XML Namespaces

- What are XML Namespaces?
- Namespace Syntax
- Default Namespaces

Name Ambiguity

- If you write XML that aggregates XML from multiple sources, there's a possibility of name ambiguity

Human Resources XML

```
<employee>
 <name>Bill</name>
 <address>
  123 Elm
 </address>
 <id>123</id>
 . . .
</employee>
```

IT Department XML

```
<server>
 <location>
  Bill's Office
 </location>
 <address>
  12.34.56.78
 </address>
 . . .
</server>
```

The scenario here is that at a large company, two departments independently define XML content and write separate XML-processing applications. As long as each application remains separate, there's no problem.

But if the company wants to write a new application that aggregates XML from the two departments, now the element name "address" is ambiguous. In other words, when this new application parses the XML, how can it distinguish between an HR "address" and an "IT" address?

One possible way is for the application to examine the hierarchy of where "address" is defined, but this complicates the application even in this simple example.

One Solution

- One way to solve the problem is to use application-defined name **prefixes**
- Can you think of any unique strings on the Web that would work for prefixes across enterprises?

Human Resources XML

```
<hr-employee>
  <hr-name>Bill</hr-name>
  <hr-address>
    123 Elm
  </hr-address>
  <hr-id>123</hr-id>
  . . .
</hr-employee>
```

IT Department XML

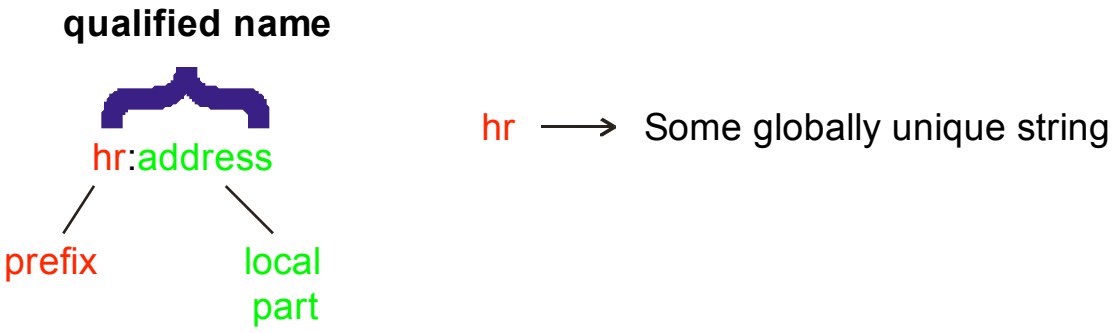
```
<it-server>
  <it-location>
    Bill's Office
  </it-location>
  <it-address>
    12.34.56.78
  </it-address>
  . . .
</it-server>
```

This solution is to use an ad-hoc approach: have each department use an application-defined prefix that eliminates ambiguity.

While this solution is feasible, it's not formal, and doesn't scale well across multiple enterprises.

Introduction to XML Namespaces

- **Namespaces in XML** became an official W3C Recommendation in 1999
- It provides a standard technique to avoid name collisions and ambiguity



The Namespaces in XML, Third Edition Recommendation is available at:

`http://www.w3.org/TR/2009/REC-xml-names-20091208/`

Namespace Syntax

- To assign the prefix, you write an **xmlns** attribute on an element that relates the prefix to a URI
- When you parse XML with namespaces, internally, the parser identifies elements by their local name and namespace URI
- The prefix definition is inherited by any descendent elements

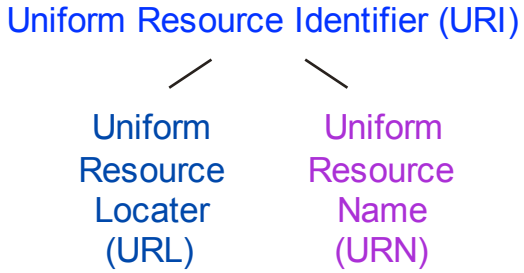
```
<hr:employee
   xmlns:hr="http://mycompany.com/HR">
  <hr:name>Bill</hr:name>
  <hr:address>
    123 Elm
  </hr:address>
  <hr:id>123</hr:id>
  . . .
</hr:employee>
```

The "xmlns" attribute is defined by the Namespace Recommendation to to bind the prefix to the URI.

Note that here we specified the prefix on the "employee" element itself - therefore, the "employee" element itself is considered to be in the namespace.

The Namespace Myth

- The Namespace Recommendation uses **Uniform Resource Identifiers** (URI) for the unique strings, but the URIs do not need to refer to actual Web resources




```
<hr:employee
    xmlns:hr="urn:xyz-123">
  <hr:name>Bill</hr:name>
  <hr:address>
    123 Elm
  </hr:address>
  <hr:id>123</hr:id>
  . . .
</hr:employee>
```


Many people are familiar with URLs, less with URNs. Both are a type of URI.

URNs do not specify a location, or address, while URLs do. Namespace URIs don't need a location since they they are simply qualifiers to avoid ambiguity, so using URNs works well. That being said, most XML namespace URIs currently are defined as URLs, where we use the uniqueness of the string and ignore its location meaning.

Using Multiple Namespaces

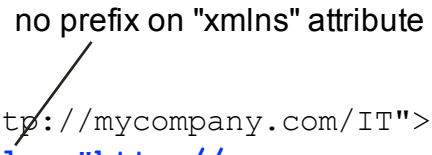
- To avoid name ambiguity, an XML document can define multiple namespaces and prefixes

```
<assets
  xmlns:hr="http://mycompany.com/HR"
  xmlns:it="http://mycompany.com/IT">
  <hr:employee>
    <hr:name>Bill</hr:name>
    <hr:address>123 Elm</hr:address>
  </hr:employee>
  <it:server>
    <it:location>Bill's Office</it:location>
    <it:address>12.34.56.78</it:address>
  </it:server>
</assets>
```

Here we define two prefixs, one for each department in the hypothetical company. Now there's no ambiguity on the "address" elements, either for the human reader, or for a program that parses this XML.

Using a Default Namespace

- To cut down on the bulk of the XML content (and save typing), you can optionally define a **default namespace**
- Default namespaces can make understanding the content more difficult for human readers



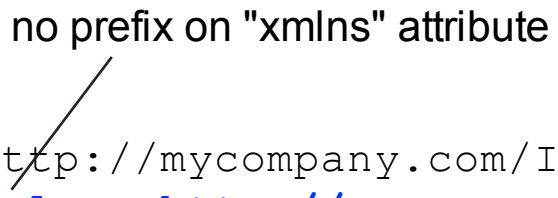

```
<assets
  xmlns:it="http://mycompany.com/IT">
  <employee xmlns="http://mycompany.com/HR">
    <name>Bill</name>
    <address>123 Elm</address>
  </employee>
  <it:server>
    <it:location>Bill's Office</it:location>
    <it:address>12.34.56.78</it:address>
  </it:server>
</assets>
```

Default namespaces using the "xmlns" attribute without specifying a prefix.

Like all namespace declarations, it applies (potentially) to to the element that carries the "xmlns" attribute (if it's not prefixed) and all unprefixed descendent elements.

Elements With No Namespace

- In which namespace is the **assets** element?




```
<assets
  xmlns:it="http://mycompany.com/IT">
  <employee xmlns="http://mycompany.com/HR">
    <name>Bill</name>
    <address>123 Elm</address>
  </employee>
  <it:server>
    <it:location>Bill's Office</it:location>
    <it:address>12.34.56.78</it:address>
  </it:server>
</assets>
```

If have XML content that uses namespaces, and you see a element without a prefix, there are two possibilities:

1. The element is in a default namespace

2. The element is in no namespace at all

The only way to determine the difference is to look at the element in question and then look at all of its ancestor elements, looking for an "xmlns" attribute with no prefix. If you find none, then the element in question is in no namespace.

Though the algorithm described in the last paragraph is no problem for a parser, it does make reading the XML more difficult for the human reader, which does violate one the guidelines for XML itself.

Namespaces and Attributes

- Attribute names can be namespace qualified, too
- One caveat: default namespaces do not apply to attributes!
- Quiz: Identify the namespace for each element and attribute in the following content:

```
<roster xmlns="http://big.edu"
        xmlns:big="http://big.edu"
        xmlns:small="http://small.edu">

  <student
    big:name="Bill" small:name="George"/>

  <student
    name="Sam" />

</roster>
```

The purpose of XML namespaces is to avoid ambiguity. According to the XML Recommendation, attribute names on a given element must be unique, so using namespace prefixes on attributes seems a bit odd.

However, the syntax is allowed, and is fairly common.

Try It Now!

- Use Windows Explorer to copy business-card.xml to business-card-namespace.xml
- Edit business-card-namespace.xml and define a namespace with URL: http://mycompany.com/Training

Namespaces and Parsers

- The two most well known APIs for parsing XML are the **Document Object Model** (DOM) and the **Simple API for XML** (SAX)
- Both APIs needed an update once the W3C finished the Namespaces in XML Recommendation
- If you wish to use namespaces in XML applications, therefor be sure that your parser implements at least DOM Level 2 or SAX version 2

Issues With Namespaces

- The namespace URI does not necessarily correspond to an actual resource on a network
- Default namespaces can be confusing to human readers, especially in large documents
- Default namespaces don't apply to attributes
- DTDs and namespaces don't mix (DTDs came before namespaces) -- XML Schemas work fine with namespace-qualifed XML

Namespace Quiz

- Make a table that shows the namespace prefix and URI for each element and attribute in the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <env:Body>
  <ns1:getModeratorResponse
      env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
      xmlns:ns1="http://mycompany.com/Meeting">
   <moderator xsi:type="xsd:string">Sue Smith</moderator>
  </ns1:getModeratorResponse>
 </env:Body>
</env:Envelope>
```

Chapter Summary

In this chapter, you learned:

- The rationale for XML namespaces
- The syntax of namespaces, including default namespaces
- The advantages and disadvantages of using namespaces