XML Namespaces

Agenda

1 **XML Namespaces**

Objectives

At the end of this module, you will be able to

- Declare Namespaces
- Apply Namespaces to elements

XML Namespaces

Need for XML Namespaces

- A method to avoid conflicts between element types and attribute names when two (or more) different specifications are in use

- **For example:** Consider the XML code

```xml
<?xml version="1.0"
encoding="UTF-8"?>
<library>
    <book>
        <message>Welcome to ABC
        Library</message>
    </book>
    <book>
        <title>Java</title>
        <author>Schildt</author
        >
    </book>
</library>
```

Here, we have two very different elements that want to use the same name: *book*. The solution to this problem is to create *XML Namespaces*, which will differentiate between these two similarly named elements!

What are XML Namespaces?

Defines a way to group element and attribute names so that schemas created by one organization will not conflict with those created by another

Provide uniquely named elements and attributes in an XML instance

Defined by a W3C recommendation called Namespaces in XML

Each namespace defined in an XML document must be associated with a distinct uniform resource identifier (URI), which is usually a URL

XML Namespaces are collections of names. That is, they contain the *names* of element types and attributes, not the elements or attributes themselves.

Declaring Namespace

- A namespace is declared as an attribute of an element

- It is declared using reserved XML attribute ***xmlns***

- *xmlns* must have a unique value that is not shared by any other namespace in the document

- The value used is the URI or the more commonly used URL

- For example:

    xmlns="http://www.abclibrary.com/fiction"

    xmlns:latestseries="http://www.xyzlibrary.com/mystery"

Declaring Namespace (Contd.).

- Namespaces can be mapped to prefixes in namespace declarations

- For example: A namespace is declared as
    <book xmlns:actionseries="http://www.abclibrary.com" />

- In the attribute xmlns:actionseries
    - xmlns is a reserved word used only to declare a namespace
    - The prefix "actionseries" is bound with the namespace "http://www.abclibrary.com"

- To use a namespace, first bind it with a prefix and then use that prefix wherever required
- Note that the prefixes are used only as a placeholder and must be expanded by the namespace-aware XML parser to use the actual namespace bound to the prefix.

Declaring Namespace - Example

- Consider the XML code:

```
<?xml version="1.0"?>
<book xmlns:actionseries="http://www.abclibrary.com">
    <actionseries:title>Bourne Supremacy</actionseries:title>
    <actionseries:author>Robert Ludlum</actionseries:author>
</book>
```

- The elements title and author are associated with namespace http://www.abclibrary.com

Namespace to avoid collision

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library SYSTEM "library.dtd">
<library>
    <m2:book xmlns:m2='http://www.simple.org/general'>
        <m2:message>Welcome to ABC Library</m2:message>
    </m2:book>
    <m3:book xmlns:m3='http://www.simple.org/technical'>
        <m3:title>Java</m3:title>
        <m3:author>Schildt</m3:author>
    </m3:book>
</library>
```

library.xml

- To rectify the overlap, we use two different made up URIs for book element.
- By placing a namespace prefix before our elements, we have solved the overlapping problem!

Namespace to avoid collision (Contd.).

- DTD library.dtd corresponding to the XML document library.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!ELEMENT library (m2:book|m3:book)*>
<!ELEMENT m2:book (m2:message)*>
<!ATTLIST m2:book xmlns:m2 CDATA #IMPLIED>
<!ELEMENT m2:message (#PCDATA)>
<!ELEMENT m3:book (m3:title|m3:author)*>
<!ATTLIST m3:book xmlns:m3 CDATA #IMPLIED>
<!ELEMENT m3:title (#PCDATA)>
<!ELEMENT m3:author (#PCDATA)>
```

Qualified Names

- They are names given to an element or an attribute, where name itself has an additional importance as it points to an URI location
- An example of a qualified name - Element type:

```
<x xmlns:edi='http://ecommerce.org/schema'>
    <edi:price units='Euro'>32.18</edi:price>
</x>
```

- An example of a qualified name - Attribute Name:

```
<x xmlns:edi='http://ecommerce.org/schema'>
    <lineItem edi:taxClass="exempt">Baby Food</lineItem>
</x>
```

Default Namespace

- A default namespace is declared as:

```
<AnElement xmlns="http://www.simple.com" />
```

- For example:

```
<book xmlns="http://www.adventurelibrary.com">
  <title>Bourne Identity</title>
  <author>Robert Ludlum</author>
</book>
```
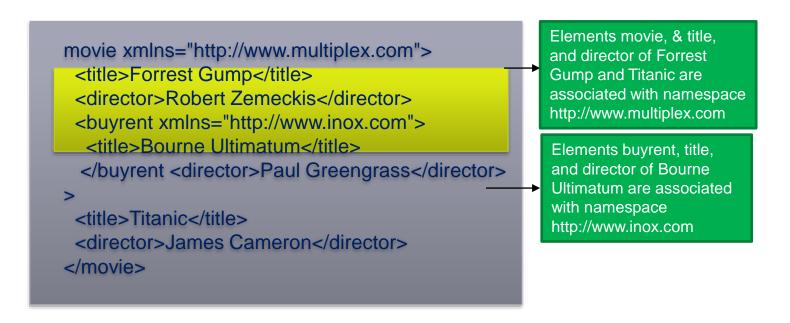
- Here the element names book, title, and author are associated with the namespace http://www.adventurelibrary.com

Default Namespace scope

- The scope of a namespace begins at the element where it is declared

- It applies to the entire content of that element, unless overridden by another namespace declaration

```
movie xmlns="http://www.multiplex.com">
  <title>Forrest Gump</title>
  <director>Robert Zemeckis</director>
  <buyrent xmlns="http://www.inox.com">
    <title>Bourne Ultimatum</title>
  </buyrent <director>Paul Greengrass</director>
>
  <title>Titanic</title>
  <director>James Cameron</director>
</movie>
```

Elements movie, & title, and director of Forrest Gump and Titanic are associated with namespace http://www.multiplex.com

Elements buyrent, title, and director of Bourne Ultimatum are associated with namespace http://www.inox.com

Namespace scope (Contd.).

- Multiple namespace prefixes can be declared as attributes of a single element

```
<!- Both namespace prefixes are available throughout -->

<bk:book xmlns:bk='urn:loc.gov.books'
                 xmlns:isbn='urn:ISBN:0-395-36341-6'>
    <bk:title>Oliver Twist</bk:title>
    <isbn:number>1658739128</isbn:number>
</bk:book>
```

Summary

In this module, you were able to

- Declare Namespaces
- Apply Namespaces to elements

Thank You