ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

Εισαγωγή στην Επιστήμη και Τεχνολογία των Υπηρεσιών

Ενότητα 3: Namespaces

Χρήστος Νικολάου
Τμήμα Επιστήμης Υπολογιστών

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.

- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ
2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

XML
Namespaces
605.444 / 635.444

David Silberberg

Lecture 7

Limitations of Syntax Definition So Far

- The XML documents that we have created work well within a single domain
  - XML documents represent data structures with values
  - All applications that generate and use the XML file must agree to a single convention for tags and attributes
  - For example, <frequent> and <casual> customers have the same definition
  - Similarly, <name> is always a person name with lastname first.
- In a small environment, this is fine
- However, XML files are meant to be exchanged by many individuals from different organizations
- It is impractical to expect that everyone use the same terms to mean the same things

Possible Solutions?

- Can ensure that everyone using a vocabulary set understands its environment and context
- However, this does not address the problem of combining XML documents
- Companies might want to produce an XML file that combines *customer, purchasing,* and *manufacturing* data
  - A customer <bill> might represent information about the bill sent to a customer.
  - A purchasing <bill> might represent information about the bill received from suppliers.
  - A manufacturing <bill> might represent bill-of-material information

Possible Solutions? (cont.)

- If an XML document that contains all three <bill>'s, a parser can get confused
  - Perhaps, if it was guaranteed that different parents in the hierarchy define the context of a <bill>, we can understand its meaning from the context.
  - Searches based on hierarchy context are more complex
  - Such searches are not always practical or guaranteed
- We need a way to embed context into the tag itself
- Fortunately, *namespaces* provide this capability

Namespaces

- The namespace mechanism is a way to package a set of tags
  - Similar to C++ namespaces
  - Similar to Java packages
- All tags within a namespace are well defined and self consistent
- Tags with identical names that are in multiple namespaces have to relationship to each other
  - The relationship can be implied by the name or agreed upon by their creators
  - However, XML parsers treat them as entirely different

Example report.xml Document

```
<?xml version=“1.0”?>
<report>
        <cust:customer id=“1”>
                <cust:lastname>Smith</cust:lastname>
                <cust:firstname>Fred</ cust:firstname>
                <cust:bill>
                    <cust:item id=“2”>
                        <cust:description>sneakers</cust:description>
                        <cust:qty>1</cust:qty> <!-- pairs -->
                        <cust:price>59.99</cust:price> <!-- per item -->
                    </cust:item>
                    <cust:item id=“3”>
                        <cust:description >sweatshirt</cust:description>
                        <cust:qty>3</cust:qty> <!-- pairs -->
                        <cust:price>39.99</cust:price> <!-- per item -->
                    </cust:item>
                </cust:bill>
        </cust:customer>
```

Example (cont.)

```
<cust:customer id="2">
        <cust:lastname>Jones</cust:lastname>
        <cust:firstname>Tom</cust:firstname>
        <cust:bill>
            <cust:item id="1">
                <cust:description>socks</cust:description>
                <cust:qty>6</cust:qty> <!-- pairs -->
                <cust:price>8.49</cust:price> <!-- per item -->
            </cust:item>
        </cust:bill>
</cust:customer>
<purchase:order id="1">
        <purchase:company>Reebok</purchase:company>
        <purchase:date>12/01/01</purchase:date>
        <purchase:bill>
            <purchase:item id="2">
                <purchase:description>sneakers</purchase:description>
```

Example (cont.)

```
                <purchase:style>143-2A88</purchase:style>
                <purchase:qty>25</purchase:qty>  <!-- cases -->
                <purchase:price>1500.00</purchase:price> <!-- per carton -->
            </purchase:item>
            <purchase:item id="3">
                <purchase:description>running shorts</purchase:description>
                <purchase:style>288-4B71</purchase:style>
                <purchase:qty>20</purchase:qty>  <!-- case -->
                <purchase:price>900.00</purchase:price> <!-- per carton -->
            </purchase:item>
        </purchase:bill>
    </purchase:order>
</report>
```

Namespaces in Document

- cust
  - customer
  - lastname
  - firstname
  - **bill**
  - **item**
  - **description**
  - **qty**
  - **price**

- purchase
  - order
  - company
  - date
  - **bill**
  - style
  - **description**
  - **item**
  - **qty**
  - **price**

Explanation of Namespaces

- Namespaces are just a way to augment the name of a tag to make it unique
  - There is no inherent meaning in the name itself
  - We could have chosen xyzzy
  - For that matter, we could have chosen mfxq instead of customer
  - XML just uses the prefixes to distinguish among tag sets
  - XML parsers do the same
- The tag names of each namespace are agreed upon by the set of users associated with that namespace
- There is no requirement that the tags of one namespace have any relationship to the tags of another

Namespace Uniqueness

- How are namespace names guaranteed to be unique?
- What prevents two different domains from choosing the same prefix?
- Are we not back to where we started?
- Fortunately, we have a way to reduce the chance of this
  - Use URI's (Uniform Resource Identifier)
  - Unambiguous Internet domain names
- By associating a URI with a namespace, we can be pretty sure that they will be unique
- Thus, each context chooses a fixed (and familiar) URI for its namespace

URIs

- Two flavors
  - URL (Uniform Resource Locator)
    - Familiar to most people who use the Web
    - Groups of people that are associated with URLs can select those URLs as namespaces
    - Since groups of people do not normally have access to other URLs, the chance that there will be duplications is greatly diminished
    - Example: http://apl.jhu.edu/~davids/653.781/example-1/
  - URN (Uniform Resource Name)
    - Not familiar to most people
    - Groups of people that have associated with URNs can select those URNs as namespaces
    - Example: urn:David-Silberberg-35:example-1

URLs

- Format
  - Protocol (http, gopher, news, ftp, telnet, file, etc.)
  - Followed by "://"
  - Followed by server name (apl.jhu.edu)
  - Followed by directory path on machine as related to the server (/~davids/653.781/example-1/)
- More examples
  - ftp://fred.flintstone.org/pub/downloads/screensaver
  - mailto:wilma.flintsone@fred.flintstone.org
  - C:\My Documents\cartoons\flintstones\FAQ.txt

URNs

- Provides a persistent and location-independent way to reference a resource
  - Resources might move over its life cycle
  - Different URLs may be used during the life cycles of the resources
  - URNs provides persistent names for the resources
- Format
  - Protocol (urn)
  - Followed by ":"
  - Followed by namespace identifier or NID (David-Silberberg)
  - Followed by ":"
  - Followed by namespace specific string or NSS (example-1)

More on URIs

- URIs do not have to point anywhere in particular!
  - Some sites have actual pages for URIs that describe it as a namespace
  - This is not required
- URLs are better for namespaces than URNs
  - There is no mechanism for ensuring URN uniqueness
  - Less chance that anyone reuses a namespace identifier
  - Again, you can put documents at the URL
  - It would be useful to have a data dictionary at this site
    - DTDs and Schemas would also be useful
    - Perhaps, a parser can automatically read this information and process a corresponding XML document

URI Usage in XML Documents

- Let us pick two URIs for our document
  - http://apl.jhu.edu/~davids/cust
  - http://apl.jhu.edu/~davids/purchase
- Let us incorporate these into the report.xml document
  - **What is displayed on the next slide is not the real namespace syntax!**
  - No XML processor can parse the syntax
  - Formulated for illustration purposes

URIs in report.xml – (not valid syntax)

```
<?xml version="1.0"?>
<report>
	<{http://apl.jhu.edu/~davids/cust}customer id="1">
		<{http://apl.jhu.edu/~davids/cust}lastname>Smith
		</{http://apl.jhu.edu/~davids/cust}lastname>
		<{http://apl.jhu.edu/~davids/cust}firstname>Fred
		</ {http://apl.jhu.edu/~davids/cust}firstname>
		<!-- … -->
	</{http://apl.jhu.edu/~davids/cust}customer>
	<{http://apl.jhu.edu/~davids/purchase}order id="1">
		<{http://apl.jhu.edu/~davids/purchase}company>Reebok
		</{http://apl.jhu.edu/~davids/purchase}company>
		<{http://apl.jhu.edu/~davids/purchase}date>12/01/01
		</{http://apl.jhu.edu/~davids/purchase}date>
	<!-- … -->
	</{http://apl.jhu.edu/~davids/purchase}order>
</report>
```

Using Namespaces in XML

- Elements are given qualified names (QNames)
  - First part is namespace prefix, which is ...
  - Separated by a colon from the ...
  - Second part, which is the local name
- Syntax
  - <report xmlns:cust=“http://apl.jhu.edu/~davids/cust”>
  - xmlns is a special namespace which identifies the namespace prefix and namespace URI
  - If the <report …> element were part of the customer namespace then the syntax would be:

    <cust:report xmlns:cust=“http://apl.jhu.edu/~davids/cust”>
  - Multiple namespaces may be defined in a single element

New report.xml Example

```
<?xml version="1.0"?>
<report  xmlns:cust="http://apl.jhu.edu/~davids/cust"
         xmlns:purchase="http://apl.jhu.edu/~davids/purchase" >
    <cust:customer id="1">
        <cust:lastname>Smith</cust:lastname>
        <cust:firstname>Fred</ cust:firstname>
        <!-- … -->
    </cust:customer>
    <purchase:order id="1">
        <purchase:company>Reebok</purchase:company>
        <purchase:date>12/01/01</purchase:date>
    <!-- … -->
    </purchase:order>
</report>
```

Namespace Prefixes Can Vary

- Namespace names can be any string
  - Like variables
  - Need to be self-consistent
- Namespace URIs must be consistent

```
<?xml version="1.0"?>
<report    xmlns:abc="http://apl.jhu.edu/~davids/cust"
           xmlns:xyz="http://apl.jhu.edu/~davids/purchase" >
      <abc:customer id="1">
           <!-- … -->
      </abc:customer>
      <xyz:order id="1">
           <!-- … -->
      </xyz:order>
</report>
```

Default Namespaces

- Default namespaces define the namespaces of elements that do not have a prefix
- For example, the <report> element is not associated with either cust or purchase
  - Let us associate it with a default namespace, say:

    http://apl.jhu.edu/~davids/
  - We include this in the XML document with no local name
- Default namespace can be identical to another defined namespace

Example with Default Namespace

```
<?xml version="1.0"?>
<report xmlns="http://apl.jhu.edu/~davids/"
        xmlns:cust="http://apl.jhu.edu/~davids/cust"
        xmlns:purchase="http://apl.jhu.edu/~davids/purchase" >
    <cust:customer id="1">
        <cust:lastname>Smith</cust:lastname>
        <cust:firstname>Fred</ cust:firstname>
        <!-- … -->
    </cust:customer>
    <purchase:order id="1">
        <purchase:company>Reebok</purchase:company>
        <purchase:date>12/01/01</purchase:date>
    <!-- … -->
    </purchase:order>
</report>
```

Another Example Default Namespace

```
<?xml version="1.0"?>
<report  xmlns="http://apl.jhu.edu/~davids/cust"
         xmlns:purchase="http://apl.jhu.edu/~davids/purchase" >
    <!-- this assumes that the report element is part of the cust namespace -->
    <customer id="1">
        <lastname>Smith</cust:lastname>
        <firstname>Fred</ cust:firstname>
        <!-- … -->
    </customer>
    <purchase:order id="1">
        <purchase:company>Reebok</purchase:company>
        <purchase:date>12/01/01</purchase:date>
    <!-- … -->
    </purchase:order>
</report>
```

Namespaces of Descendents

- Namespaces declared in the root are visible throughout the document
- Namespaces can be declared anywhere in the document
  - These are visible only to the descendents
  - Can be useful if assembling a document from multiple sources
  - Can cancel or override namespaces in descendents

Namespaces of Descendents

- Namespaces declared in the root are visible throughout the document
- Namespaces can be declared anywhere in the document
  - These are visible only to the descendents
  - Can be useful if assembling a document from multiple sources
  - Can cancel or override namespaces in descendents

Namespaces of Descendents

- Namespaces declared in the root are visible throughout the document
- Namespaces can be declared anywhere in the document
  - These are visible only to the descendents
  - Can be useful if assembling a document from multiple sources
  - Can cancel or override namespaces in descendents

Canceling a Default Namespace

- All elements with a namespace or within the scope of a namespace are associated with that namespace
  - Includes default namespaces
- If a search is executed for elements without a namespace, it may not find any if there exists a default namespace
- To cancel the default namespace within a scope, just define the namespace to be blank

Canceling Default Namespace Example

```
<?xml version="1.0"?>
<report   xmlns="http://apl.jhu.edu/~davids/cust">
    <!-- assumes that report is part of the cust namespace -->
    <customer id="1">
            <lastname>Smith</lastname>
            <firstname>Fred</ firstname>
            <!-- … -->
      </customer>
      <order id="1"  xmlns="">
            <company>Reebok</company>
            <date>12/01/01</date>
      <!-- … -->
      </order>
</report>
```

Everything within <order> </order> has no associated namespace.

Namespaces and Attributes

- Namespaces do not function for attributes
- In our original example, id does not belong to any namespace per se
- id is just *associated* with elements that have namespaces
- Attributes can be forced to be associated with a namespace
  - <cust:customer cust:id=“1”>
  - <purchase:order purchase:id=“1”>
- There is no standard regarding attributes, however
  - <cust:customer id=“1”>
  - <cust:customer cust:id=“1”>
  - The standard does not specify whether these are equivalent or not
  - Different applications will treat them either the same or different
- Practically, most applications search for the elements and then process the associated attributes, independent of its namespace

Τέλος Ενότητας