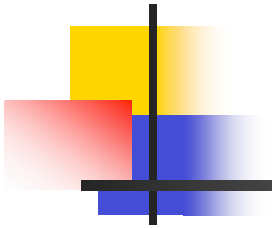
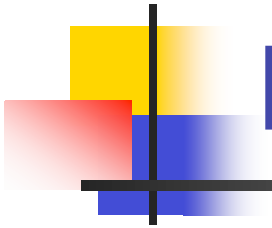


B336 Internet Systems Programming

XML Schema

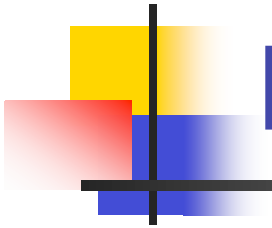
(Week 6 Lecture 1)



Learning Objectives

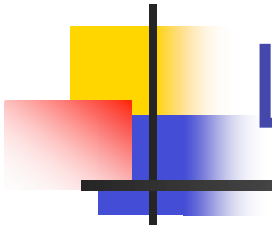
- Understand what an XML Schema is, and how it compares to a DTD.

- Know the format and syntax of XML Schemas as specified by W3C's XML Schema Recommendations.



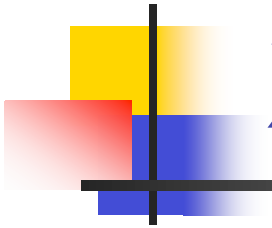
Learning Objectives

- In the scheme of what we are doing in this unit:
  - We are studying how to use XML as an important set of Internet technologies to use as solutions in different areas.
  - The ability to create new mark-up languages is a key factor in XML technologies.
  - Along with DTD, XML Schema is one of the main methods used to define the syntax of XML documents It is therefore a critical technology to allow us to create new mark-up languages.



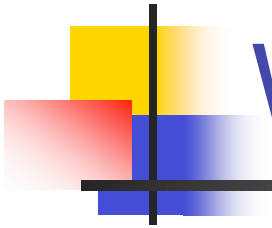
Lecture Outline

- Why XML Schema?
- Ways of defining a document structure in XML Schema.
- Example XML Schemas (compare with DTD from last lecture)



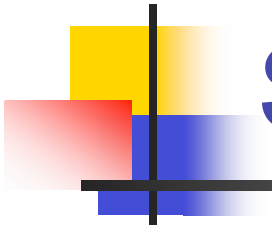
XML Schema

- In the last lecture, we discussed how W3C's XML 1.0 Recommendations specify the syntax of valid XML documents by using DTDs.
- XML Schema does the same job as DTD.
  - It specifies the structure of XML documents
  - But has different features from DTDs.
- http://www.w3.org/XML/Schema



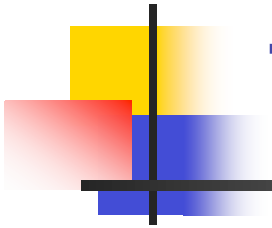
Why Have XML Schema?

- People are dissatisfied with DTDs:
  - It uses different syntax compared to XML documents
    - So can't be checked by standard XML parsers, not easy to transform using XSLT, etc.
  - Has very limited datatype capabilities
    - eg can't specify that a piece of data must be a number between 0 and 100.



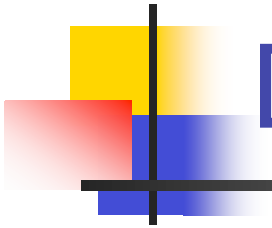
Some Features of XML Schema

- Some advancement over DTDs:
  - Enhanced datatypes
    - Over 44 built-in datatypes in Schemas versus 10 in DTDs
    - Can create your own datatypes
  - Written in the XML syntax
    - All the tools for XML (like style sheets, parsers and processors we will be discussing in future lectures) can be used on the Schema
  - More powerful support content models
    - We can specify more interesting contents for XML documents (ie. more interesting languages)
  - Extensible
    - Because it is in XML !
  - etc.



The XML Schema Specifications

- W3C's XML Schema specifications comes in 3 parts:
  - Part 0: Primer
    - just a tutorial - not really part of the specifications
  - Part 1: Structures
    - How to define the basic structure of XML documents using XML Schema
    - "this element contains these elements, which contains these other elements, etc..."
  - Part 2: Datatypes
    - How to define datatypes of elements and attributes used in an XML document
    - "this element shall hold an integer with the range 0 to 10, etc..."



Declaring Elements

- Elements are the basic components of XML documents (refer to lecture on XML Documents again)
- Just as in DTDs, the primary purpose of XML Schema is to declare elements.
- To declare elements in a Schema, XML Schema has an element called “element”!

Ways of Declaring Elements

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

- minOccurs and maxOccurs specifies how many times this element can exists.
- Eg.

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

```
<xsd:element name="Result" type="xsd:float" maxOccurs="unbounded"/>
```

```
<xsd:element name= "DateOfBirth" type="xsd:date"
                                        minOccurs="1" maxOccurs="1"/>
```

Ways of Declaring Elements (containing sub-elements)

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

Eg.

```
<xsd:element name="Staff" minOccurs="1" maxOccurs="50">
  <xsd:complexType>
    <xsd:sequence>
       <xsd:element name="Surname" type="xsd:string" minOccurs="1"/>
       <xsd:element name="Age" type="xsd:positiveInterger"/>
       <xsd:element name="Address" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Eg. XML document content based on the previous Schema:

```
<Staff>
  <Surname>Smith</Surname>
</Staff>


<Staff>
  <Surname>Jones</Surname>
  <Age>23</Age>
</Staff>


<Staff>
  <Surname>Clause</Surname>
  <Age>2003</Age>
  <Address>123, Ice St, North Pole.</Address>
</Staff>
```

Ways of Declaring Elements (extending a Simple type to defining new types)

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
      <xsd:simpleType>
            <xsd:restriction base="type">
            …
            </xsd:restriction>
      </xsd:simpleType>
</xsd:element>
```

Eg.

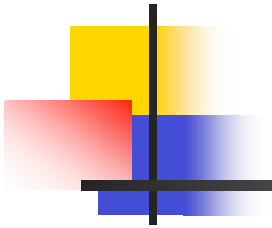
```
<xsd:element name="Age" maxOccurs="1">
  <xsd:simpleType>
    <xsd:restriction base="xsd:positiveInteger">
       <xsd:minInclusive value="0"/>
       <xsd:maxInclusive value="150"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Eg.

```
<xsd:simpleType name="ISBNType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
        <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
        <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
    </xsd:restriction>
</xsd:simpleType>
```

- This means: a new type called “ISBNType”, extended from the “string”.
- Has 3 possible patterns:
  - First Pattern: 1 digit followed by a dash followed by 5 digits followed by another dash followed by 3 digits followed by another dash followed by 1 more digit, or
  - Second Pattern: 1 digit followed by a dash followed by 3 digits followed by another dash followed by 5 digits followed by another dash followed by 1 more digit, or
  - Third Pattern: 1 digit followed by a dash followed by 2 digits followed by another dash followed by 6 digits followed by another dash followed by 1 more digit.”

- Example borrowed from http://www.xfront.com/xml-schema.html



Facets

- “minInclusive”, “maxInclusive” and “pattern” in the previous examples are called ***facets***.

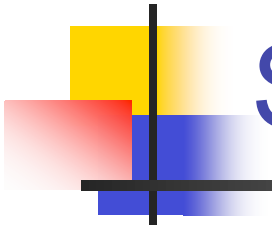
- We extend a base type by changing it’s facet values.

Ways of Declaring Elements (by Reference to another element)

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

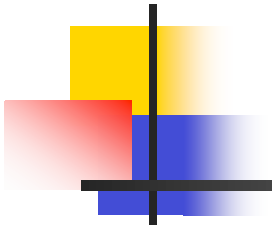
Eg.

```
<xsd:element name="Staff" minOccurs="1" maxOccurs="50">
  <xsd:complexType>
    <xsd:sequence>
       <xsd:element ref="Surname" minOccurs="1"/>
       ...
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

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



Some Useful Built-in Datatypes

- Primitive types:
  - string, boolean, number, float, double, date, time, gYear, gMonth, gDay, hexBinary, AnyURI, etc...
- Derived types
  - normalizedString, integer, short, long, negativeInteger, positiveInteger, etc...



Declaring Attributes

- We can use the “attribute” element in Schemas to add attributes to elements in our new language.

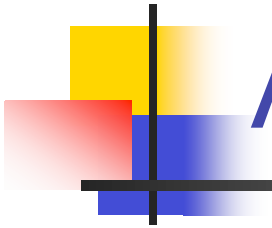
- Eg.

```
<xsd:element name="Student">
  <xsd:complexType>
    <xsd:sequence>
       <attribute name="Workrate" use="optional"/>
        <xsd:element name="Surname" type="xsd:string"/>
        ...
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<attribute name="Workrate" use="optional"/>

...

<xsd:element name="Student">
  <xsd:complexType>
    <xsd:sequence>
       <attribute ref="Workrate"/>
        <xsd:element name="Surname" type="xsd:string"/>
        ...
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Eg. XML document content from the previous Schema:

```
<Student workrate="topnotch">
  <Surname>Smith</Surname>
</Student>


<Student workrate="average">
  <Surname>Jones</Surname>
</Student>


<Student>
  <Surname>Dropout</Surname>
</Student>
```



Annotations

- Since XML Schema is suppose to be self-describing, there is an “annotation” element which we can put annotations about the Schema into.

```
...
<xsd:annotation>
    <xsd:documentation>
          This schema is to specify information about the students
          within an academic instituion. This Schema was developed
          by ...
    </xsd:documentation>
<xsd:/annotation>

...
```

```
...
<xsd:annotation>
    <xsd:documentation>
          The following "StudentID" element is extracted from ...
    </xsd:documentation>
<xsd:/annotation>

<xsd:element name="StudentID">
    ...
</xsd:element>
```

Example Schema

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://course.murdoch.edu.au"
            xmlns:p="http://course.murdoch.edu.au">

    <xsd:element name="course">
        <xsd:complexType>
            <xsd:sequence>
              <xsd:element ref="name" />
              <xsd:element ref="unit" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="unit">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="title" />
                <xsd:element ref="lecturer" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element ref="tutor" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
```

Declaring the "course" root element

Declaring the "unit" element

Declaring the "lecturer" element

```
    <xsd:element name="lecturer">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="surname" />
                <xsd:element ref="othernames" minOccurs="0" maxOccurs="1"/>
                <xsd:element ref="email" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    ...

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

</xsd:schema>
```

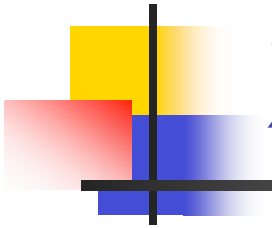
Declaring the elements as simple string types

Example XML Document

Root element Refer to the XML Schema

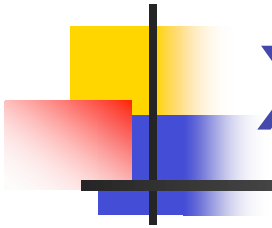
```
<?xml version="1.0" encoding="UTF-8"?>
<course xmlns ="http://course.murdoch.edu.au"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://course.murdoch.edu.au course.xsd">
    <name>Bachelor of Science - Internet Computing</name>
    <duration>3 years</duration>
    <unit>
        <title>B336 Internet Systems Programming</title>
        <lecturer>
            <surname language="English">Hiew</surname>
            <othernames language="English">Hong Liang</othernames>
            <email>h.hiew@murdoch.edu.au</email>
        </lecturer>
    </unit>
    <unit>
        <title>B108 Introduction to Multimedia and the Internet</title>
        <lecturer>
            <surname>Rai</surname>
            <othernames>Shri</othernames>
            <email>s.rai@murdoch.edu.au</email>
        </lecturer>
    </unit>
</course>
```

The same as the version with DTD in the last lecture



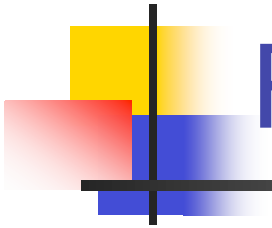
XML Schema Validating Parsers

- Like DTDs, there are some parsers that validates XML documents based on their XML Schemas.
- See examples:
  - http://www.w3.org/XML/Schema#Tools
- We will be using tools containing such parsers in the labs exercises.



XML Schema lab work…

- In this unit, you will not be dwelving very much into XML schema in your practical lab work due to the lack to time.
  - Most current tools available are for DTDs.
  - No doubt more software (like validators) will emerge for XML Schemas.
- Nevertheless, XML Schema and other emerging methods for defining document types are very important developments, and are things you will have to get familiar with if you do future work in XML.



References

- Required reading:
  - Sybex textbook chapters 7 & 8.
- Online tutorials:
  - http://www.w3.org/TR/xmlschema-0/
  - http://www.xfront.com/xml-schema.html - downloadable tutorial in a ZIP file.
- Some example XML Schemas:
  - http://www.w3.org/XML/Schema#Usage