XML Schemas, Beginning Ajax

CS174

Chris Pollett

Oct 16, 2006.

Outline

- XML Schemas
- XML and CSS
- XSLT
- Beginning AJAX

XML Schemas

- Last day, we said there were two approaches to defining XML based languages: DTDs or schemas.
- Schemas are XML documents used to specify an XML language.
- A schema document consists of a bunch of tags such as <element> <simpleType>, <complexType>, <sequence>, <all>,<restriction>, and enclosed with a <schema> tag.
- As we said last day, XML Schemas are namespace oriented as we will see from the syntax for the attributes of the 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>
```

XML and CSS

- Most modern browsers are completely happy to style any tag provided there is some style-sheet information given for it:

plane {display:block; border 3px;}

- To associate a stylesheet with an XML document we use the syntax:

<?xml-stylesheet type=“text/css” href=“mystyles.css” ?>

XSLT

- Sometimes it is useful to transform one XML markup language into some other XML language.
- For instance, suppose you want to display a pure RSS feed nicely and you want the links to work in IE. Since IE does not support the XLink language this is hard unless you do a stylesheet tranformation.
- The basic idea of XSTL (eXtensible stylesheet transformations), is that we can associate a stylesheeet transformation with an XML document and apply this transformation using a processor (for instance, a browser), to get some other XML language.

An Example

```
<?xml version="1.0"
    encoding="utf-8" ?>
<?xml-stylesheet type="text/xslt"
    href="xslplane.xsl" ?>
<plane>
   <year>1970</year>
</plane>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0"
   xmlns:xsl=
     "http://www.w3.org/1999/XSL/Transf
     orm"
  xmlns =
     "http://www.w3.org/1999/xhtml"
>
<xsl:template match="plane">
<html><head><title>result of
     applying a stylesheet to
     plane</title></head><body><h1>
     Plane Description</h1>
    <xsl:apply-templates />
    </body></html>
</xsl:template>
<xsl:template match="year">
    <p style="color:red">
    <xsl:value-of select=".">
    </p>
</xsl:template>
```

Beginning AJAX

- AJAX stands for Asynchronous Javascript and XML.
- Asynchronous means that one does not need to wait for the response to an HTTP request to do something
  - one can use Javascript to have several outstanding HTTP requests and still make things happen in the browser.
  - Further one doesn't have to reload the entire page when the results of a request are returns.
- The XML in AJAX is because the results of a request are usual XML, JSON, or YAML data which is then formatted by some Javascript code.
- The basic key tp AJAX is the ability for Javascript to make an HTTP request. This is done with the XMLHttpRequest object. As an example, look at the Javascript in the file:

http://www.cs.sjsu.edu/faculty/pollett/test/dept3.html

Proxies

- Javascript function is only allowed to make requests back to the server from which it came.
- So if you have a page http://somewhere.com/index.html
  - and you would like the Javascript on it to make use of the Yahoo API, how do you do it?
- You need to use a proxy on your server which passes the request onto Yahoo!
- One example of a PHP script to do such proxy-ing can be found at:

http://developer.yahoo.com/javascript/samples/proxy/php_proxy_simple.txt