Lecture 1: Defining **attribute** type

XML also supports attributes. As stated earlier, elements require begin and end tags. Attributes do not. Instead, they are contained by the begin tag of an element.
A given element can have one or more elements of the same type. It can only have one attribute of any given type.

<Library> <Book title="Windows Programming"> <author>Scott</author> </Book> </Library>	<Library> <Book title="Window Programming" author="Scott" /> </Library>
<Library> <Book title="Windows Programming" author="Scott" author="Paul"> </Book> </Library>	<Library> <Book title="Windows Programming" > <author name="Scott" /> <author name="Paul" /> </Book> </Library>

When creating attributes for an XML Schema, you use the attribute keyword. This word only has meaning within the schema namespace. You use attribute to define characteristics of the type.

Compositors of a complexType contain particles, which includes things like other compositors, element declarations, wildcards, and model groups. ***Attribute*** declarations are not considered particles because they don't repeat. Hence, ***attribute*** declarations are not placed within a compositor but after the compositor at the end of the complex type definition.

In XSD:

```
<?xml version="1.0"?>
<!--  my first Schema file mySchema_0040.xsd  -->
<xsd:schema targetNamespace="http://www.ntou.edu.tw/XML"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:myNS="http://www.ntou.edu.tw/XML" elementFormDefault="qualified">
```

```
    ……….
    <xsd:element name="author">
        <xsd:complexType mixed="true">
        <xsd:attribute name="age" type="myNS:priceType"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:simpleType name="priceType">
      ……….
    </xsd:simpleType>
    <xsd:element name="book">
    ……….
    </xsd:element>
    ……….
    <xsd:attribute name="ageType" type="myNS:priceType"/>
    ……….
</xsd:schema>
```

In XML Instance:

```
<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_0040.xsd">
    <author age="40">藍國桐 </author>
    <title>通訊原理與應用 </title>
    <page>288</page>
    <price>350</price>
</book>
```

(Demo) myXml_0040.xml and mySchema_0040.xsd

XML Schema makes it possible, however, to explicitly control whether a given local element/attribute should be qualified or unqualified using the form attribute on *xsd:element*/*xsd:attribute* or by using the *elementFormDefault*/ *attributeFormDefault* attributes on *xsd:schema*.

```
<xsd:schema xmlns:xsd:”http://www.w3.org/2001/xmlSchema”
    targetNamespace=”http://example.org/publishing”
    xmlns:tns=”http://exmple.org/publishing”
    elementFormDefault=”qualified”
    attributeFormDefault=”qualifited
>
…
</xsd:schema>
```

local type 的 element 才 需要作 qualify
即使是 local type 的 attribute 才 需要作 qualify

With this schema in place, the following instance would be considered a valid instance (while the previous instance wouldn't be):

```
<x:author xmlns:x=”http://example.org/publishing”
    x:id=”333-33-3333”
>
    <x:name>Aaron Skonnard</x:name>
    <x:phone>(801)390-4552</x:phone>
</x:author>
```

注意：

<x:author xmlns:x="http://www.ntou.edu.tw/XML" ……..
x:id="333-33-3456"> → 表示這個 attribute id 有被 qualified
→ element author 有 qualified

<x:author xmlns:x="http://www.ntou.edu.tw/XML" ……..
id="333-33-3456"> → 表示這個 attribute id 沒有被 qualified
→ element author 有 qualified

<author xmlns="http://www.ntou.edu.tw/XML" ……..
id="333-33-3456"> → 表示這個 attribute id 沒有被 qualified
(attribute 似乎沒有 預設 prefix 的功能 ??)
→ element author 有 qualified

advanced <attribute>

structure of <attribute> :

```
<attribute
    name="name of the attribute"
    type="global type"
    ref="global attribute declaration"
    form="qualified or unqualified"
    use="optional or prohibited or required"
    default="default value"
    fixed= "fixed value"
>
```

creating a Local Type:

```
<xsd:attribute name="priceType">
    <xsd:simpleType>
      ..........................
    </xsd:simpleType>
</xsd:attribute>
```

△ 宣告 一個 attribute type 只能用 simpleType

△ Requried an Attribute:

Unless you specify otherwise as you are defining an attribute, an attribute is always optional. In other words, it may appear or be absent from the XML document. However, you can insist that an attribute be present or not when determining if the document is valid.

```
<xsd:complexType name="priceType">
    <xsd:sequence>
      ..........................
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:strin
                   used="required"  />
    <xsd:attribute name="nameID" type="xsd:decimal"
                   value="101"  />
</xsd:complexType>
```

表示這個 attribute 是必要的！

表示這個 attribute 的值一定是 101！

(Demo) myXml_0041.xml and mySchema_0041.xsd

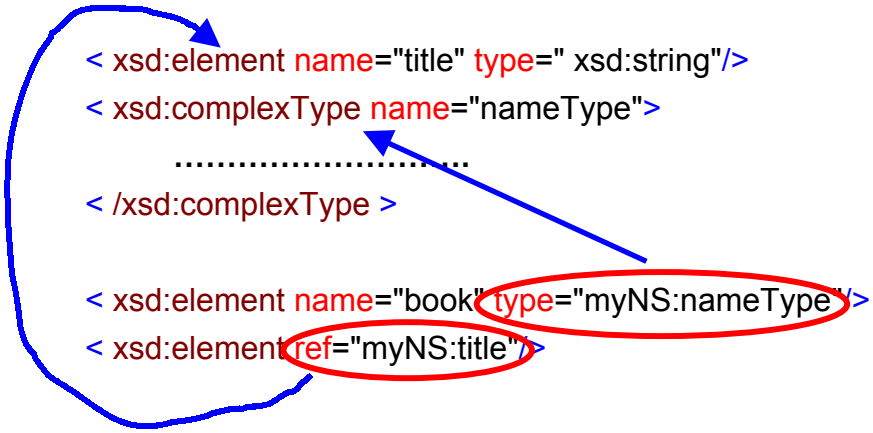
有錯誤！ <attribute> 和<element> 應該是相反的：
type 引用 complexType/simpleType
ref 引用 element/attribute/group/attributeGroup

ype" />

ageType" use="required" />

上述兩種方法 效果相同

(Demo) myXml_0042.xml and mySchema_0042.xsd



Lecture 2: Defining **attributeGroup** type

In addition to element groups, XML Schema also allows us to define attribute groups.

```
<attributeGroup
      name= "name of global attribute group">
- attribute declarations -
```

It is easier in to create a global attribute group that can be resued in out <complexType> definition. The basic structure of a global <attributeGroup> declaration follows:

```
<attributeGroup name= “NameAttGroup”>
      <attribute name=”name1” type=”xsd:string” />
      <attribute name=”name2” type=”xsd:string” />
       ……….
</attributeGroup>
```

將 幾個 attribute 形成 attributeGroup。

(Demo) myXml_0043.xml and mySchema_0043.xsd

注意：只能用 <attributeGroup ref= “…”> 去 reference <attributeGroup name= “…”> 或是用 <attribute ref= “…”> 去 reference <attribute name= “…”>。

Lecture 3: Using xsd:Group

△ An all group can only contain individual element declarations or references, not other group. In addition, no element may appear more than once.

△ An all group can only be contained- and must be the sole child- of either a complex type definition of a name group definition.

Defining Named Group:

If a collection of elements appears together in several places in your XML document, you can group the element together (like group the attributes) in order to make it easier to refer to them all at once.

```
<xsd:group name="bookGroup">
   <xsd:all>
         <xsd:element name="author" type="myNS:authorType" />
         <xsd:element name="title" type="xsd:string" />
         <xsd:element name="page" type="xsd:positiveInteger" />
         <xsd:element name="price" type="myNS:priceType" />
   </xsd:all>
</xsd:group>
```

注意：<group> 只能 group 著 element ，不能 group 著 attribute。

Referencing a Named Group:

Once your're created a group, you can reference it in other groups or in complex type definitions.

```
<xsd:element name="book">
   <xsd:complexType>
      <xsd:group ref="myNS:bookGroup"/>
      <xsd:attribute name="ISBN" type="xsd:string" />
   </xsd:complexType>
</xsd:element>
```

可以自行加入 attribute

(Demo) myXml_0060.xml and mySchema_0060.xsd

(Demo) myXml_0062.xml and mySchema_0062.xsd

Lecture 4: Using xsd:extension

<xsd:simpleContent> & <xsd:extension> 的使用：

心須是一個 simple Type，所以用 simpleContent 來加以擴充！

```
<!-- type defineitions- complexType  -->
   <xsd:complexType name="authorType">
      <xsd:simpleContent>
         <xsd:extension base="myNS:authorSelect">
            <xsd:attribute name="age" type="myNS:priceType"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
……..……
   <xsd:element name="author" type="myNS:authorType"/>
```

(Demo： myXml_0070.xml 及 mySchema_0070.xsd)

Using <xsd:simpleContent> & <xsd:extension> expand the simple Type to include a set of attributes.
<xsd:simpleContent> & <xsd:extension> 將一個 simple Type 擴充成一個 complex Type.
If what you rally want is a given simple type- whose content is defined to be a specific kind of text- but with attributes, you can derive a text-only complex type to fit the bill.

<xsd:complexContent> & <xsd:extension> 的使用：

Basing complex Types on complex Type:
You can create complex types from existing complex types. The new complex type begins with all the information from the existing type, and then adds or removes features.

```
   <xsd:complexType name="bookDescribType">
      <xsd:all>
         <xsd:element name="author" type="myNS:authorType"/>
         <xsd:element name="title" type="xsd:string"/>
```

```
        <xsd:element name="page" type="xsd:positiveInteger"/>
        <xsd:element name="price" type="myNS:priceType"/>
      </xsd:all>
    </xsd:complexType>
……..……
    <xsd:element name="book">
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="myNS:bookDescribType">
            <xsd:sequence>
              <xsd:element name="publish_date"
                                type="xsd:gYearMonth"/>
              <xsd:element name="publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>
```

表示除了原有的 bookDescribType 之外，另外加上一個 sequence 的 type！所以共有 6 個 element！

(Demo：先 review: myXml_0060.xml 及 mySchema_0060.xsd)
(Demo：再 review: myXml_0070.xml 及 mySchema_0070.xsd)
(Demo： myXml_0071.xml 及 mySchema_0071.xsd)

△ 表示除了原有的 chrType 之外，另外加上一個 sequence 的 Type，所以共有 4 個 element！

△ <xsd:simpleContent> & <xsd:extension base="…"> 以及 <xsd:complexContent> &<xsd:extension base="….">
都必須放在 <xsd:complexType name="…"> 之下。(可看成是一個 content model)

△ 除了在 <xsd:extension ...> 中使用增加 <xsd:attribute...> 不能另外如 content model 一般的額外增加 <xsd:attribute...>，<xsd:extension ...> 不是 content model ，所以所有的 attribute 或是 element 都必放在其內部。

```
<xsd:complexType name="authorType">
    <xsd:simpleContent>
        <xsd:extension base="myNS:authorSelect">
            <xsd:attribute name="age" type="myNS:ageType"/>
        </xsd:extension>
    </xsd:simpleContent>
    <xsd:attribute name="CHNage" type="xsd:integer" />
</xsd:complexType>
```

以下的原 <xsd:complexType> 的宣告是允許的！！

```
<xsd:complexType name="authorType">
    <xsd:choice>
        <xsd:element name="mytest" />
    </xsd:choice>
    <xsd:attribute name="CHNage" type="xsd:integer" />
</xsd:complexType>
```

△ <xsd:attribute> 不可以放到 <xsd:choice> 之中，因為 <xsd:choice> 是一個 content model – 只能放 element 的 node。

△ 使用 anyType

In Schema file:

```
<xsd:complexType name="priceType">
    <xsd:all>
        <xsd:element name="fullPrice" type="xsd:positiveInteger" />
        <xsd:element name="disPrice" type="xsd:positiveInteger" />
    </xsd:all>
</xsd:complexType>

<xsd:complexType name="priceType2">
  <xsd:complexContent>
    <xsd:extension base="xsd:anyType" >
      <xsd:all>
        <xsd:element name="costPrice" type="xsd:positiveInteger" />
      </xsd:all>
      <xsd:attribute name="imp" use="required" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
……
<xsd:element name="price" type="myNS:priceType2"/>
……
```

因為設了 xsd:anyType
所以下面設的 content
model 都白廢了！

<xsd:extension base="xsd:anyType" > indicate essentially that there is no type upon with this complex type is based.

in XML Instace:

```
<author age="40">藍國桐</author>
<title>通訊原理與應用</title>
<page>288</page>
<price imp="High">
    (沒有 element)
</price>
```

(Demo： myXml_0072.xml 及 mySchema_0072.xsd)

Lecture 5: Using xsd:any & xsd:anyAttribute

By default complex types have closed content models. This means that only the specified particles are allowed to appear in an instance. XML Schema makes it possible, however, to define an open content model using what are known as wildcards. Using **xsd:any** within a complex type means that any element can appear at that location, effectively making it a placeholder for things that you cannot predict ahead of time. You can also use **xsd:anyAttribute** to define placeholders for attributes.

In XSD:

```
…..
<xsd:complexType name=”AuthorType” >
    <!-- compositor goes here -->
    <xsd:sequence>
        <xsd:element name=”name” type=”xsd:string” />
        <xsd:element name=”phone” type=”tns:Phone” />
        <xsd:any minOccurs=”0” maxOccurs=”unbounded” />
    </xsd:sequence>
    <xsd:anyAttribute />
</xsd:complexType>
<xsd:element name=”author” type=”tns:AuthorType” />
…….
```

In XML Instance:

```
<x:author xmlns:x=”http://www.ntou.edu.tw/XML”
xmlns:aw=”http://www.tnit.edu.tw/XML_1”    aw:auId=”02-2434532”>
    <!-- explicitly defined by the complexType -->
    <name>Aron</name>
    <phone>802-234-5498</phone>

    <!-- extra elements that replace wildcard -->
    <aw2:contract xmlns:aw2=”http://www.tnit.edu.tw/XML_2” >
        sential Web Service Wuick Reference</title>
        >2003-06-01</deadline>
    >
</x:author>
```

補充 <any> 的範例

(12/1)

結論：(檢視一個複雜的 schema 檔)

Let's inspect the type defining in a Namespace. Learining from the previous lecture, the **xsd:schema** element scopes what's in the namespace and the **targetNamespace** attribute specifies the namespace's name. For example, the following XML Schema template defines a new namespace called **http://www.ntou.edu.tw/XML** :

```
<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">

<!-- type defineitions- simpleType   -->
   <xsd:simpleType name="priceType">
      <xsd:restriction base="xsd:decimal">
         <xsd:minExclusive value="0"/>
         <xsd:maxExclusive value="500"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:simpleType name="authorSelect">
   ……. user defined Type ……
  </xsd:simpleType>


<!-- type defineitions- complexType   -->
   <xsd:complexType name="authorType">
      <xsd:simpleContent>
         <xsd:extension base="myNS:authorSelect">
            <xsd:attribute name="age" type="myNS:priceType"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>


<!-- type defineitions- global attribute declarations   -->
   <xsd:attribute name="ageType" type="myNS:priceType"/>
```

呼叫到一個自定的 Type
叫作 authorSelect

```
<!-- type defineitions- global element declarations  -->
   <xsd:element name="book">
      <xsd:complexType>
         <xsd:all>
            <xsd:element name="author" type="myNS:authorType"/>
            <xsd:element name="title" type="xsd:string"/>
            <xsd:element name="page" type="xsd:positiveInteger"/>
            <xsd:element name="price" type="myNS:priceType"/>
         </xsd:all>
      </xsd:complexType>
   </xsd:element>
```

(Demo： myXml_0060.xml 及 mySchema_0060.xsd)

說明一、Everything placed within the *xsd:schema* element (as an immediate child) is considered global and therefore automatically associated with the target namespace. In this example, there are four things in the http://www.ntou.edu.tw/XML namespace including priceType, authorSelect, authorType, ageType(attribute) and book(element). As a result, whenever you refer to one of these things within your schema you must use a namespace-qualified name.

說明二、To use a namespace-qualified name you'll need another namespace declaration that maps to the schema's targetNamespace value. The 'myNS' namespace declaration shown above serves this purpose. Hence, whenever I need to reference something I've defined in my schema, I can prefix the name with 'myNS' as illustrated in the example.

說明三、There are two classes o types that you can define within the xsd:schema element; simple types and complex types. Simple types can only be assigned to text-only elements and attributes since they don't define structure, but rather, value spaces. An element with additional structure, such as one that caries attributes or has child elements, must be defined as a complex type.

說明四、It's also possible to explicitly assign a type to an element in an instance document using the type attribute from http://www.w3.org/2001/XMLSchema-instance namespace. (允許在 instance document 的文件(即 XML 文件)中宣告 tag 的 type) You can

assign simple types to either attributes or text-only elements to constrain their values respectively. Also, it's important to note that the *xsi:type* technique for assigning type only applies to elements and not attributes.