Internet Fundamentals & Introduction to Web Technologies

Course: IT (044615)

Lecture: 9

XML

Dr. Ramez Hajislam

Introduction

- eXtensible Markup Language
- Developed from SGML
- A *meta-markup* language
- Deficiencies of HTML and SGML
  - Lax syntactical rules
  - Many complex features that are rarely used
- HTML is a markup language, XML is used to define markup languages
- Markup languages defined in XML are known as *applications*
- XML can be written by hand or generated by computer
  - Useful for data exchange

The Syntax of XML

- Levels of syntax
  - *Well-formed documents* conform to basic XML rules
  - *Valid documents* are well-formed and also conform to a *schema* which defines details of the allowed content

The Syntax of XML

- Well-formed XML documents
  - All begin tags have a matching end tag
    - Empty tags
  - If a begin tag is inside an element, the matching end tag is also
  - There is one *root* tag that contains all the other tags in a document
  - Attributes must have a value assigned, the value must be quoted
  - The characters <, >, & can only appear with their special meaning
  - http://www.w3.org/TR/2006/REC-xml-20060816/#sec-well-formed is the official definition
- Validity is tested against a schema, discussed later

XML Document Structure

- Auxiliary files
  - Schema file
    - DTD or XML Schema or one of several other
  - Style file
    - Cascading Style Sheets
    - XSLT
- Breaking file up
  - Document entities
  - Entity syntax
- Character data
  - <![CDATA …..]]>

Document Type Definitions

- A set of *declarations*
- Define tags, attributes, entities
- Specify the order and nesting of tags
- Specify which attributes can be used with which tags
- General syntax
  - <!keyword …. >
  - Note, not XML!

Declaring Elements

- General syntax
  - <!ELEMENT *element-name content-description*)>
  - Content description specifies what tags may appear inside the named element and whether there may be any plain text in the content
- Sequence of tags
- Alternate tags
- Multiplicity
  - +
  - *
  - ?
- #PCDATA

Declaring Attributes

- General syntax
  - <!ATTLIST *element-name*
    (attribute-name attribute-type default-value?)+ >
- Default values
  - A value
  - #FIXED value
  - #REQUIRED
  - #IMPLIED (default, if not specified)

Declaring Entities

- General Syntax
  - <!ENTITY [%] *entity-name “entity-value”*>
  - With %: a parameter entity
  - Without %: a general entity
- Parameter entities may only be referenced in the DTD
- Remote form
  - <!ENTITY *entity-name* SYSTEM *“file-location”*>
  - The replacement for the entity is the content of the file

Sample DTD

Internal and External DTDs

- A document type declaration can either contain declarations directly or can refer to another file
- Internal
  - <!DOCTYPE *root-element* [

        declarations

    ]>
- External file
  - <!DOCTYPE *root-name* SYSTEM *“file-name”*>
- A public identifier can also be specified, that would be mapped to a system identifier by the processing system

Namespaces

- “XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references.”
  - From the specification

    http://www.w3.org/TR/2006/REC-xml-names-20060816/
- A namespace can be declared for an element and its descendants by
  - <*element xmlns*[*:prefix*]=*“URI”*>
  - The prefix is used to qualify elements that belong to the namespace
  - Multiple namespaces can be used in a single document
  - Default namespace
- DTDs do not support namespaces very well

XML Schemas

- Schema is a generic term for any description of an XML content model
- DTDs have several deficits
  - They do not use XML syntax
  - They do not support namespaces
  - Data types cannot be strictly specified
    - Example date vs. string

Schema Fundamentals

- Documents that conform to a schema’s rules are considered *instances* of that schema
- Schema purposes
  - Structure of instances
  - Data types of elements and attributes
- XML Schemas support namespaces
  - The XML Schema language itself is a set of XML tags
  - The application being described is another set of tags

Defining a Schema

- The root of an XML Schema document is the schema tag
- Attributes
  - `xmlns` attributes for the schema namespace and for the namespace being defined
  - A `targetNamespace` attribute declaring the namespace being defined
  - An `elementFormDefault` attribute with the value qualified to indicate that all elements defined in the target namespace must be namespace qualified (either with a prefix or default) when used

Defining a Schema Instance

- The xmlns attribute declares a namespace for an element and its descendants
  - <element xmlns[:prefix]="URI">
  - The element itself may not be in the namespace
  - Multiple elements may be defined
- The http://www.w3.org/2001/XMLSchema-instance namespace includes one attribute, schema Location
  - That attribute value is pairs, separated by spaces
  - Each pair consists of a namespace and the location of a file that defines that namespace

An Overview of Data Types

- Data types are of two kinds
  - Simple data types with string content
  - Complex data types with elements, attributes and string content
- Predefined types
  - Primitive
  - Derived
- Restrictions
  - Facets
- Anonymous and named types

Simple Types

- Named types can be used to give the type of
  - an attribute (which must be simple) or
  - an element (which may be simple or complex)
- Elements or attributes with simple type may have default values specified
- New simple types can be defined by restriction of base types
  - Facet maxLength
  - Facet precision

Complex Types

- Definition of a complex type can specify
  - Elements in content (either sequence or choice)
    - Individual elements may specify a multiplicity
  - Attributes that can appear for an element of that type
  - Whether plain text is allowed in the content, a *mixed* type
- An element definition can be associated with a type by
  - Referring to a named type directly in the type attribute
  - Including an anonymous type definition

Validating Instances of Schemas

- Various systems for validating instances against schemas
  - Online http://www.w3.org/2001/03/webdata/xsv
  - XML support libraries include validation: Xerces from Apache, Saxon, Altova XML tools
  - Some IDE’s have automatic validation: Altova Spy, Eclipse with Oxygen, Eclipse with XML Buddy Pro
- Certain IDE’s will use schemas to provide support for XML file creation

Displaying Raw XML Documents

- Plain XML documents are generally displayed literally by browsers
  - Firefox notes that there is no style information

Displaying XML Documents with CSS

- An xml-stylesheet processing instruction can be used to associate a general XML document with a style sheet
  - `<?xml-stylesheet type="text/css" href="planes.css">`
- The style sheet selectors will specify tags that appear in a particular document

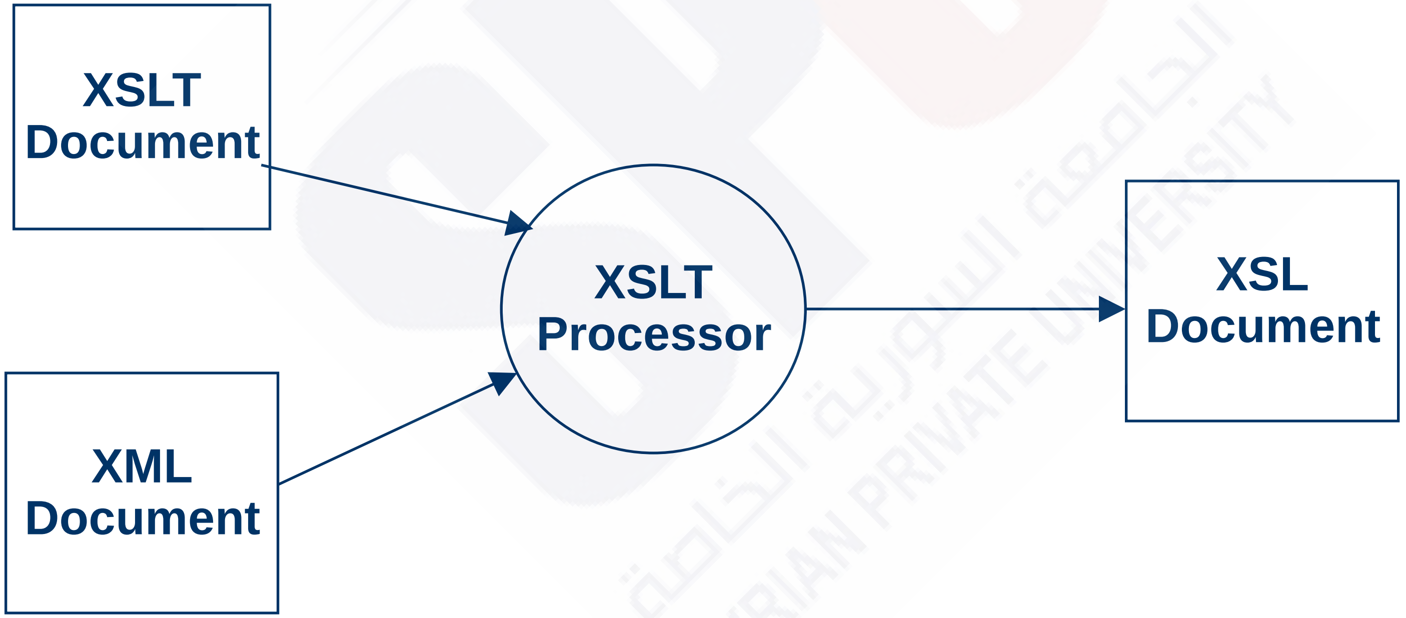
XSLT Style Sheets

- A family of specifications for transforming XML documents
  - XSLT: specifies how to transform documents
  - XPath: specifies how to select parts of a document and compute values
  - XSL-FO: specifies a target XML language describing the printed page
- XSLT describes how to transform XML documents into other XML documents such as XHTML
  - XSLT can be used to transform to non-XML documents as well

Overview of XSLT

- A functional style programming language
- Basic syntax is XML
  - There is some similarity to LISP and Scheme
- An XSLT processor takes an XML document as input and produces output based on the specifications of an XSLT document

XSLT Processing



XSLT Structure

- An XSLT document contains templates
- XPath is used to specify patterns of elements to which the templates should apply
- The content of a template specifies how the matched element should be processed
- The XSLT processor will look for parts of the input document that match a template and apply the content of the template when a match is found
- Two models
  - Template-driven works with highly regular data
  - Data-driven works with more loosely structured data with a recursive structure (like XHTML documents)

XSL Transformations for Presentation

- One of the most common applications of XSLT is to transform an XML document into an XHTML document for display
- A XSLT style sheet can be associated with an XML document by using a processor instruction
- <?xml-stylesheet type="text/xsl" href="*stylesheet-ref*"?>
- The example xslplane.xml is an xml file with data about a single plane
  - The file is linkded to the stylesheet xslplane.xsl

XSLT Organization

- Root element stylesheet
  - Specifies namespaces for XSL and for non-XSLT elements included in the stylesheet

```
<xsl:stylesheet xmlns:xsl =
   "http://www.w3.org/1999/XSL/Format"
   xmlns =
"http://www.w3.org/1999/xhtml">
```

- Elements in XSLT itself will have the prefix `xsl:`
- Elements from XHTML will have no prefix (default namespace)

XSLT Templates

- There must be at least one template element in an style sheet
- The value of the `match` attribute is an XPath expression which specifies to which nodes the template applies
- Two standard choices for the `match` expression of the first template
  - '/' to match the root node of the entire document structure
  - '*root-tag*' to match the root element of the document
- The first template is applied automatically
- All other templates are applied only in response to `apply-template` elements

XPath Basics and Node Selection

- An XPath expression beginning with a / specifies nodes in an absolute position relative to the document root node
- Otherwise, the expression specifies nodes relative to the *current node*, that is the node being processed before the matched node
- The expression ‘.’ refers to the current node
- The apply-templates tag uses the select attribute to choose which nodes should be matched to templates
- There is a default template applied if one is not provided that matches a selected node

Producing Transformation Output

- Elements not belonging to XSLT and other text will be copied to the output when the containing template is applied
- The value-of tag causes the select attribute value to be evaluated and the result is put into the output
  - The value of an element is the text contained in it and in sub-elements
  - The value of an attribute is the value
- Example xslplane1.xsl transforms the xslplane.xml file into XHTML for display purposes
  - If the style sheet is in the same directory as the XML file, some browsers will pick up the transformation and apply it
  - This works with Firefox and Internet Explorer but not Opera

Processing Repeated Elements

- File xslplanes.xml contains data about multiple airplanes
- The style sheet xslplanes.xsl uses a for-each element to process each plane element in the source document
- A sort element could be included to sort output
  - The element

    ```
    <xsl:sort select="year" data-type="number"/>
    ```
  - Specifies sorting by year

XML Processors

- XML processors provide tools in programming languages to read in XML documents, manipulate them and to write them out

Purposes of XML Processors

- Four purposes
  - Check the basic syntax of the input document
  - Replace entities
  - Insert default values specified by schemas or DTD’s
  - If the parser is able and it is requested, validate the input document against the specified schemas or DTD’s
- The basic structure of XML is simple and repetitive, so providing library support is reasonable

Purposes of XML Processors

- Examples
  - Xerces-J from the Apache foundation provides library support for Java
  - Command line utilities are provided for checking well-formedness and validity
- Two different standards/models for processing
  - SAX
  - DOM

Parsing

- The process of reading in a document and analyzing its structure is called *parsing*
- The parser provides as output a structured view of the input document

The SAX Approach

- In the SAX approach, an XML document is read in serially
- As certain conditions, called events, are recognized, event handlers are called
- The program using this approach only sees part of the document at a time

The DOM Approach

- In the DOM approach, the parser produces an in-memory representation of the input document
  - Because of the well-formedness rules of XML, the structure is a tree
- Advantages over SAX
  - Parts of the document can be accessed more than once
  - The document can be restructured
  - Access can be made to any part of the document at any time
  - Processing is delayed until the entire document is checked for proper structure and, perhaps, validity
- One major disadvantage is that a very large document may not fit in memory entirely

Web Services

- Allow interoperation of software components on different systems written in different languages
- Servers that provide software services rather than documents
- Remote Procedure Call
  - DCOM and CORBA provide impllementations
  - DCOM is Microsoft specific
  - CORBA is cross-platrom

Web Service Protocols

- Three roles in web services
  - Service providers
  - Service requestors
  - Service registry
- The Web Services Definition Language provides a standard way to describe services
- The Universal Description, Discovery and Integration service provides a standard way to provide information about services in response to a query
- SOAP is used to specify requests and responses