IE 不會去向 web server 要 Schema file (.xsd))



說明三、The XML Schema Namespace:

We declared the namespace within our <schema> element. This allows us to indicate that the <schema> element is part of the XML Schema vocabulary. Remember, because XML is case-sensitive, namespaces are case-sensitive.

For example, you could use any of the following <schema>:

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

Default namespace.

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

prefixed namespace.

說明四、The targetNamespace:

The primary purpose of XML Schemas is to declare vocabularies. These vocabularies can be identified by a namespace that is specified in the targetNamespace attribute. It is important to realize that not all XML Schemas will have a targetNamespace. Many XML Schemas define vocabularies that will be reused in another XML Schema, or vocabularies that will be used in documents where the namespace is not necessary.

When declaring a targetNamespace, it is important to include a matching namespace declaration. Some possible targetNamespace declarations include:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ntou.edu.tw/XML"
xmlns:target="http://www.ntou.edu.tw/XML" >
```

W3C 的 vocabularies (name space) 被視為 default 值。我們定義的vocabularies 必需加上的prefix 是 “target”。

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ntou.edu.tw/XML"
xmlns="http://www.ntou.edu.tw/XML" >
```

W3C 的 vocabularies (name space) 必需加上的prefix 是 “xsd”。我們定義的vocabularies 被視為 default 值。

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ntou.edu.tw/XML"
xmlns:target="http://www.ntou.edu.tw/XML" >
```

W3C 的 vocabularies (name space) 必需加上的prefix 是 “xsd”。我們定義的vocabularies 必需加上的prefix 是 “target”。

targetNamespace 所指定的 value 就是本 schema 內部所定義的各種 element 及 type 的 name space。TargetNamespace 替本 schema 定義了一個 name space。

說明五、The elementFormDefault & attributeFormDefault:

The elementFormDefault and attributeFormDefault attributes allow you to control the default qualification form for elements and attributes in the instance documents. An element or attribute is qualified if it has an associated namespace URI. If you use a default name space declaration, the element is qualified but no prefixed. On the other hand, if you use a prefixed name space declaration, the element is qualified and prefixed.

The default value for both elementFormDefault and attributeFormDefault is unqualified.

Most of your documents should qualify all of the their elements. In some cases, you will need to create a document that uses both qualified and unqualified elements. For example, XSLT and SOAP documents may contain both qualified and unqualified elements.

If your XML Schema has no targetNamespace you must refer to the XML Schema using the noNamespaceSchemaLocation attribute within our instance document.

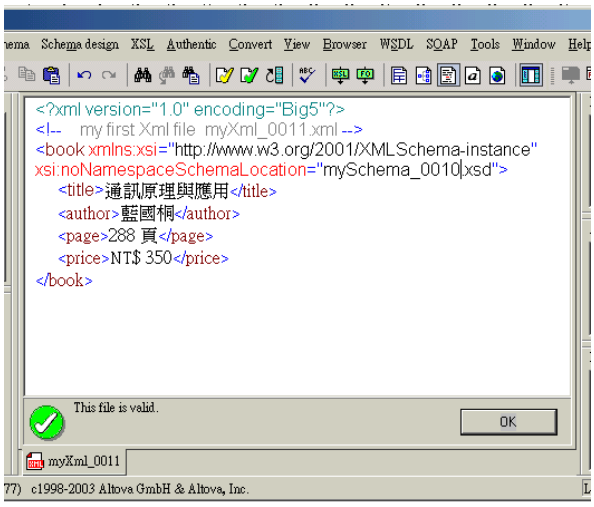
For example: myXml_0011.xml

```
<?xml version="1.0" encoding="Big5"?>
<!--       my first Xml file    myXml_0011.xml -->
<book xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="mySchema_0010.xsd" >
        <title>通訊原理與應用</title>
        <author>藍國桐</author>
        <page>288 頁</page>
        <price>NT$ 350</price>
</book>
```

屬性：noNamespaceSchemaLocation 只包括一項資料：描述本 xml 檔的 schema 檔的位置 URL。

這個 noNamespaceSchemaLocation 是說 該 xsd 檔雖有定義了一些 element (attribute) type 但是它本身並不形成一個 Namepspace。

執行結果(Demo)：



對 XML 檔而言，若其 Schema 檔中並沒有設 targetNamespace 這個屬性，則在 XML 檔中可以設 noNamespaceSchemaLocation (表示 該 xsd 檔並沒有形成一個 NameSpace)，這個屬性只要設 Schema 檔所在的 file-location 即可！但若其 Schema 檔中有設了 targetNamespace 則在本 XML 檔中一定要設 schemaLocaton 這個屬性。因為 schemaLocation 必須給定 namespace 及 file-location！

Q1: 由 myXml_0010.xml 知道：雖然 schema 檔中沒有設 targetNamespace 但 XML 檔仍可以設 schemaLocation。

Q2: 若 Schema 檔中已經有設 targetNamespace 則 XML 檔中是否一定要設 schemaLocation 或允許設 noNamespaceSchemaLocation？

Lecture 2: 使用檔名：myXml_0015.xml & mySchema_0015.xsd

一、 myXml_0015.xml。

```
<?xml version="1.0" encoding="Big5"?>
<!--       my first Xml file    myXml_0015.xml -->
<book xmlns="http://www.ntou.edu.tw/XML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ntou.edu.tw/XML mySchema_0015.xsd">
        <title>通訊原理與應用 </title>
        <author>藍國桐</author>
        <page>288  頁</page>
        <price>NT$ 350</price>
</book>
```

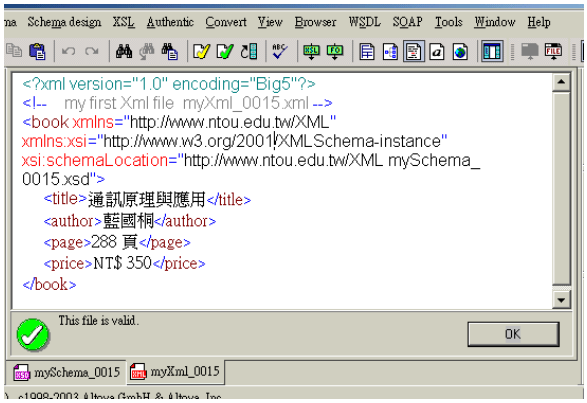
二、mySchema_0015.xsd。

```
<?xml version="1.0"?>
<!--    my first Schema file mySchema_0015.xsd    -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ntou.edu.tw/XML"
xmlns:target="http://www.ntou.edu.tw/XML" elementFormDefault="qualified">
        <xsd:element name="book">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element name="title"/>
                                <xsd:element name="author"/>
                                <xsd:element name="page"/>
                                <xsd:element name="price"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
</xsd:schema>
```

Global declaration

Local declaration

執行結果(Demo)：



說明一、The <element> :

When declaring an element in the schema file, we are actually performing two primary tasks: specifying the element name and defining the allowable content.

<element

name= “name of the element”

type= “global type in the XSD file or namespace”

ref= “global element declaration”

form= “qualified or unqualified”

minOccurs= “non negative number”

maxOccurs= “non negative number or ‘unbounded’ ”

default= “default value”

fixed= “fixed value” >

XML Schema declarations can be divided intotwo broad categories: global declarations and local declarations. **Global declararions** are decalrations that appear as direct children of the <schema> element. Global element decalrations can be reused throughout the XML Schema. **Local declarations** do not have the <schema> element as their drirect parent and are only valid in their specific context.

說明二、Creating a Local Type :

To create a local type, you simply include the type declaration as a child of the element declaration:

<element name= “book”>

<complexType>

<!-- type information -->

```
    </complexType>
</element>
```

說明三、Using a Global Type :

Often, many of our elements will have the same content. Instead of declaring duplicate local types throughout our schema, we can create a global type. In the following examples, our element declaration refers to a global type named string. This type is a member of the XML Schema vocabulary.

```
<schema xmlns ="http://www.w3.org/2001/XMLSchema"      …>
    <element name="book" type="string" />
```

default NameSpace

or,

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"      …>
    <xsd:element name="book" type="xsd:string" />
```

prefixed NameSpace

三、mySchema_0016.xsd。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ntou.edu.tw/XML"
xmlns:myNS="http://www.ntou.edu.tw/XML"
elementFormDefault="qualified">
    <xsd:complexType name="nameType">
        <xsd:sequence>
            <xsd:element name="title" type="xsd:string"/>
            <xsd:element name="author" type="xsd:string"/>
            <xsd:element name="page" type="xsd:string"/>
            <xsd:element name="price" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="book" type="myNS:nameType"/>
</xsd:schema>
```

直接使用某個 complexType type="myNS:nameType" 只能用某一種 global type。

宣告有一種 element 其 type 是由 myNS 定義的 nameType !!

四、mySchema_0017.xsd。

```
<?xml version="1.0" encoding="UTF-8"?>
<!--       my first Xml file    mySchema_0017.xsd -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.ntou.edu.tw/XML"
xmlns:myNS="http://www.ntou.edu.tw/XML"
elementFormDefault="qualified">
        < xsd:element name="title" type=" xsd:string"/>
        < xsd:element name="author" type=" xsd:string"/>
        < xsd:element name="page" type=" xsd:string"/>
        < xsd:element name="price" type=" xsd:string"/>
        < xsd:complexType name="nameType">
            < xsd:sequence>
                < xsd:element ref="myNS:title"/>
                < xsd:element ref="myNS:author" />
                < xsd:element ref="myNS:page"/>
                < xsd:element ref="myNS:price" />
            </ xsd:sequence>
        </ xsd:complexType>
        < xsd:element name="book" type="myNS:nameType"/>
</ xsd:schema>
```

參考了數個 global type 的 element 以形成本 complexType。
所以本 complexType 內的 element 均是 global type！

說明四、Referring to an Existing Global element :

Referring to global types allows us to easily reuse content model definitions within our XML Schema. XML Schemas allow you to reuse global element declarations within your content model. To refer to a global element declaration, simply include a *ref* attribute and specify the name of the element as the value.

Lecture 3: Declaring an Element with a Complex Type

A complex type can allow an element to contain one or more child element or attributes, in adition to character data. An element declared with element content can contain one or more child elements of the specified type or types, as well as attributes. To declare an element with element content, you define the element's type using *xsd:complexType* and include within it a *content model* that describes the permissible child elements(1), the allowed arrangement of these elements(2), and the rules for their occurrences(3). You create the content model by using the *xsd:sequence*, *xsd:choice*, or *xsd:all* schema element, or a bombination of these elements.

1. A group of child elements decalred in an *xsd:sequence* element must appear in the exact order listed.

(Demo) 利用 *myXml_0020*.xml 及 *mySchema_0020*.xsd，將 mySchema_0020.xsd 中的 price element 刪去，再看看是否你所收到的myXml_0020.xml 仍為 valid？將 price 及 page element 掉轉順序，test if it is valid or not?

2. if a group of child elements is declared within an *xsd:choice* schema element, any one of the child elements can appear within the parent element.
(只能有一個 element 存在！(Demo) try *myXml_0021*.xml and *mySchema_0021*.xsd)
(因為 choice 的預設的 *maxOccurs=1*，若將 *maxOccurs* 設成較大的值或是 "unbound"，則在 xml 中可以出現數個 element 而且同名的element 才可以重複出現)

3. if you declare a group of child elements within an *xs:all* schema element, the child elements can occur in any order. By default, each child element must occur exactly once. (順序沒有關係，但都必須只出現一次) However, you can make a particular child element optional by including the attributes specification *minOccurs= "0"* in the *xsd:element* start-tag. For an elemetn in an *xsd:all* group, you can assign the *minOccurs* or *maxOccurs* attribute only the value 0 or 1. (加上 attribute *minOccurs= "0"* 則該 element 可以不用出現。是加到個別的element 上面，而不是加到 all 上面。就上述三個 complexType 而言，只有 choice 設 maxOccurs/minOccurs 才有用處。在 all/ sequence 中設 maxOccurs/minOccurs 似乎都無效！但可以將 maxOccurs/minOccurs 設在 element 中！)

(Demo) 利用 *myXml_0022*.xml 及 *mySchema_0022*.xsd，test above description to see if the XML file is valid? (刪掉 <title> 但 XML 文件仍是valid ！但刪掉 <pirce> 則會變成 invalid！因為 <title> 在 Schema 中的定義是 *minOccurs="0"*)

4. To create a more complex content mode, you can nest an *xsd:sequence* element within an *xsd:choice* element, or an *xsd:choice* element within and *xsd:sequence* element. (You can't nest an *xsd:all* element within an *xsd:sequence* or *xsd:choice* element. Nor can you nest an *xsd:sequence* or *xsd:choice* element within an *xsd:all* element. 這個論點有些疑問？ Demo Check *mySchema_0023*.xsd/ *myXml_0023*.xml)

作一些 sample codes

5. sequence, choice, all 所形成的 nesting 結構：

<xsd:choice> … … … <xsd:sequence> … … … </xsd:sequence> … … … </xsd:choice>	<xsd:choice> … … … <xsd:choice> … … … </xsd:choice> … … … </xsd:choice>	<xsd:sequence> … … … <xsd:sequence> … … … </xsd:sequence> … … … </xsd:sequence>
<xsd:sequence> … … … <xsd:choice> … … … </xsd:choice> … … … </xsd:sequence>	<xsd:sequence> … … … <xsd:all> … … … </xsd:all> … … … </xsd:sequence>	<xsd:all> … … … <xsd:sequence> … … … </xsd:sequence> … … … </xsd:all>

<xsd:choice> … … … <xsd:all> … … … </xsd:all> … … … </xsd:choice>	<xsd:all> … … … <xsd:choice> … … … </xsd:choice> … … … </xsd:all>	

△ A sequence can also contain other sequence, choices, or references to named group.

△ A sequence maybe contained in a complex type definition, in other sequences, or in a set of choices or in a named group definitions.

△ A set of choices can also contain nested sequences, additional choice sets, or references to named group.

△ A set of choices maybe contained in a complex type definition, in sequences, in other sets of choices, or in named group definitions.

Lecture 4: Declaring an Element with a Simple Type

A simple type can permit the element to contian only character data (i.e., text node in DOM), whereas a complex type can allow the element to contain one or more child elements or attributes in addition to character data. To declare an element (or attribute) with a simple type, you can use a built-in simple type- that is, one defined as a part of the XML Schema definition laguage. Or, you can use a new simple type that you define by deriving it from an existing simple type. For example,

```
<xsd:element name="author" type="xsd:string"/>
```

The following table describes a smapling of some useful built-in simple types.

Built-in simple Type	Description	Example(s)
xsd:string		
xsd:boolean		true/ false/1/0
xsd:decimal		-5.3, 1, 2.5
xsd:integer		-389,-7, 5
xsd:positiveInteger		5, 229, 389
xsd:negativeInteger		-5, -29,-389
xsd:date		1948-05-21, 2003-10-15
xsd:time		11:30:00.00
xsd:gYear		2001, 1953
xsd:gDay		---05, ---31
xsd:gYearMonth		1948-05

在 Schema file 中定義了 title element 如下：

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

則在 XML file 中使用這個 element 必需遵守 string 的規定：

```
<title>通訊原理與應用</title>
```

在 Schema file 中定義了 author element 如下：

```
<xsd:element name="author" type="xsd:string"/>
```

則在 XML file 中使用這個 element 必需遵守 string 的規定：

```
<author>藍國桐</author>
```

在 Schema file 中定義了 page element 如下：

```
<xsd:element name="page" type="xsd:positiveInteger"/>
```

則在 XML file 中使用這個 element 必需遵守 positiveInteger 的規定：

```
<page>288</page>
```

在 Schema file 中定義了 price element 如下：

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

則在 XML file 中使用這個 element 必需遵守 decimal 的規定：

```
<price>350</price>
```

在 Schema file 中定義了 publishDate element 如下：

```
<xsd:element name="publishDate" type="xsd:gYearMonth" minOccurs="0"/>
```

則在 XML file 中使用這個 element 必需遵守 gYearMonth 的規定：

```
<publishDate>2001-05</publishDate>
```

在 Schema file 中定義了 onSale element 如下：

```
<xsd:element name="onSale" type="xsd:boolean" minOccurs="0"/>
```

則在 XML file 中使用這個 element 必需遵守 boolean 的規定：

```
<onSale>true</onSale>
```

■ You can control the number of occurrences of the element within the context where it is declared by including the *minOccurs* attribute (the minimum number of occurrences). The *maxOccurs* attribute (the maximum number of occurrences), or both attributes.

```
<xsd:element name="references" type="xsd:positiveInteger" minOccurs="0"
maxOccurs="unbounded"/>
```

(就算這個 simpleType 是屬於某個 sequence complexType，在 element 中設 *minOccurs*/*maxOccurs* 仍然會發生效用)

(Demo) myXml_0030.xml and mySchema_0030.xsd

Lecture 4: User-defined **Simple Type**

■ **A Defined Simple Type**: You can use a new simple type that you define by deriving it from one of the built-in simple types (or from another derived simple type already defined in the schema).

Table 2. Simple Type Construction Techniques

Derivation Element	Description
xsd:restriction	The new type is a restriction of the existing type, which means it has a narrower set of legal values.
xsd:list	The new type is a whitespace-delimited list of another simple type.
xsd:union	The new type is a union of two or more other simple types.

1. To restrict the value space by using *xsd:restriction*

(The new type is a restriction of the existing type, which means it has a narrower set of legal values.)

xsd:minExclusive & xsd:maxExclusive

例如：在 Schema file 中定義了 price element，而且規定其值介於 0～500，則將

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

改成；

```
<xsd:element name="price">
    <xsd:simpleType>
        <xsd:restriction base="xsd:decimal">
            <xsd:minExclusive value="0" />
            <xsd:maxExclusive value="500" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

The example uses the *xsd:minExclusive* and *xsd:maxExclusive* facets to indicate a permissible range of values that doesn't include the specified end values (0 and 500). To indicate a range that does include the specified end values, you can use the similar

xsd:minInclusive and *xsd:maxInclusive* facet elements.

(Demo) myXml_0031.xml and mySchema_0031.xsd

xsd:enumeration

You can use a series of *xsd:enumeration* facets to limit the element's content (or attribute's value) to one of a set of specific values.

例如：在 Schema file 中定義了 author element，而且規定其值僅限幾個值，則將

```
<xsd:element name="author" type="xsd:string"/>
```

改成；

```
<xsd:element name="author">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="藍國桐" />
            <xsd:enumeration value="王李吉" />
            <xsd:enumeration value="涂相麟" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

(Demo) myXml_0032.xml and mySchema_0032.xsd

You can require that an element's content (or an attribute's value) match a particular pattern of characters by restricting the *xsd:string* type using the *xsd:pattern* facet.

xsd:pattern

例如：在 Schema file 中定義了 ISBN element，而且規定其值需符合 ISBN 的編碼規定，則將其寫成：

```
<xsd:element name="ISBN" minOccurs="0">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\d{1}-\d{4}-\d{4}-\d{1}" />
```

```
            </xsd:restriction>
        </xsd:simpleType>
</xsd:element>
```

則在 XML file 中使用這個 element 必需遵守 xsd:pattern 的規定：

```
<ISBN>0-7355-1020-9</ISBN>
```

例如：

```
<xsd:simpleType name="Phone">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\(\d{3}\)\d{3}-\d{4}" />
    </xsd:restriction>
</xsd:simpletype>
```

則在 xml file 中：

```
<x:phone xmlns:x="http://example.org/putlishing">(801)390-4552</x:phone>
```

(Demo) myXml_0030.xml and mySchema_0030.xsd

For a description of the regular expressions you can use with the xsd:pattern element, see Appendix D "**Regular Expressions**" in the "*XML Schema Part 0: Primer*" page at http://*www.w3.org/TR/xmlschema-0/*.

XML Schema defines the facets available for each type. Most facets don't apply to all types (some only make sense on certain types). Most facets restrict a type's value sapce while the *pattern* facet restricts the type's lexical space. Only strings matching the regular expression (specified in the pattern facet) are considered a valid instance of the given type.

The following table describes some useful facets for each **restrict types**.

Facet Element	Description
xsd:enumeration	Specifies a fixed value that the type must match.
xsd:maxExclusive	Specifies the exclusive upper-bound on the value space of the type.
Xsd:maxInclusive	Specifies the inclusive upper-bound on the value space of the type.
Xsd:maxLength	Spevifies the maximum number of characters in a string-based type, the maximum number of octets in a

	binary-based type, or the maximum number of items in a list-based type.
xsd:minExclusive	
xsd:minInclusive	
xsd:minLength	Specifies the minimum number of characters in a string-based type, the number of items in a list-based type.
xsd:pattern	Specifies a pattern, based on a regular expression, the type must match.
xsd:length	Specifies the number of characters in a string-based type, the number of octets in a binary-based type, or the number of items in a list-based type.
xsd:factionDigits	Specifies the maximum number of decimal digits to the right of the decimal point.
xsd:totalDigits	Specifies the maximum number of decimal digits for types derives from number.
xsd:whiteSpace	Sprcifies rules for whitespace normalization.

(使用 xsd:pattern 時要注意：xml 中該 element 的內容不能任意用 whitespace)

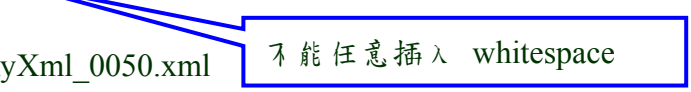
In XSD:

```
<xsd:simpleType name="Phone">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="\(\d{2}\)\d{4}-\d{4}"/>
   </xsd:restriction>
</xsd:simpleType>
```

In XML:

```
<phone xmlns="http://www.ntou.edu.tw/XML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ntou.edu.tw/XML
mySchema_0050.xsd">(02)8662-5925</phone>
```

(Demo) mySchema_0050.xsd and myXml_0050.xml



2. To list the value space by using *xsd:list*

(The new type is a whitespace-delimited list of another simple type. 相同類型的資料放在一起，在 XML 檔案中，同類型的資料可以重複出現)

In addition to restricting a type's values space, it's also possible to construct new simple types that are lists or unions of other simple types. To do this you use either the *xsd:list* or *xsd:union* element instead of *xsd:restriction*. When using *xsd:list*, you're essentially defining a whitespace-delimited list of values form the specifies value space.

In XSD: (xsd:list 要使用 itemType 這個屬性)

```
<xsd:simpleType name="AuthorIDList">
   <xsd:list itemType="myNS:SSN"/>
</xsd:simpleType>
<xsd:element name="authorIDs" type="myNS:AuthorIDList"/>
```

In XML:

```
<authorIDs xmlns="http://www.ntou.edu.tw/XML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ntou.edu.tw/XML mySchema_0050.xsd"
>123-01-0021 123-01-0022 123-01-0024</authorIDs>
```

(Demo) mySchema_0050.xsd and myXml_0051.xml

3. To display the value space by using *xsd:union*

(The new type is a union of two or more other simple types. 不同類型的資料放在一起，但是在 XML 檔案中，資料只能出現一個)

In the case of *xsd:union*, you're creating a new type that combines multiple value spaces into a new value space. An instance of a union type can be a value from any of the specified value spaces.

In XSD: (xsd:union 要使用 memberTypes 這個屬性)

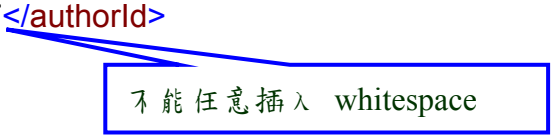
```
<xsd:simpleType name="AuthorIdType">
   <xsd:union memberTypes="myNS:SSN myNS:PublisherAssignedId"/>
</xsd:simpleType>
<xsd:element name="authorId" type="myNS:AuthorIdType"/>
```

In XML:

```
<authorId xmlns="http://www.ntou.edu.tw/XML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ntou.edu.tw/XML mySchema_0050.xsd"
>123-01-0021</authorId>
```

或者是：

```
<authorId xmlns="http://www.ntou.edu.tw/XML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ntou.edu.tw/XML mySchema_0050.xsd"
>22-23242567</authorId>
```



(Demo) mySchema_0050.xsd and myXml_0052.xml

Lecture 5: Examples

Ex.1

```
<xsd:element name="FILM">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="TITLE" type="xsd:string" />
                <xsd:element name="CLASS">
                    <xsd:simpleType >
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="fictional" />
                            <xsd:enumeration value="documentary" />
                            <xsd:enumeration value="instructional" />
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:choice>
                    <xsd:element name="STAR" type="xsd:string" />
                    <xsd:element name="NARRATOR" type="xsd:string" />
                    <xsd:element name="INSTRUCTOR" type="xsd:string" />
                </xsd:choice>
            </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

xml Instance:

```
<FILM>
    <TITLE>The Net</TITLE>
    <CLASS>fictional</CLASS>
    <STAR>Sandra Bullock</STAR>
</FILM>
```

Ex.2 Defining Element with Mixed Content

To specify a mixed content element, declare the element with element content exactly as described in the previous section, but add the *mixed= "true"* attribute specification to the *xsd:complexType* element's start-tag.

```
<xsd:element name="Title">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="SUBTitle" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

xml Instance:

<Title> Moby-Dick <SUBTitle> Or, The Whale </SUBTitle> </Title>

Ex.3

```
<xsd:element name="PARA">
    <xsd:complexType mixed="true">
        <xsd:choice minOccurs="0" maxOccurs="unbounded" >
            <xsd:element name="BOLD" type="xsd:string" />
            <xsd:element name="ITALIC" type="xsd:string" />
            <xsd:element name="UNDERLINE" type="xsd:string" />
        </xsd:choice>
    </xsd:complexType>
</xsd:element>
```

xml Instance:

<PARA> A PARA element may contain character data plus any number of nested elements for marking <ITALIC>italic</ITALIC> text, <BOLD>boldface</BOLD> text, or <UNDERLINE>underlined</UNDERLINE> text. These nested elements can be used in any order. </PARA>

Ex.4 Defining Empty Elements

An element declared with empty content cannot contain either child elements or character data. (It can have attributes if they are declared.) To define the element's type using the *xsd:complexType* element, but omit the content model.

Schema: `<xsd:element name="BR"></xsd:element>`

Instance: `<BR />`

To declare an empty content model in a <complexType> definition, you simply create the <complexType> definition without any <element> or content model declarations.

Schema:

```
<xsd:element name="BR">
    <complexType></complexType>
</xsd:element>
```

Instance: `<BR />`