The XML Namespace

It is Implicitly Declared
It Contains 4 Great Attributes

Table of Contents

The XML Namespace is Implicitly Declared

Consider this XML document:

```
<?xml version="1.0"?>
<N1:NumberList xmlns:N1="http://www.example1.org"
               xmlns:N2=" http://www.example2.org ">
        <Number>23</Number>
        <Number>41</Number>
        <Number>70</Number>
        <Number>103</Number>
        <Number>99</Number>
        <Number>6</Number>
</N1:NumberList>
```

For this element:

```
        <Number>23</Number>
```

what namespaces are in scope? Clearly these two namespaces are in-scope:

http://www.example1.org

http://www.example2.org

Is that all?

Here is an XSLT template that reveals the in-scope namespaces:

```
<xsl:template match="//Number[1]">
       <xsl:for-each select="namespace::*">
           <xsl:value-of select="."/></p>
       </xsl:for-each>
</xsl:template>
```

Here is the output:

http://www.w3.org/XML/1998/namespace

http://www.example1.org

http://www.example2.org

There are three (3) in-scope namespaces!

Lesson Learned: Implicit on the root element of every XML document is a namespace declaration for the XML namespace:

```
   <root xmlns:xml="http://www.w3.org/XML/1998/namespace">
```

The XML namespace is built into every XML application.

Note: it is legal to explicitly declare the XML namespace, provided that it's bound to the 'xml' prefix with namespace name 'http://www.w3.org/XML/1998/namespace'.

The implicit namespace declaration gives you access to the attributes that are in that namespace, such as the xml:lang attribute:

```
<movie>
     <title>The Laughing Cow</title>
     <title xml:lang="fr">La Vache Qui Rit</title>
</movie>
```

This XSLT template outputs the in-scope namespaces and their prefixes:

```
<xsl:template match="//Number[1]">
    <xsl:for-each select="namespace::*">
        Namespace in scope = <xsl:value-of select="."/>
        Namespace prefix = <xsl:value-of select="name(.)"/>
    </xsl:for-each>
</xsl:template>
```

Here is the output:

```
Namespace in scope = http://www.w3.org/XML/1998/namespace
Namespace prefix   = xml

Namespace in scope = http://www.example1.org
Namespace prefix   = N1

Namespace in scope = http://www.example2.org
Namespace prefix   = N2
```

Using the 4 Great XML Namespace Attributes

The XML namespace has 4 great attributes: xml:lang, xml:id, xml:space, and xml:base. Before discussing them, let's review the great attributes in HTML (as an analogy) and see how they are used.

The Great HTML Attributes

HTML has some great attributes—id, class, and title—that can be dropped into just about every element. Here's an example:

```
<div id="Mercury">Mercury has a diameter
       of <span class="diameter">3,032.4 miles</span>.
       It's distance from earth is
       <span class="distance">57 million miles</span>,
       at the closest point in its orbit.
</div>
```

This is very nice.

The Great XML Namespace Attributes

The XML namespace contains 4 great attributes:

1. The **lang** attribute: use this on an element to state the language of its content.
2. The **id** attribute: use this on an element to uniquely identify it.
3. The **space** attribute: use this on an element to state that you want its whitespace nodes preserved during processing.

4. The **base** attribute: use this on an element to create a base URL; URL's within the element can be relative to the base URL.

Instance document authors should be able to drop these attributes into just about every element in their XML document.

Here are some examples to illustrate their use:

```
<movie>
     <title>The Laughing Cow</title>
     <title xml:lang="fr">La Vache Qui Rit</title>
</movie>
```

```
<Book xml:id="RB">
        <Title>Illusions</Title>
        <Author>Richard Bach</Author>
        <Date>1977</Date>
        <ISBN>0-440-34319-4</ISBN>
        <Publisher>Dell Publishing Co.</Publisher>
</Book>
```

```
<f xml:space="preserve">       </f>
```

```
<Books xml:base="http://www.xfront.com/schemas/">
        <Book xsi:schemaLocation="Book.xsd" …>
        ...
        <Book xsi:schemaLocation="Book.xsd" …>
        ...
        <Book xsi:schemaLocation="Book.xsd" …>
        ...
</Books>
```

This is very nice.

Status of the Great XML Namespace Attributes

The xml:lang and xml:space attributes are defined in the XML REC:

http://www.w3.org/TR/REC-xml/#sec-lang-tag

http://www.w3.org/TR/REC-xml/#sec-white-space

Consequently, they are supported by the XML parser and therefore are available for use with any XML application.

The xml:base and xml:id attributes are not defined in the XML REC. Consequently, they are not supported by the XML parser and therefore may not be available for use with every XML application.

The xml:base attribute is defined in the XML Base specification:

http://www.w3.org/TR/xmlbase/#syntax

Here are two XML applications that support xml:base

- SAXON
- XERCES

The xml:id attribute is defined in the xml:id specification:

http://www.w3.org/TR/xml-id/

Here is a list of XML applications that support xml:id

http://www.w3.org/XML/2005/01/xml-id-implementation.html

Enabling Use of the Great XML Namespace Attributes

Recall from above that the XML namespace (and the xml: prefix) is implicitly declared in every XML document. Thus, XML documents are primed for dropping the xml:lang attribute into any element, dropping the xml:id attribute into any element, and so forth.

Suppose your XML instance document conforms to an XML Schema. While it's true that the XML namespace is automatically available in every instance document, the attributes in it cannot be used unless the XML Schema has been explicitly designed to allow their use. Here's what you need to add to each element declaration:

```
<anyAttribute namespace="http://www.w3.org/XML/1998/namespace"/>
```

It enables any of the 4 great attributes from the XML namespace to be dropped into the element.

This <BookStore> element declaration states that any of the 4 great attributes from the XML namespace can be dropped into it:

```
<element name="BookStore">
   <complexType>
      <sequence>
          ...
      </sequence>
      <anyAttribute
            namespace="http://www.w3.org/XML/1998/namespace"/>
   </complexType>
</element>
```

You must do this for every element declaration. Yes, it is tedious. The good news is that in XML Schema 1.1 you will be able to declare "schema-wide attributes." Thus, in one fell swoop you will be able to state that the 4 great XML namespace attributes are available on every element.

Example

Create an XML Schema for BookStore.

The BookStore element contains Books and each of them contain Title, Author, Date, ISBN, and Publisher.

Design the XML Schema so that instance document authors can drop into any of the elements any of the 4 great XML namespace attributes. Here is a sample instance document:

```
<?xml version="1.0"?>
<BookStore xml:lang="en">

        <Book xml:id="PM" xml:space="preserve">
                <Title>My Life and Times</Title>
                <Author>Paul McCartney</Author>
                <Date>1998</Date>
                <ISBN>1-56592-235-2</ISBN>
                <Publisher>McMillin Publishing</Publisher>
        </Book>

        <Book xml:id="RB">
                <Title>Illusions</Title>
                <Author>Richard Bach</Author>
                <Date>1977</Date>
                <ISBN>0-440-34319-4</ISBN>
                <Publisher>Dell Publishing Co.</Publisher>
        </Book>

        <Book xml:id="JK">
```

```
                <Title>The First and Last Freedom</Title>
                <Author>J. Krishnamurti</Author>
                <Date>1954</Date>
                <ISBN>0-06-064831-7</ISBN>
                <Publisher>Harper &amp;amp; Row</Publisher>
        </Book>

</BookStore>
```

Note the use of xml:lang, xml:id, and xml:space.

Here is the XML Schema:

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.bookstore.org"
        xmlns:bk="http://www.bookstore.org"
        elementFormDefault="qualified">

    <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
                schemaLocation="http://www.w3.org/2001/xml.xsd"/>


    <element name="BookStore">
        <complexType>
            <sequence>
                <element name="Book" maxOccurs="unbounded">
                    <complexType>
                        <sequence>
                            <element name="Title"
                            type="bk:safe-string-plus-XML-attributes"/>

                            <element name="Author"
                            type="bk:safe-string-plus-XML-attributes"/>

                            <element name="Date"
                            type="bk:year-plus-XML-attributes"/>

                            <element name="ISBN"
                            type="bk:safe-string-plus-XML-attributes"/>

                            <element name="Publisher"
                            type="bk:safe-string-plus-XML-attributes"/>

                        </sequence>
                        <anyAttribute
                        namespace="http:// www.w3.org/XML/1998/namespace"/>
                    </complexType>
                </element>
            </sequence>
            <anyAttribute
            namespace="http://www.w3.org/XML/1998/namespace"/>
        </complexType>
    </element>
```

```
    <simpleType name="safe-string">
        <restriction base="string">
            <maxLength value="100" />
            <!-- ASCII characters -->
            <pattern value="[&#09;-&#127;]*" />
        </restriction>
    </simpleType>

    <complexType name="safe-string-plus-XML-attributes">
        <simpleContent>
            <extension base="bk:safe-string">
                <anyAttribute
                namespace="http://www.w3.org/XML/1998/namespace" />
            </extension>
        </simpleContent>
    </complexType>

    <complexType name="year-plus-XML-attributes">
        <simpleContent>
            <extension base="gYear">
                <anyAttribute
                namespace="http://www.w3.org/XML/1998/namespace" />
            </extension>
        </simpleContent>
    </complexType>

</schema>
```

Acknowledgements

Thank you to the following people for their contributions to this paper:

- David Carlisle
- Roger Costello
- Michael Glavassevich
- Michael Kay
- David Lee
- Evan Lenz
- Amelia Lewis
- Simon St. Laurent
- Hermann Stamm-Wilbrandt
- Andrew Welch