

# Lecture 5 (1/2 hour)

---

# Namespaces

Sang Shin

Java™ Technology Evangelist

sang.shin@sun.com

(You can use this material in any way you want,  
but if you can drop me an email when you do,  
that will be greatly appreciated.)

# Topics



- Need for Namespaces
- Namespace syntax
- Default Namespace
- Parsers and Namespaces
- DTDs and Namespaces

# Need for Namespaces

---

- A XML document could use **multiple XML vocabularies**
- Examples
  - ◆ XHTML document might contain both SVG and MathML elements
  - ◆ XSLT stylesheet contain both XSLT and result-tree elements
- How to avoid **Name collisions?**
  - ◆ Both SVG and MathML have “set” element

# Need for Namespaces

- Example 4-2 from “XML in a Nutshell”

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<catalog>

  <RDF>
    <Description about="http://ibiblio.org/examples/impressionists.xml">
      <!-- title of a webpage -->
      <title> Impressionist Paintings </title>
      <creator> Elliotte Rusty Harold </creator>
      <description>
        A list of famous impressionist paintings organized
        by painter and date
      </description>
      <date>2000-08-22</date>
    </Description>
  </RDF>
```

# Continued

```
<painting>
```

```
  <!-- title of a painting -->
```

```
  <title>Memory of the Garden at Etten</title>
```

```
  <artist>Vincent Van Gogh</artist>
```

```
  <date>November, 1888</date>
```

```
  <description>
```

```
    Two women look to the left. A third works in her garden.
```

```
  </description>
```

```
</painting>
```

```
<painting>
```

```
  <title>The Swing</title>
```

```
  <artist>Pierre-Auguste Renoir</artist>
```

```
  <date>1876</date>
```

```
  <description>
```

```
    A young girl on a swing. Two men and a toddler watch.
```

```
  </description>
```

```
</painting>
```

```
  <!-- Many more paintings... -->
```

```
</catalog>
```

# Need for Namespaces

---

- Changing element names (to avoid collision) is not a convenient option
- Some collisions are inevitable
  - ◆ If both are standard vocabularies
    - SVG's "set" vs. MathML's "set"
- Grouping names is useful anyway
  - ◆ XSLT processor needs to know which are XSLT instructions and which are result-tree elements

# Namespace Example

- Example 4-3 from “XML in a Nutshell”

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<catalog>
```

```
  <rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#">
    <rdf:Description xmlns:dc="http://purl.org/dc/"
      about="http://ibiblio.org/examples/impressionists.xml">
      <dc:title> Impressionist Paintings </dc:title>
      <dc:creator> Elliotte Rusty Harold </dc:creator>
      <dc:description>
        A list of famous impressionist paintings organized
        by painter and date
      </dc:description>
      <dc:date>2000-08-22</dc:date>
    </rdf:Description>
  </rdf:RDF>
```

...

# Namespace Syntax

---

- Two parts
  - ◆ Namespace declaration
  - ◆ Elements and attributes

# Namespace Declaration

- A *prefix* is associated with **URI**
- The association is defined as an attribute within an element
  - ◆ *xmlns:prefix*
- xmlns is Namespaces keyword, prefix is user-defined

```
<classes xmlns:XMLclass="http://www.brandeis.edu/rseg-0151-g">  
  <XMLclass:syllabus>  
    ...  
  </XMLclass:syllabus>  
</classes>
```

# Namespace Declaration

---

- Can be declared in a root element or at lower level element
- Same prefix can be redefined within a same document
  - ◆ Scope of Namespace declaration is within the element where it is defined

# Elements and attributes with Namespace prefix

---

- Example
  - ◆ XMLClass:syllabus
  - ◆ svg:set
  - ◆ mathml:set
- *prefix: local part*
  - ◆ prefix identifies the namespace an element and attribute belongs to
  - ◆ local part identifies the particular element or attribute within the namespace
  - ◆ Qualified name

# Elements and attributes with Namespace prefix

---

- Prefix
  - ◆ Can be composed from any legal XML name character except the “:”
  - ◆ “xml” (in any case combination) are reserved
- Local part
  - ◆ Cannot contain “:”

# Namespace URI

---

- URI cannot be prefix
  - ◆ “/”, “%”, and “~” are not legal in XML element names
- URI could be **standardized** while prefixes are just convention
- URI are just “identifiers”
  - ◆ URI does not have to be in “http” form
  - ◆ URI does not have to be resolved
  - ◆ It is like a “constant value”

# Default Namespace

---

- Denoted with **xmlns** attribute with **no prefix**
- Applied only to unprefixed element and its descendant elements
- Applies only to elements not attributes

# Default Namespace

```
<?xml version="1.0"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <head><title>Three Namespaces</title></head>
  <body>
    <h1 align="center">An Ellipse and a Rectangle</h1>
    <svg xmlns="http://www.w3.org/2000/svg"
         width="12cm" height="10cm">
      <ellipse rx="110" ry="130" />
      <rect x="4cm" y="1cm" width="3cm" height="6cm" />
    </svg>
    <p xlink:type="simple" xlink:href="ellipses.html">
      More about ellipses
    </p>
    <p xlink:type="simple" xlink:href="rectangles.html">
      More about rectangles
    </p>
    <hr/>
    <p>Last Modified May 13, 2000</p>
  </body>
</html>
```

# Parsers and Namespaces

---

- Namespaces were after-thought of XML 1.0
- Backward compatibility
  - ◆ SAX 1.0 and DOM level 1 Parsers (XML 1.0) are not namespace-aware
  - ◆ They can still read Namespace-enabled XML document
- SAX 2.0 and DOM level 2 are now namespace-aware

# Namespaces and DTDs

---

- Namespaces and DTDs are independent
- Name of an element in a XML document must match the name in DTD regardless of namespaces
  - ◆ Validator does not know if a name is made of namespace prefix and local part
- W3C XML Schema is Namespaces-aware

# Parameter Entity Reference for Prefixes

---

- Issue
  - ◆ Changing prefix in XML documents forces changes in DTD
  - ◆ Same DTD could be used for multiple XML documents
- Solution
  - ◆ Use parameter entity reference for more flexible approach

# Parameter Entity Reference Primer

---

- Like an **alias** within DTD
- Can be **redefined** - powerful feature
- In the event of conflicting entity declaration, the first one encountered takes precedence
- As a XML document writer, **you can override the definition of parameter entity** by redefining them in your internal DTD

# Usage of Parameter Entity Reference with Namespaces

---

- 1st step - parameterize prefix  
`<!ENTITY % dc-prefix "dc">`
- 2nd step - parameterize qualified names  
`<!ENTITY % dc-title "%dc-prefix::title">`  
`<!ENTITY % dc-creator "%dc-prefix::creator">`

# Continued

---

- 3rd step - define others using parameter entity references

```
<!ELEMENT %dc-title; (#PCDATA)>
<!ELEMENT %dc-creator; (#PCDATA)>
<!ELEMENT rdf:Description
    ((%dc-title; | %dc-creator; )*)
```
- 4th step - **redefine prefix** in your internal DTD

```
<!DOCTYPE topic [
    <!ENTITY % dc-prefix "somethingElse">
]>
```

# Summary

---

- Designed to avoid name collisions
- Attaching a prefix to each element and attribute
- Each prefix is mapped to a URI
- Default URI can be provided
- Use parameter entity reference for flexible prefix'ing

# References

---

- “XML in a Nutshell” written by Elliotte Rusty Harold & W. Scott Means, O’Reilly, Jan. 2001(1st Edition), Chapter 4 “Namespaces”

