XML DTDs and Namespaces

CS174

Chris Pollett

Oct 3, 2007.

Outline

- Internal versus External DTDs
- Namespaces
- XML Schemas

Internal versus External DTDs

- There are two ways to associate a DTD with an XML document:
- Internally --

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

<!DOCTYPE planes [<!--DTD for planes]>

<!-- The planes document -->

- Externally --

<!DOCTYPE planes_for_sale SYSTEM "planes.dtd" >

Namespaces

- It is sometimes useful to mix and match markup from several XML languages into one file.
- For instance, you might want to have mark-up for HTML as well as mark-up for SVG, a vector graphics language, and MathML, a language for mathematics.
- One could do this with a declaration like:

<div xmlns="http://www.w3.org/1999/xhtml"

xmlns:svg="http://www.w3.org/2000/svg"

xmlns:mml="http://www.w3.org/1998/Math/MathML" >

Now to use an html tag within this div I don't need a prefix.

To use the svg tag rect, I might write <svg:rect …>

- This solves the problem where one might have two languages both with the same name for a tag. For intancs, both with a <table> tag.

XML Schemas

- DTDs are defined in a language which is different from XML.
- We also cannot finely control what kinds of character data can appear within tags.
- The XML schema language was created to address both these issues.

More on Schemas

- Schemas are namespace centric.
- The namespace for XML Schema is

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

- xsd is the common namespace to use when dealing with XML schema tags.
- A XML schema is used to define a target namespace of the language we are defining.
- A defining schema can be used to allow or restrict other schemas from being used with the target namespace.
- At its simplest, a schema document consists of a bunch of tags such as <element> <simpleType>, <complexType>, <sequence>, <all>,<restriction>, and enclosed with a <schema> tag.

The Schema Tag

- A typical schema tag might look like:

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    <!-- the namespace for XML schemas. Having th xsd is optional,
       though, if we have it we must precede all tags with xsd: -->
    targetNameSpace="http://somewhere.com/planeSchema"
     <!-- the namespace for the elements we are defining with this document
                                                          -->
    xmlns="http://somewhere.com/planeSchema"
    <!-- Default namespace for this document -->
    elementFormDefault="qualified"
    <!-- allow non-top level elements to appear in the target namespace -->
>
```

Associating a XML schema with a n XML document

- An instance of a schema must specify the namespaces it uses.
- These include the default namespace, the standard namespace for instances, and the schema location:

  <plane xmlns="http://somewhere.com/planeSchema"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation ="http://somewhere.com/planeSchema/plane.xsd"
  >
- Programs such as XMLSpy, oXygen XML, xsv can be used to validate schemas, as well as validate documents against a schema.

Overview of data types

- There are two categories of data types in XML Schemas:
  - simple types -- which are restricted to strings and cannot have attributes or nested elements
  - complex types -- which can have attributes and can include other data types as elements.
- There are 44 different types in XML schemas, 19 of which are primitive and the remainder are derived.
- Example primitive types include: string, Boolean, time and anyURI.
- Example predefined derived types include: byte, long, decimal, unsignedInt, positiveInteger, and NMTOKEN.
- User defined types are defined by specifying restrictions on an existing type (called a base type).
- For example, for integer there are 8 so-called “facets” on which restrictions to base types can be made: totalDigits, maxInclusive, maxExclusive minInclusive, minExclusive, pattern, enumeration, and whitespace.
- Types can be named or anonymous. Anonymous elements cannot be used outside of the element in which it was declared
- Elements in DTDs are all global. For schemas one can have local elements. These are elements which defined within the scope of some child of the schema tag. child elements themselves are global.

Simple Types

- The simplest way to define a simple a new tag would be with a command like:

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

  <!-- notice namespace applied to the word string. In general namespaces can be applied to lots of things. For example, they also can be applied to attributes. -->

- In an instance of the plane schema we could then have:

  <engine>an example of the content of an engine</engine>

- You can also give default values or force fixed values with slight variations of this declaration:

  <xsd:element name="engine" type="xsd:string" default="V-6" />

  <xsd:element name="plane" type="xsd:string" fixed="single wing" />

- A simple user-derived type can be defined using the <restriction> tag:

  <xsd:simpleType name="firstName">
  <xsd:restriction base="xsd:string">
  <xsd:maxLength value="10" />
  </xsd:restriction>
  </ xsd:simpleType>

Complex Types

- There are several kinds of complex types that can be used with XML schemas.
- We will only look at the complex types which are restricted to having subelement-only -- not both subelements and text. the complexContent tag can be used to handle other kinds of content.
- To define what subelement occur for an element we can use either sequence or all. sequence -- forces an order on the subelement, all doesn't.

```
<xsd:complexType name="car">
    <xsd:sequence>
      <xsd:element name="make"  type="xsd:string" />
      <xsd:element name="year" type="xsd:decimal" />
    </xsd:sequence><!-- you can use minOccurs, maxOccurs to specify number of occurrences ; you
     can have more than one sequence, all tag here-->
</ xsd:complexType>
```

References

- You can define element to be used as a subelement outside of another element and use a reference to refer to it:

```
<xsd:element name="year">
<xsd:simpleType >
    <xsd:restriction base="xsd:decimal">
      <xsd:minInclusive value="1900" />
      <xsd:maxInclusive value="2010" />
    </xsd:restriction>
</ xsd:simpleType>
</xsd:element>
<xsd:complexType name="car">
        <xsd:sequence>
         <xsd:element name="make"  type="xsd:string" />
          <xsd:element ref="year" />
       </xsd:sequence>
</ xsd:complexType>
<xsd:element name="sport_car" type="car">
                              <xsd:attribute name="color" type="xsd:string" />
</xsd:element>
```