XML/DTD理論實務與應用(3)－XML Schema介紹

華藝數位 陳嵩榮

XML Schema的功能

- 和DTD的角色相同，都是用來限定XML文件的結構：
  - 包含哪些元素(elements)、屬性(attributes)
  - 元素的順序與重複性
  - 限定每個元素與屬性的資料型態與值域(例如0~99的整數)

為什麼需要XML Schemas？

- 因為XML DTDs 有以下的限制：
  - DTD的語法和XML語法不相容
  - 支援的資料型態的能力較弱
    - 例如無法限定<年齡>元素的值是0到999的整數。
    - DTD只支援10種資料型態，XML Schemas supports 41種以上的資料型態

XML Schemas 的特點(1)

- 使用的語法與XML相同
- XML Schemas 支援40多種資料型態，遠比DTD只支援10種資料型態多
- 可以自行定義資料型態，例如<手機號碼>元素為字串型態，內容必須符合dddd-ddd-ddd 格式，d代表數字。
- 物件導向設計：可以延伸或限制舊的資料型態，衍生出新的資料型態
- 可以表達集合，定義子元素以任何順序出現。
- 可以限定元素內容必須具唯一性，或在某個值域內具唯一性

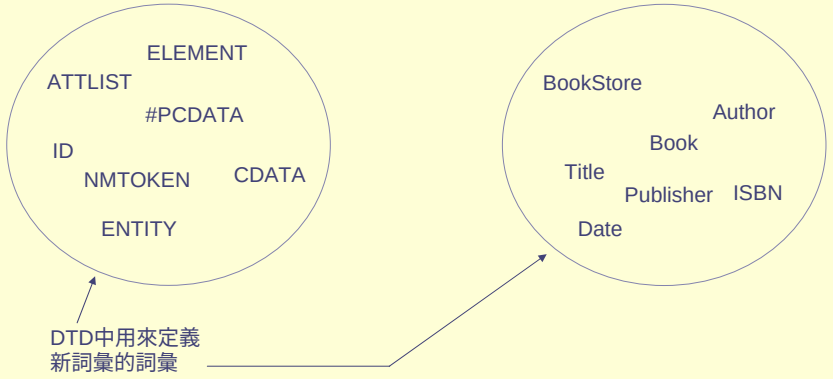
XML Schemas 的特點(2)

- 可定義多個 elements 有相同的名稱，但有不同的內容
- 可定義 elements 的內容為空值 (與Empty Element 意義不同)
- 可定義可取代的 elements，例如 “subway” element 統一被 “train” element 所取代

以一個書店的DTD為例

```
<!ELEMENT BookStore (Book)+>
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

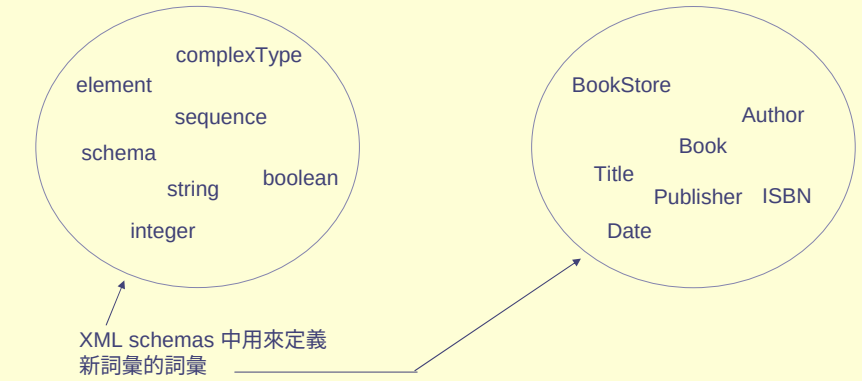
DTD中用來定義新詞彙的詞彙



XML Schema中用來定義新詞彙的詞彙

http://www.w3.org/2001/XMLSchema

http://www.books.org (targetNamespace)



XML Schema 與 DTDs 的一個不同點是 XML Schema 的詞彙可以與 namespace 結合
同樣的，由XML Schema 定義的新詞彙也可與 namespace 結合。DTDs 則不行

書店的 XML Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

所有的 XML Schemas 均以 “schema” 作為 root element

書店DTD與 XML Schema的對照

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
            targetNamespace="http://www.publishing.org"
            xmlns="http://www.publishing.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

<!ELEMENT BookStore (Book)+>

```
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>

```
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

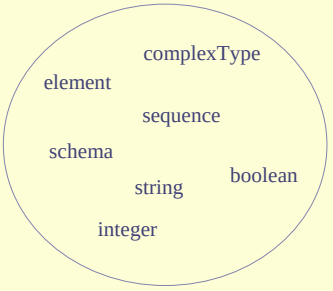
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
            targetNamespace="http://www.publishing.org"
            xmlns="http://www.publishing.org"
            elementFormDefault="qualified">
  <xsd:element name="BookCatalogue">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

用來描述這個 XML Schema 從那一個 XML Schema Namespace 衍生而來

XML Schema Namespace

http://www.w3.org/2000/10/XMLSchema

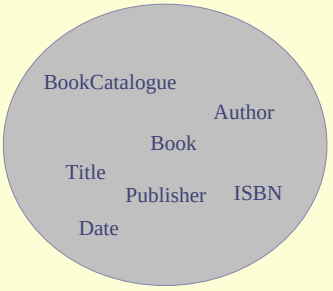


```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
            targetNamespace="http://www.publishing.org"
            xmlns="http://www.publishing.org"
            elementFormDefault="qualified">
  <xsd:element name="BookCatalogue">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

說明這個 shecma 所宣告的elements (BookCatalogue, Book, Title, Author, Date, ISBN, Publisher) 被定義在這個 namespace

Publishing Namespace (targetNamespace)

http://www.publishing.org (targetNamespace)



```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
            targetNamespace="http://www.publishing.org"
            xmlns="http://www.publishing.org"
            elementFormDefault="qualified">
  <xsd:element name="BookCatalogue">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

預設的 namespace 是 http://www.publishing.org 亦即是 targetNamespace

參引的 Book element 宣告並沒有指定它來自哪一個 namespace。表示採用預設的 namespace, 亦即 targetNamespace!

Element 宣告的兩種方法

1

```
<xsd:element name="name" type="type" minOccurs="int" maxOccurs="int"/>
```

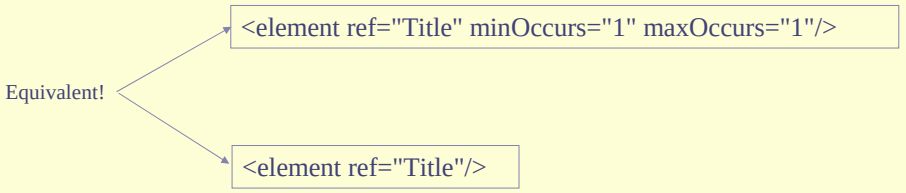
- type: simple type (e.g., xsd:string) or complexType 的名稱
- minOccurs: 非負整數
- maxOccurs: 非負整數or "unbounded"

2

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
    <xsd:complexType>
        …
    </xsd:complexType>
</xsd:element>
```

minOccurs 與 maxOccurs 的預設值

- minOccurs 的預設值是 "1"
- maxOccurs 的預設值是 "1"



XML Schema的元素型態

- 簡單型態(Simple Type)
  - 內容只包含值，不包含子元素或屬性的元素，例如：

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

- 複雜型態(Complex Type)
  - 內容包含子元素或屬性的元素，例如：

```
<xsd:element name="Book">
   <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
      <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

故事書的XML文件實例

```
<?xml version="1.0" encoding="big5"?>
<book isbn="9573224984">
    <title>三國英雄傳</title>
    <author>吉川英治</author>
    <character>
        <name>諸葛亮</name>
        <friend-of>龐統</friend-of>
        <since>公元181年</since>
        <alias>臥龍</alias>
    </character>
    <character>
        <name>龐統</name>
        <since>公元179年</since>
        <alias>鳳雛</alias>
    </character>
</book>
```


從故事書的DTD來看XML Schema的寫法

```
<!ELEMENT book (title, author, character*)>
<!ATTLIST book isbn CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT character (name, friend-of*, since, alias)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT friend-of (#PCDATA)>
<!ELEMENT since (#PCDATA)>
<!ELEMENT alias (#PCDATA)>
```

XML Schema的設計方式(1)

- Russian Doll設計方式
  - 依照文件類型的語義結構，由最外層的根元素向內逐層宣告，像俄羅斯玩偶一般，一層包著一層

Russian Doll 設計方式實例

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="book">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="title" type="xs:string"/>
                <xs:element name="author" type="xs:string"/>
                <xs:element name="character" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="name" type="xs:string"/>
                            <xs:element name="friend-of" type="xs:string" minOccurs="0"
                                        maxOccurs="unbounded"/>
                            <xs:element name="since" type="xs:date"/>
                            <xs:element name="alias" type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="isbn" type="xs:string"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema的設計方式(2)

- Flat Catalog設計方式
  - 先宣告所有簡單型態元素與屬性
  - 往上逐層宣告複雜型態元素，使用ref 屬性來參引下層的子元素或屬性
  - 宣告根元素。

Flat Catalog
設計方式實例

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="title" type="xs:string"/>
    <xs:element name="author" type="xs:string"/>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="friend-of" type="xs:string"/>
    <xs:element name="since" type="xs:date"/>
    <xs:element name="alias" type="xs:string"/>
    <xs:attribute name="isbn" type="xs:string"/>

    <xs:element name="character">
       <xs:complexType>
          <xs:sequence>
                <xs:element ref="name"/>
                 <xs:element ref="friend-of" minOccurs="0" maxOccurs="unbounded"/>
                 <xs:element ref="since"/>
                  <xs:element ref="alias"/>
          </xs:sequence>
       </xs:complexType>
    </xs:element>

    <xs:element name="book">
       <xs:complexType>
          <xs:sequence>
                <xs:element ref="title"/>
                <xs:element ref="author"/>
                <xs:element ref="character" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute ref="isbn"/>
       </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema的設計方式(3)

- 自訂資料型態的設計方式
  - 先宣告所有自訂的簡單型態，如：

```
<xs:simpleType name="nameType">
   <xs:restriction base="xs:string">
      <xs:maxLength value="32"/>
   </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="isbnType">
   <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{10}"/>
   </xs:restriction>
</xs:simpleType>
```

  - 往上逐層宣告自訂的複雜型態，使用type 屬性來參引下層的簡單型態
  - 宣告根元素，使用type 屬性來參引下層的複雜型態或簡單型態

自定資料型態的設計方式實例

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:simpleType name="nameType">
      <xs:restriction base="xs:string"> <xs:maxLength value="32"/>  </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="sinceType">  <xs:restriction base="xs:date"/>   </xs:simpleType>
   <xs:simpleType name="descType">   <xs:restriction base="xs:string"/>  </xs:simpleType>
   <xs:simpleType name="isbnType">
      <xs:restriction base="xs:string">
         <xs:pattern value="[0-9]{10}"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="characterType">
      <xs:sequence>
         <xs:element name="name" type="nameType"/>
         <xs:element name="friend-of" type="nameType" minOccurs="0"  maxOccurs="unbounded"/>
         <xs:element name="since" type="sinceType"/>
         <xs:element name="alias" type="descType"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="bookType">
      <xs:sequence>
         <xs:element name="title" type="nameType"/>
         <xs:element name="author" type="nameType"/>
         <xs:element name="character" type="characterType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="isbn" type="isbnType" use="required"/>
   </xs:complexType>
   <xs:element name="book" type="bookType"/>
</xs:schema>
```

群組與組合

- 群組(group)
  - 可以將幾個元素與屬性定義為一個群組，在隨後的複雜型態定義中引用，以使XML Schema更簡潔。
- 組合(compositor)
  - sequence : 按順序出現，類似DTD內容模型的 ”,”
  - choice : 任選一者，類似DTD內容模型的 ”|”
  - all : 可依任何順序出現

群組的
定義與使用

```
<xs:group name="mainBookElements">
   <xs:sequence>
      <xs:element name="title" type="nameType"/>
      <xs:element name="author" type="nameType"/>
   </xs:sequence>
 </xs:group>

<xs:attributeGroup name="bookAttributes">
   <xs:attribute name="isbn" type="isbnType" use="required"/>
   <xs:attribute name="available" type="xs:string"/>
</xs:attributeGroup>

<xs:complexType name="bookType">
  <xs:sequence>
    <xs:group ref="mainBookElements"/>
    <xs:element name="character" type="characterType"
        minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attributeGroup ref="bookAttributes"/>
 </xs:complexType>
```

組合的定義與使用

```
<xs:group name="nameTypes">
  <xs:choice>
    <xs:element name="name" type="xs:string"/>
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="middleName" type="xs:string" minOccurs="0"/>
      <xs:element name="lastName" type="xs:string"/>
    </xs:sequence>
  </xs:choice>
</xs:group>
```

```
<xs:complexType name="bookType">
  <xs:all>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="author" type="xs:string"/>
    <xs:element name="character" type="characterType" minOccurs="0"
                maxOccurs="unbounded"/>
  </xs:all>
  <xs:attribute name="isbn" type="isbnType" use="required"/>
</xs:complexType>
```


資料型態的衍生

- restriction
  <xs:restriction base="xs:string">
- pattern
  <xs:pattern value="[0-9]{10}"/>
- maxLength
  <xs:maxLength value="32"/>
- union : 由兩個以上的資料型態聯集而成
- list : 數個符合指定型態的值以空白隔開

union的定義與使用

```
<xs:simpleType name="isbnType">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="[0-9]{10}"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="TBD"/>
        <xs:enumeration value="NA"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```


list的定義與使用

```
<xs:simpleType name="isbnTypes">
  <xs:list itemType="isbnType"/>
</xs:simpleType>
<xs:simpleType name="isbnTypes10">
  <xs:restriction base="isbnTypes">
    <xs:minLength value="1"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
```

內容型態

- 空元素(empty element)：沒有純文字內容與子元素，可以有屬性。
- 純文字內容(simple content)：沒有子元素，可以有屬性。(簡單型態元素則是沒有子元素與屬性)
- 混合型內容(mixed content)：內容包含純文字與子元素。

空元素的定義與使用

```
<xs:element name="book">
  <xs:complexType>
    <xs:attribute name="isbn" type="isbnType"/>
  </xs:complexType>
 </xs:element>
```

純文字內容的定義與使用

```
<xs:element name="book">
   <xs:complexType>
     <xs:simpleContent>
       <xs:extension base="xs:string">
         <xs:attribute name="isbn" type="isbnType"/>
       </xs:extension>
     </xs:simpleContent>
  </xs:complexType>
</xs:element>

<book isbn="0836217462">
          Funny book by Charles M. Schulz.
          Its title (Being a Dog Is a Full-Time Job) says it all !
</book>
```

混合型內容的定義與使用

```
<xs:element name="book">
   <xs:complexType mixed="true">
     <xs:all>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="author" type="xs:string"/>
     </xs:all>
     <xs:attribute name="isbn" type="xs:string"/>
   </xs:complexType>
</xs:element>

<book isbn="0836217462">
          Funny book by <author>Charles M. Schulz</author>.
          Its title (<title>Being a Dog Is a Full-Time Job</title>)
          says it all !
</book>
```


定義限制

- 唯一性(unique)
- 鍵值(key)
  - 內容不能是Null
  - 鍵值不一定要具唯一性
- 鍵值參引(keyref)
  - 內容必須是已存在的鍵值

唯一性
的定義與使用

```
<xs:unique name="charName">
    <xs:selector xpath="character"/>
    <xs:field xpath="name"/>
</xs:unique>
```

鍵值和鍵值參引
的定義與使用

```
<xs:key name="charName">
  <xs:selector xpath="character"/>
  <xs:field xpath="name"/>
</xs:key>

<xs:keyref name="charNameRef" refer="charName">
  <xs:selector xpath="character"/>
  <xs:field xpath="friend-of"/>
</xs:keyref>
```


增加XML Schema的可讀性－加入註解

```
<xs:element name="book">
 <xs:annotation>
        <xs:documentation xml:lang="en">
                Top level element.
        </xs:documentation>
        <xs:documentation xml:lang="fr">
                Element racine.
        </xs:documentation>
        <xs:appinfo source="http://example.com/foo/">
                <bind xmlns="http://example.com/bar/">
                        <class name="Book"/>
                </bind>
        </xs:appinfo>
 </xs:annotation>
```

在 XML 文件中參引 schema

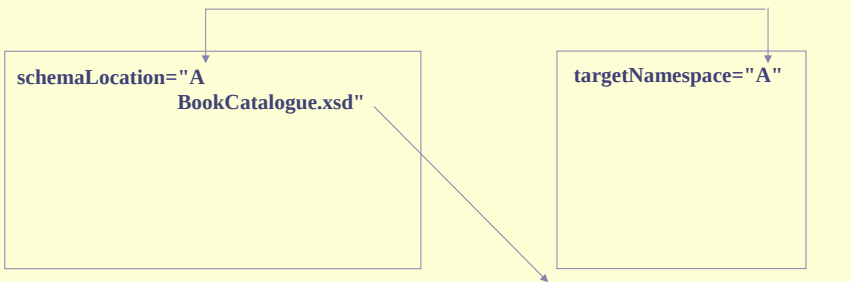
```
<?xml version="1.0"?>
<BookCatalogue xmlns ="http://www.publishing.org"
               xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
               xsi:schemaLocation="http://www.publishing.org
                                   BookCatalogue.xsd">
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>July, 1998</Date>
        <ISBN>94303-12021-43892</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    ...
</BookCatalogue>
```

1. 使用預設的 namespace 宣告，告訴 schema-validator 這份文件中所出現的 elements 都來自於 publishing namespace.

2. schemaLocation 是告訴 schema-validator *"http://www.publishing.org"* 這個 namespace 被定義在 BookCatalogue.xsd.

3. 告訴 schema-validator 正在使用的 schemaLocation 屬性是定義在schema instance namespace.

在 XML 文件中參引 schema



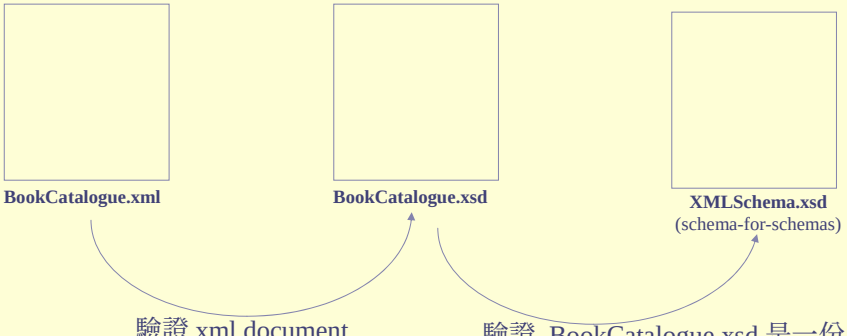
BookCatalogue.xml

- 使用來自 namespace A 的 element

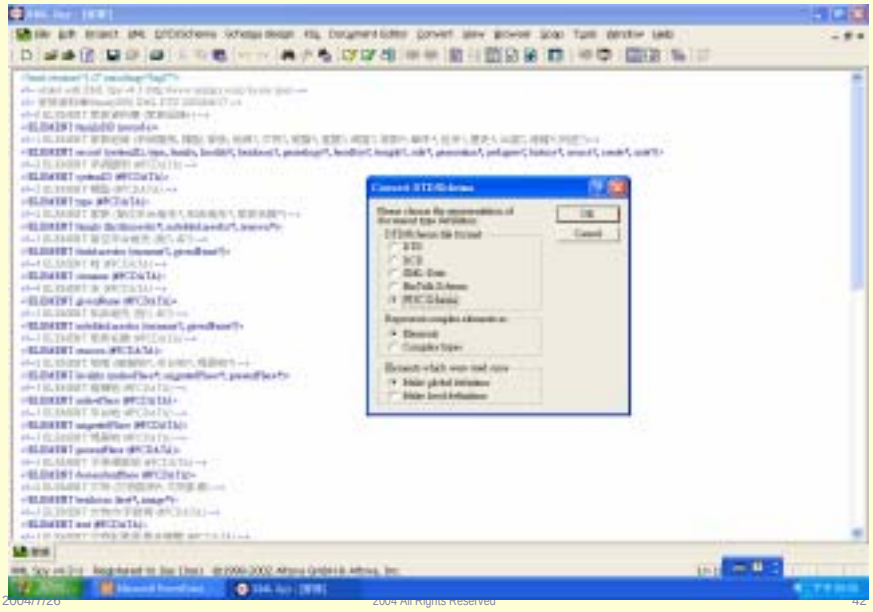
BookCatalogue.xsd

- 定義一個在 namespace A 的 namespace

多層次的驗證

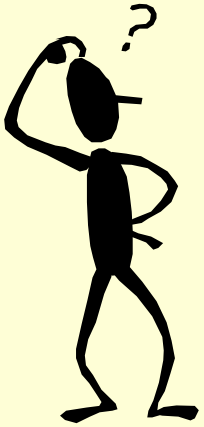


利用XML Spy將DTD轉成XML Schema



參考資料

- “XML Schema Tutorial”, by Roger L. Costello, September 2001 (http://www.xfront.com/)
- “Using W3C XML Schema”, by Eric van derVlist, October 2001, (http://www.xml.com/pub/a/2000/11/29/schemas/part1.html)



Questions?

任何問題，歡迎來信：
joe@airiti.com