Unit 3

XML Namespaces

What is the need of XML namespace

- One of the major goals of XML is to add uniformity to tags.
- Potential problem
  - Several people create their own tags.
  - When combined, this may lead to ambiguity.

What is the need of XML namespace

Example:

<subject>Geometry</subject>

<subject>Cardiology</subject>

The first subject is the one we study in college, the other one in medicine – but there is nothing to differentiate between them

We can resolve this problem by using namespaces as follows:

<highschool:subject>Geometry</highschool:subject>

<medicine:subject>Cardiology</medicine:subject>

Both *highschool* and *medicine* are *namespace prefixes*

We can provide any namespace prefix we want, except for the reserve words

- A Namespace is a set of unique names.
- Namespace is a mechanisms by which element and attribute name can be assigned to a group.
- The Namespace is identified by URI(Uniform Resource Identifiers).
- XML Namespace is used to avoid element name conflict in XML document.

Namespaces: The Big Idea

- Solve the problem of ambiguity and provide more information about the element.
- Allow the element and attribute names to be distinguished when merging occurs.
- Provide a unique prefix to the beginning of every element and attribute name.
- Usually it is the Web address of the organization.
- The prefix that we add to an element to make it unambigious is called as a namespace prefix.
- The namespace prefix is defined in the root element and is used inside the elements to describe data.

- Technically, the prefix can be any URI (Uniform Resource Identifier).
  - URI is more generic than Uniform Resource Locator (URL)
  - URI can be any unique name

- When namespaces are used, the element and attribute names become two-part names
  - XML namespace and the actual element/attribute name

Uniform Resource Identifier (URI)

- A Uniform Resource Identifier (URI) is a string of characters which identifies an Internet Resource.
- The most common URI is the Uniform Resource Locator (URL) which identifies an Internet domain address. Another, not so common type of URI is the Universal Resource Name (URN).
- URL is used for lookup where as URI is used for reference
- The purpose of using an URI is to give the namespace a unique name.

Namespace Declarations

- Declare a namespace in a top-level element
  - All the elements and attributes under that element inherit the namespace
- Example

```
<book xmlns = “http://www.test.com”>
  <isbn>0-596-0058-8</isbn>
  <title>My first book</title>
  <author>Mr XYZ</author>
</book>
```

- All elements under the book element are a part of the namespace

- An XML namespace is declared using the reserved XML attribute.
- This attribute name must be started with "xmlns".
- Ex: <element xmlns:name = "URL">

- If a namespace is declared at an element that is not the root element
  - It applies only till the boundaries of that element
- Allows different namespaces for different elements in an XML document
  - One per element or sub-element, etc
- Syntax of a namespace declaration :

  <prefix:elementName xmlns: prefix ='URI'>

- If most elements in an XML file belongs to one namespace, we can make it the default namespace
  - No need to prefix elements belonging to that namespace
  - Defining default namespace
    - xmlns="namespace"

Namespace Shortcuts

- Makes namespace declarations and usage more readable
- Example

```
<zbooks:book xmlns:zbooks = “http://www.test.com”>
   <zbooks:isbn>0-596-0058-8</zbooks:isbn>
   <zbooks:title>My first book</ zbooks: title>
   <zbooks:author>Mr XYZ</ zbooks: author>
</zbooks:book>
```

- All elements starting with zbooks: belong to the *http://www.test.com* namespace

- Consider an XML file

```
<employees>
  <employee>
  <id>9662</id>
  <name>Ram</name>
  <hireDate>2001-08-13</hireDate>
  </employee>
  <employee>
  <id>10000</id>
  <name>Parag</name>
  <hireDate>2004-01-12</hireDate>
  </employee>
</employees>
```

- Now imagine that the payroll department wants to share this XML file and wants to add payroll data Modified XML file is shown below

```
<employees>
   <employee>
        <id>9662</id>
        <name>Ram</name>
        <hireDate>2001-08-13</hireDate>
        <salary> 10000 </salary>
        <taxes> 2000 </taxes>
   </employee>
   <employee>
        <id>10000</id>
        <name>Parag</name>
        <hireDate>2004-01-12</hireDate>
        <salary> 10000 </salary>
        <taxes> 2000 </taxes>
   </employee>
</employees>
```

- How to handle this change?
  - Update the schema owned by the HR department?

    OR
  - Separate the data items owned by HR and Payroll and make them responsible for their data items?
    - Use namespaces
    - Different “buckets” for data items owned by HR and Payroll

```
<HRData:employees>
   <HRData:employee>
   <HRData:id>9662</HRData:id>
   <HRData:name>Ram</HRData:name>
   <HRData:hireDate>2001-08-13</HRData:hireDate>
   <payrollData:salary> 10000 </payrollData:salary>
   <payrollData:taxes> 2000 </payrollData:taxes>
   </HRData:employee>
   <HRData:employee>
   <HRData:id>10000</HRData:id>
   <HRData:name>Parag</HRData:name>
   <HRData:hireDate>2004-01-12</HRData:hireDate>
   <payrollData:salary> 10000 </payrollData:salary>
   <payrollData:taxes> 2000 </payrollData:taxes>
   </HRData:employee>
</HRData:employees>
```

- Using a short-hand notation

```
<employees xmlns:hr="HRData" xmlns:py="payrollData">
   <hr:employee>
        <hr:id>9662</hr:id>
        <hr:name>Ram</hr:name>
        <hr:hireDate>2001-08-13</hr:hireDate>
        <py:salary> 10000 </py:salary>
        <py:taxes> 2000 </py:taxes>
   </hr:employee>
   <hr:employee>
        <hr:id>10000</hr:id>
        <hr:name>Parag</hr:name>
        <hr:hireDate>2004-01-12</hr:hireDate>
        <py:salary> 10000 </py:salary>
        <py:taxes> 2000 </py:taxes>
   </hr:employee>
</hr:employees>
```

- Syntax: **xmlns:prefix="Actual namespace" (where prefix is the shorthand prefix)**

- Effectively, we will create two names, one for HR and one for payroll
- Suppose now you want to send the payroll data to the income tax department in XML format
- What if another company also uses namespace shorthands such as `hr` and `py`?
  - Need to make namespaces globally unique

```
<employees xmlns:hr="http://www.test.com/hr/"
   xmlns:py="http://www.test.com/payroll/">
   <hr:employee>
        <hr:id>9662</hr:id>
        <hr:name>Ram</hr:name>
        <hr:hireDate>2001-08-13</hr:hireDate>
        <py:salary> 10000 </py:salary>
        <py:taxes> 2000 </py:taxes>
   </hr:employee>
   <hr:employee>
        <hr:id>10000</hr:id>
        <hr:name atf:fname=””>
<></>Parag</hr:name>
        <hr:hireDate>2004-01-12</hr:hireDate>
        <py:salary> 10000 </py:salary>
        <py:taxes> 2000 </py:taxes>
   </hr:employee>
</employees>
```

Qualified Names(QNames)

- A qualified name is a name subject to namespace interpretation
- XML namespaces provide us more information about elements in the form of qualifiers.
- QNames are used in place of element and attribute names.
- QNames have a prefix and a local part

  For eg:

  <http://www.test.com/book : title>

  – prefix : book
  – localpart : title

  or {http://www.test.com/book}title

Namspace Scoping

- Namespace scope means that the declaration is available for reference.
- The namespace of an element depends on:
  - The namespace prefix used.
  - The declaration of a default namespace.
- Attribute MUST be prefixed to be associated with a namespace
- When a namespace prefix is declared,it remains in scope for:
  - Attributes of the element where it is declared.
  - Child elements and their attributes of the element where it is declared.
  - Unless the prefix is redefined on a nested element.

Default Namespace

- Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

  xmlns="namespaceURI"

- To reduce the number of places where the namespace prefix needs to be used,we can specify a default namespace.
- A default namespace can be specified for an element and all its sub element
- For default namespace , we need to specify a namespace without a prefix. (just use xmlns without a prefix)
- For eg:

```
<employees xmlns="http://www.test.com/hr/"
 xmlns:py="http://www.test.com/payroll/">
```

Default Namespace

- Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

  xmlns="namespaceURI"

- To reduce the number of places where the namespace prefix needs to be used,we can specify a default namespace.
- A default namespace can be specified for an element and all its sub element
- For default namespace , we need to specify a namespace without a prefix. (just use xmlns without a prefix)
- For eg:

```
<employees xmlns="http://www.test.com/hr/"
 xmlns:py="http://www.test.com/payroll/">
```

```
<employees xmlns="http://www.test.com/hr/"
  xmlns:py="http://www.test.com/payroll/">
  <employee>
  <id>9662</id>
  <name>Ram</name>
  <hireDate>2001-08-13</hireDate>
  <py:salary> 10000 </py:salary>
  <py:taxes> 2000 </py:taxes>
  </employee>
  <employee>
  <id>10000</id>
  <name>Parag</name>
  <hireDate>2004-01-12</hireDate>
  <py:salary> 10000 </py:salary>
  <py:taxes> 2000 </py:taxes>
  </employee>
</employees>
```

Documents with Multiple Namespace

- Namespace also allow documents to use names from multiple namespace without interfering with each other.
- <book xmlns = "www.test.com"

```
        xmlns :amazon = " www.hello.com">
 <title> Tom </title>
 <isbn xmlns = "www.loc.com">
    12005436235
  </isbn>
 <amazon : skuno> A25 </amazon:skuno>
</book>
```

- In the above example , there are three namespace declaration:
  - Declaration of a default namespace on the root.
  - Declaration of a single prefix on the root element.
  - Declaration of a default namespace on the isbn element - the scope of this declaration is the isbn element and its children(if any)

Elements with no Namespace

- <book xmlns = "www.test.com"
  xmlns :amazon = " www.hello.com">
  <title> Tom </title>
  <isbn xmlns = " ">
  12005436235
  </isbn>
  <amazon : skuno> A25 </amazon:skuno>
  </book>

- The xmlns = " " syntax resets the default namespace for the scope in which it occurs. The <isbn> element is not in a namespace , because there is no default null namespace.
- If a name has no prefix and there is no default namespace , then the name is not in any namespace.

Attributes and Namespace

- There are two interacting rules that affect attributes and namespace:
  - Attributes are not affected by a default namespace declaration.
  - Attributes on a single element must be unique.

- For eg:

```
<bad xmlns : ns1 = "www.w3.org"
        xmlns : ns2 = "www.w3.org">
    <invalid  att = "1" att ="2" />
    <invalid  ns1:att="1" ns2:att="2" />
</bad>
<good xmlns : ns1 = "www.w3.org"
        xmlns = "www.test.org">
    <valid  a= "1" b ="2" />
    <valid  a="1" ns1:a="2" />
<good>
```

- both <invalid> elements are invalid because :
  - **First :**there are two unprefixed attributes of the same name.
  - **Second** : two attributes are also same because ns1 and ns2 are the prefix for the same namespace.

- both <valid> elements are valid because
  - **First :**a and b are unprefixed , and a is not the same as b.
  - **Second:** a is in no namespace (default declaration don`t affect attributes) and ns1: a attribute is in "www.w3.org" namespace- they are in different namespace.

Namespace Processing

- To deal with namespace XML parser needs the right API.
- The parser simply reports the prefix, localName, and URI associated with the element or attribute.
- There is no validation rules associated with Namespaces.

Example

```
<united : airplanes xmlns = "www.test.org"
        xmlns : united = "www.ual.com"
        xmlns : boeing = "www.boeing.com"
        xmlns:pratt = "www.pratt.com"
        xmlns : goodyear = "www.goodyear.com"
        xmlns :pire = "www.pire.com"
        xmlns :rolls = "www.rools.com"
       xmlns : airbus = "www.airbus.com">
  <boeing : airplane>
     <wing/>
     <pratt: engine/>
     <goodyear:tyre/>
  </boeing : airplane>
  <airbus:airplane>
     <wing/>
     <rolls:engine/>
     <pire : tire/>
  </airbus:airplane>
</united : airplanes>
```

Problems with Namespace

- In the XML 1.0 , namespace still have some problems :
  - Namespace recommendation came after XML 1.0, so it is not considered in the specification.
  - DTD don`t integrate well with namespace
  - Testing the equality of namespace is not handled by the parser.