An Introduction to XML and Web Technologies

XML Documents

Anders Møller & Michael I. Schwartzbach
© 2006 Addison-Wesley

Objectives

- What is XML, in particular in relation to HTML
- The XML data model and its textual representation
- The XML Namespace mechanism

What is XML?

- XML: Extensible Markup Language
- A framework for defining markup languages
- Each language is targeted at its own application domain with its own markup tags
- There is a common set of generic tools for processing XML documents
- XHTML: an XML variant of HTML
- Inherently internationalized and platform independent (Unicode)
- Developed by W3C, standardized in 1998

Recipes in XML

- Define our own “Recipe Markup Language”
- Choose markup tags that correspond to concepts in this application domain
  - recipe, ingredient, amount, ...
- No canonical choices
  - granularity of markup?
  - structuring?
  - elements or attributes?
  - ...
Example (1/2)

```xml
<collection>
  <description>Recipes suggested by Jane Dow</description>
  <recipe id="r117">
    <title>Rhubarb Cobbler</title>
    <date>Wed, 14 Jun 95</date>
    <ingredient name="diced rhubarb" amount="2.5" unit="cup"/>
    <ingredient name="sugar" amount="2" unit="tablespoon"/>
    <ingredient name="fairly ripe banana" amount="2"/>
    <ingredient name="cinnamon" amount="0.25" unit="teaspoon"/>
    <ingredient name="nutmeg" amount="1" unit="dash"/>
    <preparation>
      <step>
        Combine all and use as cobbler, pie, or crisp.
      </step>
    </preparation>
  </recipe>
</collection>
```

Example (2/2)

```xml
<comment>
  Rhubarb Cobbler made with bananas as the main sweetener. It was delicious.
</comment>
<nutrition calories="170" fat="28%"
  carbohydrates="58%" protein="14%"/>
<related ref="42">Garden Quiche is also yummy</related>
</recipe>
```

Building on the XML Notation

- Defining the syntax of our recipe language
  - DTD, XML Schema, ...
- Showing recipe documents in browsers
  - XPath, XSLT
- Recipe collections as databases
  - XQuery
- Building a Web-based recipe editor
  - HTTP, Servlets, JSP, ...
- ...

-- the topics of the following weeks...

XML Trees

- Conceptually, an XML document is a tree structure
  - node, edge
  - root, leaf
  - child, parent
  - sibling (ordered), ancestor, descendant

```
A
  B
    C
      D
        E
          F
```

An Analogy: File Systems

Tree View of the XML Recipes

Nodes in XML Trees

- **Text nodes**: carry the actual contents, leaf nodes
- **Element nodes**: define hierarchical logical groupings of contents, each have a *name*
- **Attribute nodes**: unordered, each associated with an element node, has a *name* and a *value*
- **Comment nodes**: ignorable meta-information
- **Processing instructions**: instructions to specific processors, each have a *target* and a *value*
- **Root nodes**: every XML tree has one root node that represents the entire tree

Textual Representation

- **Text nodes**: written as the text they carry
- **Element nodes**: start-end tags
  - `<bla ...>`...
  - `<bla/>`
  - short-hand notation for empty elements: `<bla/>`
- **Attribute nodes**: `name="value"` in start tags
- **Comment nodes**: `<!-- bla -->`
- **Processing instructions**: `<target value?>`
- **Root nodes**: implicit
Browsing XML (without XSLT)

More Constructs

- XML declaration
- Character references
- CDATA sections
- Document type declarations and entity references explained later...

Whitespace?

Well-formedness

- Every XML document must be well-formed
  - start and end tags must match and nest properly
  - exactly one root element
  - in other words, it defines a proper tree structure

XML parser: given the textual XML document, constructs its tree representation
Simpler Alternatives?

S-expressions, 1958:

```
(collection
  (recipe
    (title "Rhubarb Cobbler") (date "Wed, 14 Jun 95")
    ...
  )
)
```

- XML is defined as a simplified subset of SGML
- XML could have been designed simpler...
- ... but it wasn’t [end of discussion]

Applications

Rough classification:
- Data-oriented languages
- Document-oriented languages
- Protocols and programming languages
- Hybrids

Example: XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>Hello world!</title></head>
  <body>
    <h1>This is a heading</h1>
    This is some text.
  </body>
</html>
```

Example: CML

```
<molecule id="METHANOL">
  <atomArray>
    <stringArray builtin="id">a1 a2 a3 a4 a5 a6</stringArray>
    <stringArray builtin="elementType">C O H H H H</stringArray>
    <floatArray builtin="x3" units="pm">-0.748 0.558 ...
    </floatArray>
    <floatArray builtin="y3" units="pm">-0.015 0.420 ...
    </floatArray>
    <floatArray builtin="z3" units="pm">0.024 -0.278 ...
    </floatArray>
  </atomArray>
</molecule>
```
Example: ebXML

```xml
<MultiPartyCollaboration name="DropShip">
  <BusinessPartnerRole name="Customer">
    <Performs initiatingRole='//binaryCollaboration[@name="Firm Order"]/InitiatingRole[@name="buyer"]'/>
  </BusinessPartnerRole>
  <BusinessPartnerRole name="Retailer">
    <Performs respondingRole='//binaryCollaboration[@name="Firm Order"]/RespondingRole[@name="seller"]'/>
    <Performs initiatingRole='//binaryCollaboration[...]/InitiatingRole[@name="buyer"]'/>
  </BusinessPartnerRole>
  <BusinessPartnerRole name="DropShip Vendor">
    ...
  </BusinessPartnerRole>
</MultiPartyCollaboration>
```

Example: ThML

```html
<h3 class="s05" id="One.2.p0.2">Having a Humble Opinion of Self</h3>
<p class="First" id="One.2.p0.3">EVERY man naturally desires knowledge; but what good is knowledge without fear of God? Indeed a humble rustic who serves God is better than a proud intellectual who neglects his soul to study the course of the stars.
</p>
</h3>
```

XML Namespaces

- Assign a URI to every (sub-)language:
  - e.g. `http://www.w3.org/1999/xhtml` for XHTML 1.0

- Qualify element names with URIs:
  ```html
  {http://www.w3.org/1999/xhtml}head
  ```

The Idea

- When combining languages, element names may become ambiguous!
- Common problems call for common solutions
An Introduction to XML and Web Technologies

The Actual Solution

- **Namespace declarations** bind URIs to prefixes

  ```xml
  <... xmlns:foo="http://www.w3.org/TR/xhtml1">
  ...<foo:head>...</foo:head>
  ...
  </...>
  ```

- Lexical scope
- Default namespace (no prefix) declared with

  ```xml
  xmlns="http://www.w3.org/
  ```

- Attribute names can also be prefixed

Widgets with Namespaces

- **Namespace map**: for each element, maps prefixes to URIs

```xml
<widget type="gadget" xmlns="http://www.widget.inc">
<head size="medium"/>
<big><subwidget ref="gizmo"/></big>
<info xmlns:xhtml="http://www.w3.org/TR/xhtml1">
<xhtml:head>
<xhtml:title>Description of gadget</xhtml:title>
</xhtml:head>
<xhtml:body><xhtml:h1>Gadget</xhtml:h1>
A gadget contains a big gizmo
</xhtml:body>
</info>
</widget>
```

Summary

- **XML**: a notation for hierarchically structured text
- Conceptual tree model vs. concrete textual representation
- Well-formedness
- Namespaces

Essential Online Resources

- [http://www.w3.org/TR/xml11/](http://www.w3.org/TR/xml11/)
- [http://www.w3.org/TR/xml-names11](http://www.w3.org/TR/xml-names11)
- [http://www.unicode.org/](http://www.unicode.org/)