Information Systems

WS 2005, JKU Linz
Course 9: XML

Gábor Bodnár

URL: `http://www.risc.uni-linz.ac.at/education/courses/ws2005/is/`

Overview

- XML syntax definitions.
- Universal Resource Identifiers.
- Namespaces.

What is XML?

Extensible Markup Language (XML) is a globally accepted, vendor independent standard for representing text-based data.

The organization behind XML and many other web related technologies is the World Wide Web Consortium (W3C).

Goals of XML:

Simplicity XML documents should be strictly and simply structured.

Compatibility XML is platform independent. It should be easy to write or update applications that make use of XML.

Legibility XML documents should be human readable.

Applications of XML

Applications of XML include (but not limited to)

Data storage Providing a standard way for storing text data in a hierarchically structured way.

Data exchange Serving as a common language to which text based data can be transformed from various formats.

Templating Creating template documents that describe various fields and attributes, for instance in case of business forms.

HTML translation Data stored in XML format can easily be transformed into HTML for Web publication.

How Does an XML Document Look Like

```
<?xml version="1.0" encoding="UTF-8"?>
<folder>
  <email date='20 Aug 2003'>
    <from>robert@company.com</from>
    <to>oliver@company.com</to>
    <subject>Meeting</subject>
    Could we meet this week to discuss the
    interface problem in the NTL project? --Rob
  </email>
</folder>
```

The structure is described by the *markup* (the text marked by <,>).

Generic Rules of XML Syntax

The text of the XML document consists of

- The text data which is being represented: *character data*.
- The text of the *markup* (enclosed by $<$,$>$).

The markup consists of *tags* (e.g. the `<to>`,`</to>` pair).

The part of the document enclosed by a tag is an *element*.

The outermost tag encloses the *root element*.

An XML document must have exactly one root element and the nesting of elements must be a proper one.

Elements

Elements are the primary structuring units of XML documents.

An element is delimited by its start and end tags.

The content of elements can be

element if the element contains only elements (e.g. `folder` in the example above),

character if it contains only character data (e.g. `to`),

mixed if it contains both (e.g. `email`),

empty if it contains nothing (e.g. `<x></x>`).

Elements: Children and Parents

Relationships between the elements:

Child element An element inside another one in the first nesting level.

Parent element It is the reverse of the child relationship.

Sibling element These are elements with the same parent.

```
<email date='20 Aug 2003'>
  <from>robert@company.com</from>
  <to>oliver@company.com</to>
  <subject>Meeting</subject>
</email>
```

Elements: Descendants and Ancestors

Descendant element It is an element in the transitive closure of the child relationship.

Ancestor element It is an element in the transitive closure of the parent relationship.

```
<folder>
  <email date='20 Aug 2003'>
    <from>robert@company.com</from>
    <to>oliver@company.com</to>
    <subject>Meeting</subject>
  </email>
</folder>
```

Naming Conventions

Names for elements can be chosen according to the following rules.

- Names are taken case sensitive.
- Names cannot contain spaces.
- Names starting with "xml" (in any case combination) are reserved for standardization.
- Names can only start with letters or with the '_', ':' characters.
- They can contain alphanumeric characters and the '_', '-', ':', '.' characters.

Attributes

Attributes are `name='value'` pairs, listed in the start-tags of elements.

```
<email date='20 Aug 2003'> ... </email>
```

- The naming rules of elements apply also for attributes.
- Elements can contain zero or more attributes.
- The names of the attributes must be unique within a start-tag.
- Attributes cannot appear in end-tags.
- Attribute values must be enclosed in single or double quotes.

Elements vs Attributes

Attributes can be resolved into elements and elements with character content can be put into attributes.

```
<email>
  <date>20 Aug 2003</date>
  <from>robert@company.com</from>
  ...
</email>

<email date='21 Aug 2003' from='oliver@company.com'
  to='rob@company.com' cc='amy@company.com'>
  <subject>Re: Meeting</subject>
  ...
</email>
```

Additional XML Syntax

Special characters have to be substituted with the corresponding entity references.

Character	Entity reference
<	`<`
>	`>`
"	`&quot;`
'	`&apos;`
&	`&`

Comments can be included in the XML document anywhere outside other markup with the following syntax.

```
<!-- Comment text comes here. -->
```

Summary on XML Syntax

A well-formed XML document must fulfill the requirements:

- XML elements are marked with start- and end-tags whose names must match.
- There can be only one root element.
- The elements must be properly nested into each other.
- Elements can have attributes to store additional character data.
- We have to obey certain naming rules for elements and attributes.
- Special characters must be replaced by entity references.

Example: XHTML

http://www.w3.org/TR/xhtml1/ http://www.w3.org/TR/html4/

The overall structure:

- The prolog: XML and document type declarations

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- The root element must be `html`:

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

Example: XHTML (Header)

Inside the root:

- The header can contain the title, metadata about the document, links to other resources (e.g. stylesheets, script libraries):

```
<head>
  <meta name="Author" content="Gabor Bodnar"/>
  <link rel="stylesheet" type="text/css" href="my.css"/>
  <title>Course Records WS2003</title>
</head>
```

- The main part:

```
<body> ... </body>
```

Example: XHTML (Text)

Structured text:

`h1, ..., h6` headings in decreasing sizes,

`em, strong, cite, code` emphasis, strong emphasis, citations, program codes,

`p, br, pre` paragraph, line breaking (empty element), preformatted text.

```
<h2>Status of this document</h2>
<p>This document is the second edition of the
  <strong>XHTML 1.0</strong> specification incorporating
  the errata changes as of<br/><em>1 August 2002.</em></p>
```

Example: XHTML (Lists)

Lists: `ul`, `ol`: unordered/ordered lists, `li` list elements.
`dl`, `dt`, `dd`: definition list/definition term/definition description.

```
<ul>
 <li> level one </li>
 <ol>
  <li> level two.A </li>
  <li> level two.B </li>
 </ol>
</ul>
```

- level one
  1. level two.A
  2. level two.B

```
<dl>
  <dt>Hacker</dt>
  <dd>a clever programmer</dd>
  <dt>Nerd</dt>
  <dd>technically bright but
    socially inept person</dd>
</dl>
```

Hacker
: a clever programmer

Nerd
: technically bright but socially inept person

Example: XHTML (Tables)

Tag names for tables: `table`, `tr`, `th`, `td`, `caption`.

Some of the applicable attributes:

`table` : `border`, `width`, `cellspacing`, `cellpadding`, `align`, `bgcolor`, `id`, `class`,

`th,td` : `rowspan`, `colspan`, `align`, `valign`, `width`, `height`, `bgcolor`, `id`, `class`, `nowrap`.

Example: XHTML (Tables)

The HTML code:

```
<table border="1" align="center">
<tr><th colspan="3">Personal data</th></tr>
<tr><th>First name</th><th>Last name</th><th>Age</th></tr>
<tr><td>John</td>      <td>White</td>    <td>39</td></tr>
</table>
```

results:

Personal data		
First name	**Last name**	**Age**
John	White	39

Example: XHTML (Links)

The HTML document is identified by its URL. Within a document one can define anchors:

```
...
<h2 id="section1">Section 1</h2>
<p><a name="section1intro">Introduction</a><br/>
...
```

Referencing anchors and other resources is done as

```
In the previous <a href="#section1">section</a> ...
More information can be found
<a href="http://www.w3.org/">here</a>.
```

Uniform Resource Identifiers (URI)

URI are character sequences with restricted syntax to reference things with identity.

`mailto:gbodnar@risc.uni-linz.ac.at`
`http://www.risc.uni-linz.ac.at/education/courses/`

Absolute URIs refer to the resource independent of the context in which the identifier is used.

Relative URIs describe the difference between the current context and the absolute identifier of the resource in a hierarchical namespace.

Base URI

A relative URI requires to determine an absolute URI, called *base URI*, which determines the context for it.

- The base URI may be embedded into the document.
- If not in the previous case, the base URI might be determined by the retrieval context of the entity (resource) of one encapsulation level higher.
- If not in the previous cases, i.e. the resource is not encapsulated into another entity, the last URI used to retrieve the document which contains the current URI is used as a base URI.
- If none of the above conditions apply, the base URI is defined by the context of the application.

Structure of URIs

The *scheme* component defines the semantics of the rest of the URI. For Example: `http` or `mailto`.

The *authority* component defines the top hierarchical element in the namespace. Example: `www.netbanking.at:443`.

The *path* component identifies the resource within the context of the scheme and authority. The hierarchical structure is indicated by '/' signs. Example: `/education/courses`.

The *query* component is a string to be interpreted by the resource. It is separated from the path component by a '?'.

`http://dict.leo.org/?search=namespace&lang=en`

Namespaces

Namespaces provide a standard way to resolve naming conflicts.

The `name` tag in this example marks entities of different categories.

```
<name>Springer</name>
<name>Donald E. Knuth</name>
```

A namespace defines a character sequence to be prepended for the class of names belonging to it.

Namespaces in XML

To ensure uniqueness of namespaces, usually they are identified by URIs (e.g. `http://big.company.com/product-info`).

However they need not correspond to existing URLs!

Namespaces can be declared within any element of an XML document, using the reserved attribute name `xmlns`.

```
<list xmlns:p="http://big.company.com/product-info">
  <p:product>
    <p:name>Disney Mouse Pad</p:name>
    <p:id>231069948</p:id>
    <p:price>3.49 EUR</p:price>
  </p:product>
</list>
```

Qualifying Names

Whenever a prefix is given in a namespace declaration, all descendants of the element which belong to the namespace must be named with the prefix prepended and separated by a colon from the tag name (e.g. `p:id`).

This is also called *qualifying* the name (e.g. `id` is qualified to be in the namespace with alias `p`) and the name with a prefix (`p:id`) is called a *qualified name*.

If no prefix is given, so that the attribute is just `xmlns`, in a namespace declaration, it will declare the *default namespace* for the element. In all descendants of the element unqualified tag names will automatically fall into the default namespace.

Namespace Scope

The section of the XML document to which the namespace declaration applies is called its *scope*.

```
<catalog>
  <b:book xmlns:b="http://www.mybookstore.com/catalog"
      isbn="0-321-19784-4">
    <b:author>C. J. Date</b:author>
    <b:title>An Introduction to Database Systems</b:title>
  </b:book>
  <b:book isbn="0-8129-9191-5">
    <b:author>G. Brill</b:author>
    <b:title>Codenotes for XML</b:title>
  </b:book>
</catalog>
```

Subtle Differences

```
<list xmlns:p="http://big.company.com/product-info">
  <p:product> ... </p:product>
</list>

<list xmlns="http://big.company.com/product-info">
  <product> ... </product>
</list>

<p:list xmlns:p="http://big.company.com/product-info">
  <p:product> ... </p:product>
</p:list>
```