Fundamentals of Web Programming [a]

Namespaces (NS)

Teodor Rus

rus@cs.uiowa.edu

The University of Iowa, Department of Computer Science

Motivation

It is often convenient to construct XML documents that use tag sets defined for and used by other documents.

Rationale

- Application modularity: if a tags set is available, better use it than reinvent it;
- Documents containing multiple tag sets pose problems of recognition and collision;
- Software modules need to be able to recognize the elements and attributes which they are designed to process even when collision may occur.

Example Tag Set Reuse

Suppose that we develop an XML document for a furniture catalog which contains terms:
`chair, sofa, table (data), table (furniture), etc.`

- Since table (data) is a tag used in XHTML, it is convenient to reuse the XHTML $<$ `table` $>$ element and all its subelements and attributes;
- However, since table is an important furniture, we also need to use the element $<$ `table` $>$ furniture;
- An XML namespace is a mechanism that allow an XML processor to resolve the collision between the two elements $<$ `table` $>$.

Origin

Namespaces are commonly used as *abstract containers* providing context for the items (names, or technical terms, or words) they holds.

Thus, a namespace allows disambiguation of homonym items (items having the same name) residing in different namespaces.

Facts

1. Names in a namespace cannot have more than one meaning. Since the valid meaning of a name can change with the namespace, a namespace is also called *context*.
2. For many programming languages, a namespace is the scope of identifiers. Local and global names are determined by language specific scope definition rules.
3. In an operating system, an example of namespace is a directory. It contains items which must have unique names.

XML Namespace

An XML namespace is a collection of element and attribute names used in XML documents.

- The name of a namespace usually has the form of a URI;
- A namespace for the elements and attributes of the hierarchy rooted at a particular element is declared as the value of the attribute xmlns;
- The form of a namespace declaration for the hierarchy rooted at the element elemName is:

  ```
  <elemName xmlns[:prefix] = URI>
  ```

  where `prefix`, if included, is the name attached to the names in the namespace.
- If the `prefix` is not included, the namespace is the default for the document.

Prefix Usage

A prefix is used for two reasons:

1. Most URI are too long to be typed on every occurrence of every name from the namespace;
2. URI include characters that are illegal in XML documents.

Note: The `elemName` for which a namespace is declared is usually the root of the document.

Example: the namespace for XHTML documents is declared by:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
```

Example Prefixed Namespace

Consider the following namespace declaration:

```
<birds xmlns:bd = "http://www.audubon.org/names/species">
```

Meaning: within the `birds` element, including all of its children elements, the name from the given namespace must be prefixed with `bd`, as in `<bd:lark>`.

Facts

1. An element may have more than one namespace declaration. In this case the attributes that distinguish namespace declarations have the forms `xmlns` : $\texttt{ns}_1$, . . ., `xmlns` : $\texttt{ns}_n$, where ns_1, . . ., ns_n are different strings.
2. **Default namespace** of an element is the namespace declared with the value of the attribute `xmlns` (note, with no prefix specification).
3. **Example multi-namespace declaration:**

```
<birds xmlns:bd = "http://www.audubon.org/names/species"
       xmlns:html = "http://www.w3.org/1999/html">
```

Example Namespace Use

```
<states xmlns = "http://www.states-info.org/states"
        xmlns:cap = "http://www.states-info.org/state-capitals" >
<state>
   <name> South Dakota </name>
   <population> 777888 </population>
   <capital>
      <cap:name> Pierre </cap:name>
      <cap:population> 111222 </cap:population>
   </capital>
</state>
<!-- More states here -->
</states>
```

Observations

The following observations are pertinent to the previous example:

1. Each `state` element has `name` and `population` elements from both spaces. They are distinguished by the prefix name of the namespace;
2. Attribute names are not included in the namespace because attribute names are local to elements. Hence, a tag set may use the same attribute name in more than one element without causing ambiguity;
3. If an XML document uses a DTD and a prefixed name, the DTD must define an element with exactly the same prefix and name.

Namespaces and Web Resource

Because of their form it is tempting to think that a namespace is a Web resource that lists element names!

But this is never the case.

1. The standard namespaces (e.g. `http://www.w3.org/1999/xhtml`) often are valid URL-s but they are documents that describe far more than a set of element names;
2. User defined namespaces need not use URI form, also this is a good way to prevent conflict with namespace names;
3. The Formal management of namespace is the subject of the document `http://www.w3.org/TR/2009/REC-xml-names-20091208/`.

Basic Concepts

1. An XML namespace is identified by a URI reference (`RFC 3986:Uniform Resource Identifier (URI):Generic Syntax`). Element and attribute names may be placed in an XML namespace using the mechanism described here.
2. An expanded name is a pair consisting of a namespace name and a local name. For a name N in the namespace identified by URI I the namespace name is I; for a name N that is not in a namespace, the namespace name of N has no value; in either case the local name is N.
3. A qualified name is a name subject to namespace interpretation; syntactically qualified names are either *prefixed names* or *unprefixed names*.

URI as Namespace Names

1. The empty string ϵ is a legal URI but it cannot be used as a namespace name;
2. The use of relative URI references, including some document references, in namespace declarations is deprecated (see `http://www.w3.org/2000/09/xppa`).

Comparing URI References

URI are treated as strings. Consequently:

1. Two URIs are identical iff they are identical strings. The comparison is case sensitive and no %-escaping is done or undone.
2. In a namespace declaration, the URI reference is the normalized value of the attribute, so the replacement of XML character and entity references has already been done before any comparison.
3. Examples different URI references:

```
http://www.example.org/wine
http://www.Example.org/wine
http://www.example.org/Wine
http://www.example.org/~wine
http://www.Example.org/%7ewine
```

Declaring Namespaces

The following BNF rules specify attributes used in a namespace declaration:

```
NSAttName ::= PrefixedAttName | DefaultAttName
PrefixedAttName := 'xmlns:' NCName
DefaultAttName := 'xmlns'
NCName ::= Name-(Char* ':') /* An XML name minus ":" */
```

Note:

- If the attribute is a `PrefixedAttName` then its NCName component gives the namespace prefix;
- If the attribute is a `DefaultAttName` then the namespace name in the attribute value is that of default namespace.

Example NS Declaration

```
<x xmlns:edi = "http://ecommerce.example.org/schema">
<-- Here "edi" prefix is bound to
    "http://ecommerce.example.org/scheme for x lement -->
```

Reserved Prefixes

1. The prefix `xml` is bound to the namespace name

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

2. The prefix `xmlns` is used only to declare namespace bindings and is by definition bound to the namespace name

   `http://www.w3.org/2000/xmlns/`

3. Prefixes that begin with the three-letter sequence `x, m, l` in any combination are reserved. This means that users should not use them and processors should not treat them as fatal errors.

Qualified Names

XML qualified names are defined by the following BNF rules:

```
QName ::= PrefixedName | UnprefixedName
PrefixedName ::= Prefix ':' LocalPart
UnprefixedName ::= LocalPart
Prefix ::= NCName
LocalPart ::= NCName
```

Using Qualified Names

XML documents conforming to the namespace usage, element names are given as qualified names defined by the following BNF rules:

```
STag ::= '<' QName (S Attribute)* S? '>'
ETag ::= '<' QName S? '>'
EmptyElemTag ::= '<' QName (S Attribute)* S? '/>'
Attribute ::= NSAttribute Eq AttValue | QName Eq AttValue
```

Examples

- A qualified name serving as an element name:

```
<!-- the "price" element's namespace is
     http://ecommerce.example.org/schema -->
<edi:price xmlns:edi = "http://ecommerce.example.org/schema"
     units = "Euro" >
   32.18
</edi:price>
```

- A qualified name serving as an attribute name:

```
<x xmlns:edi = "http://ecommerce.example.org/schema">
<!-- the "taxClass" attribute's name space is
     http://ecommerce.example.org/schema -->
<lineItem edi:taxClass = "excempt"> Baby food </lineItem>
```

Restriction

- Namespace constraint: Prefix Declared
  The namespace prefix, unless it is xml or xmlns, MUST have been declared in a namespace declaration attribute in either the start-tag (`STag`) of the element where the prefix is used or in an ancestor element (i.e., an element in whose content the prefixed markup occurs).
- Namespace constraint: No Prefix Undeclaring
  In a namespace declaration for a prefix (i.e., where the `NSAttName` is a `PrefixedAttName`), the attribute value MUST NOT be empty.

Observations

1. The above constraint may lead to operational difficulties in cases where the namespace declaration attribute is provided, not directly in the XML document entity, but via a default attribute declared in an external entity. Such declarations may not be read by software which is based on a non-validating XML processor.
2. Many XML applications, presumably including namespace-sensitive ones, fail to require validating processors.
3. If correct operation with such applications is required, namespace declarations MUST be provided either directly or via default attributes declared in the internal subset of the DTD.

Qualified names in DTDs

Element names and attribute names are also given as qualified names when they appear in declarations in the DTDs, defined by BNF rules:

```
doctypedecl ::= "<!DOCTYPE" S QName
        (S ExternalID)?S?("["(markupdecl|PEReference|S)*"]"S?)?">"
elementdecl ::= "<!ELEMENT" S QName S contentspec S? ">"
contentspec ::= (QName | choice | seq) (’?’ | ’*’ | ’+’)?
Mixed  ::= "(" S? "#PCDATA" (S? "|" S? QName)* S? ")*" |
           "(" S? "#PCDATA" S? ")"
AttlistDecl ::= "<!ATTLIST" S QName AttDef* S? ">"
AttDef      ::= S (QName | NSAttName) S AttType S DefaultDecl
```

Note: Here ’S’ denotes the ’white space’.

Observations

1. DTD-based validations are not namespace-aware in the following sense: a DTD constrains the elements and attributes that may appear in a document by their uninterpreted names, not by (namespace name, local name) pairs.
2. To validate a document that uses namespaces against a DTD, the same prefixes must be used in the DTD as in the instance.
3. A DTD may however indirectly constrain the namespaces used in a valid document by providing #FIXED values for attributes that declare namespaces.

Namespace Scoping

The scope of a namespace declaration declaring a prefix extends from the beginning of the start-tag in which it appears to the end of the corresponding end-tag, excluding the scope of any inner declarations with the same NSAttName part.
In the case of an empty tag, the scope is the tag itself.

- A namespace declaration applies to all element and attribute names within its scope whose prefix matches that specified in the declaration.
- The expanded name corresponding to a prefixed element or attribute name has the URI to which the prefix is bound as its namespace name, and the local part as its local name.

Example

```
<?xml version="1.0"?>
<html:html xmlns:html='http://www.w3.org/1999/xhtml'>
  <html:head><html:title>Frobnostication</html:title></html:head>
  <html:body>
     <html:p> Moved to
        <html:a href='http://frob.example.com'>here.</html:a>
     </html:p>
  </html:body>
</html:html>
```

Multiple Namespaces

Multiple namespace prefixes can be declared as attributes of a single element, as shown below:

```
<?xml version="1.0"?>
<!-- both namespace prefixes are available throughout -->
<bk:book xmlns:bk = "urn:loc.gov:books"
         xmlns:isbn = "urn:ISBN:0-395-36341-6">
    <bk:title>Cheaper by the Dozen</bk:title>
    <isbn:number>1568491379</isbn:number>
</bk:book>
```

Namespace Defaulting

The scope of a default namespace declaration extends from the beginning of the start-tag in which it appears to the end of the corresponding end-tag, excluding the scope of any inner default namespace declarations.

In the case of an empty tag, the scope is the tag itself.

Facts

1. A default NS declaration applies to all unprefixed element names within its scope.
2. Default NS declarations do not apply directly to attribute names; the interpretation of unprefixed attributes is determined by the element on which they appear.
3. If there is a default NS declaration in scope, the expanded name corresponding to an unprefixed element name has the URI of the default NS as its NS name.
4. If ∄ default NS declaration in scope, the NS name has no value.
5. The NS name for unprefixed attribute names has no value.
6. In all cases, the local name is local part (which is of course the same as the unprefixed name itself).

Example 1

```
<?xml version="1.0"?>
<!-- elements are in the HTML namespace, in this case by default -->
<html xmlns = "http://www.w3.org/1999/xhtmls">
  <head><title>Frobnostication</title></head>
  <body>
    <p>
     Moved to <a href='http://frob.example.com'>here</a>.
    </p>
  </body>
</html>
```

Example 2

```
<book xmlns='urn:loc.gov:books'
      xmlns:isbn='urn:ISBN:0-395-36341-6'>
    <title>Cheaper by the Dozen</title>
    <isbn:number>1568491379</isbn:number>
    <notes>
      <!-- make HTML the default namespace for some commentary -->
      <p xmlns='http://www.w3.org/1999/xhtml'>
          This is a <i>funny</i> book!
      </p>
    </notes>
</book>
```

Example 3

A larger example of namespace scoping:

```
<?xml version="1.0"?>
<!-- initially, the default namespace is "book" -->
<book xmlns = "urn:loc.gov:books"
      xmlns:isbn =s "urn:ISBN:0-395-36341-6s">
    <title>Cheaper by the Dozen</title>
    <isbn:number>1568491379</isbn:number>
    <notes>
      <!-- make HTML the default namespace for some commentary -->
      <p xmlns = "http://www.w3.org/1999/xhtml">
          This is a <i>funny</i> book!
      </p>
    </notes>
</book>
```

Attribute Value in Default NS

The attribute value in a default namespace declaration MAY be empty. This has the same effect, within the scope of the declaration, as there being no default namespace.

Example

```
<?xml version='1.0'?>
<Beers> <!-- the default namespace inside tables is that of HTML -->
  <table xmlns = "http://www.w3.org/1999/xhtml">
   <th><td>Name</td><td>Origin</td><td>Description</td></th>
   <tr> <!-- no default namespace inside table cells -->
     <td><brandName xmlns = ""> Huntsman </brandName></td>
     <td><origin xmlns = ""> Bath, UK</origin></td>
     <td>
       <details xmlns = ""><class>Bitter</class><hop>Fuggles</hop>
         <pro>Wonderful hop, light alcohol, good summer beer</pro>
         <con>Fragile; excessive variance pub to pub</con>
         </details>
       </td>
    </tr>
  </table>
</Beers>
```

Uniqueness of Attributes

In XML documents conforming to this spec, no tag may contain two attributes which:

1. have identical names, or
2. have qualified names with the same local part and with prefixes bound to identical namespace names.

This constraint is equivalent to requiring that no element have two attributes with the same expanded name.

Example Illegal Tags

- Each of the bad empty-element tags is illegal in the following:

```
<!-- http://www.w3.org is bound to n1 and n2 -->
<x xmlns:n1="http://www.w3.org"
   xmlns:n2="http://www.w3.org" >
  <bad a="1"     a="2" />
  <bad n1:a="1"  n2:a="2" />
</x>
```

- However, each of the following good empty-element tag is legal
  (the second because the default namespace does not apply to attribute names)

```
<!-- http://www.w3.org is bound to n1 and is the default -->
<x xmlns:n1 = "http://www.w3.org"
   xmlns = "http://www.w3.org" >
  <good a="1"     b="2" />
  <good a="1"     n1:a="2" />
</x>
```