XML Tutorial

Andy Scholand
andrew.scholand@cad.gatech.edu
2/29/00

Motivation - Why XML?

- Industry Focus (Resume Fodder!)
- HTML is broken

Microsoft Goals for XML

- Incubate/Integrate/Innovate
- Make XML mainstream
- Deeply integrate XML with platform
- Enable applications
- Create opportunity
- Evolve in steps

"XML is a breakthrough technology"

Bill Gates,
Oct 97, Seybold

Motivation

HTML Evolution

- Started with very few tags ...
- Language evolved, as more tags were added
  - forms
  - tables
  - fonts
  - frames

HTML Problems

- Desire for personalized tags
- Want to put data into HTML form
  - mathematics, database entries, literary text, poems, purchase orders ....
- HTML just isn't designed for that!
Goals for XML

- Better search results
- Presenting various views of same data
- Integration of data from different sources
- Easy use over the Internet
- Easy development of applications that process documents
- Documents readable by humans
- Documents easy to create
- Interchange of data

XML Background

- Where does XML come from?
- What is it (in general)
- Who specifies it?

Idea: Back to the Basics

- HTML was defined using SGML
  - Standard Generalized Markup Language
  - A meta-language for defining languages.
- Complex, sophisticated, powerful
- Idea: Use SGML

Problems with SGML

- Too complicated a language
- Rules are too strict
- Not good in a distributed environment
- Can't mix different data together

Idea (2): "Webified" SGML

- New eXtensible Markup Language: XML
- Can use XML to define new languages
- Distributes easily on the Web
- Can mix different types of data together

Basic XML Rules

- Tags like in HTML, but ...
- Technical details
  - Always need end tags
  - Special empty-element tags
  - Always quote attribute values
Just what is XML?

- It's a markup language used for annotating text
- It is concerned with logical structure. It identifies sections, titles, section headers, chapters, paragraphs, etc.
- It is not concerned with appearance. You say 'this is a subtitle' not 'this is in bold, 14pt, centered'
- You say 'this is an example' not 'this is in verbatim, indented by 5pts.'

Like this example .....

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<xml xmlns="http://www.w3.org/TR/xhtml1">
<head>
<title>Title of text XHTML Document</title>
</head>
<body>
<div class="myDiv">
<h1>Heading of Page</h1>
<p>And here is another paragraph, this one containing an <img src="image.gif" alt="waste of time" /></p>
</div>
</body></html>
```

Who Specifies XML?

- Extensible Markup Language
- A text-based, data description meta language
- Design your own markup language
- A streamlined subset of SGML
- Designed for use on the Internet
- A W3C Technical Recommendation (February 10, 1998)

The W3C

- The W3C is the World Wide Web Consortium, a voluntary association of companies and nonprofit organizations. Membership costs serious money, confers voting rights. Complex procedures, with the Chairman (Tim Berners-Lee) holding all the high cards, but the big vendors (e.g. Microsoft, Adobe, Netscape) have a lot of power.

XML in Depth

- What is an XML Document?
- What is it - syntax

Documents

- Well-formed document: obeys the syntax of XML
- Valid documents: a well-formed document that contains a proper document type declaration and obeys the constraints of that declaration
Structure of XML Document

- Prolog
  - XML declaration - information about document XML version and encoding
    Example:
    ```xml
    <?xml version="1.0"?>
    ```
  - Document Type Declaration (DTD)
    - internal - embedded in the document
    - external - pointer to the file with defined grammar

- Body - structured data with one root element

XML Markup

XML markup specifies the structure of the document. All text that is not markup is the character data of the document.

- Comments
  - Comments make the structure of the document clearer
  - Can appear anywhere in a document
  - Comments are not part of the document data (content of comments may be ignored by XML parsers)
  - Example:
    ```xml
    <name>
    <!--This is a short comment-->
    Smith
    </name>
    ```

Entity References

- Entity is a term that represents certain data
  - XML parser will substitute that data for the entity
  - Entities can be used to store binary data
  - Predefined entities: amp, lt, gt, apos, quot that stand for: &, <, >, ' ', "
  - Example:

Character References

- Character reference is a character in the ISO 10646 character set, usually not directly accessible from available input devices
- Character reference is specified as a hexadecimal or decimal code for a character
- Example:
  ```xml
  &#x000d; is a carriage return
  ```

Processing Instructions

- Processing instructions are not part of the document's data but must be passed through to the application
  - By-passing the XML processor and delivering instructions directly to a process
  - PI begins with the target application identifier
    ```xml
    <?xml version="1.0"?>
    <?xml-stylesheet type="text/xsl"
    href="mystyle.xsl"?>
    ```
**CDATA Sections**

- CDATA section can be used to store marked-up text so that the markup is not evaluated.
- CDATA sections are useful if the user wants to store XML markup as a data.
- Example:
  
  ```xml
  <buffer>
  <![CDATA[<price>50</price>]]>
  </buffer>
  ```

**Elements - Start & End Tags**

- Format: start tag, end tag, and data.
- The tags describe the data contained between tags.
- The data within the tags is the value of the element.

  ```xml
  <title>The Fish Slapping Dance</title>
  <dance>The Fish Slapping Dance</dance>
  ```

**Attributes**

- Attributes associate name-value pairs with elements.
- Attributes may appear only within start tags and empty-element tags.

  ```xml
  <size unit="KB">629</size>
  ```

**Empty Elements**

- Empty element tag has special form.
- Represent elements that have no content.
- Example:

  ```xml
  <img align="left"
  source="picture.jpg" />
  ```

**Example of XML Document**

```xml
<xml version="1.0" ?
<!DOCTYPE doc SYSTEM "pubgrammar.dtd">
<doc>
  <publication number="pdf">  
  <title>Collaborative Virtual Workspace</title>
  <author>
    <lastname>Spelman</lastname>
    <firstname>Tom</firstname>
  </author>
  <date>1997</date>
  <keywords>
    <keyword>collaboration framework</keyword>
    <keyword>virtual environments</keyword>
  </keywords>
</publication>
</doc>
```

**Documents - Example**

- The example shows plain XML document displayed in Internet Explorer 5.0.
- URL:
  
The XML Success Story

- XML has been a runaway success, on a much greater scale than its designers anticipated
  - Not for the reason they had hoped
  - But for a reason they barely thought about
  - The emphasis in the XML design was on simplifying SGML to move it to the Web
- Tree structured documents are a useable transfer syntax for just about anything
- So data-oriented web users think of XML as a transfer mechanism for their data

XML is ASCII for the 21st century

- ASCII (ISO 646) solved a fundamental interchange problem
  - What bits encode what characters
- UNICODE/ISO 10646 extends to world-wide
- XML thought it was doing the same for simple tree-structured documents
  - The emphasis in the XML design was on simplifying SGML to move it to the Web
- XML didn't touch SGML's architectural vision
  - Flexible linearization/transfer syntax
  - For tree-structured documents with internal links

Take Two: Just what is XML?

- It's a markup language used for transferring data
- It is concerned with data models
  - To convert between application-appropriate and transfer-appropriate forms
- It is not concerned with human beings
  - It's produced and consumed by programs

Structured Markup

```xml
<store>
  <part id="p001">
    <part-name>window</part-name>
    <price>45</price>
    <instock>yes</instock>
  </part>
  <part id="p002">
    <part-name>mouse</part-name>
    <price>150</price>
    <instock>yes</instock>
  </part>
</store>
```

Application Data

The Cambridge Communiqué

- A W3C Note resulting from a meeting this August (http://www.w3.org/TR/schema-arch)
- Signalled a widespread acceptance of XML as a data layer:
  "XML has defined a transfer syntax for tree-structured documents;
  "Many data-oriented applications are being defined which build their own data structures on top of an XML document layer, effectively using XML documents as a transfer mechanism for structured data;"

```xml
<store>
  <part id="p001">
    <part-name>window</part-name>
    <price>45</price>
    <instock>yes</instock>
  </part>
  <part id="p002">
    <part-name>mouse</part-name>
    <price>150</price>
    <instock>yes</instock>
  </part>
</store>
```
The Communique, cont'd

- Called for support in XML Schema for specifying mapping between the XML document data model (or XML Infoset) and application-specific data models
- XML Schema is a W3C recommendation-in-progress for defining the structure of document families
- A grammar for markup structure
  - artice -> title, subtitle?, section+
  - POORDERHDR -> DATETIME, ORDERSMRT

Other XML Basic Concepts

- Content Descriptions
  - DTDs
  - Namespaces
  - Content Descriptions
  - Schemas
  - Document Object Model
  - Data Islands

Content Description

- Documents have structure
  - Document types
  - Document instances
- Structure can be defined
  - Informally
  - SGML DTD
  - XML DTD
  - Schema using XML

Document Grammar Specifications

Document Type Definition (DTD)

- Provides formal definition of:
  - tags used in XML document
  - order of the tags
  - containment relationships between tags
  - types of data contained in the elements
- Used for:
  - XML document validation
  - Describing grammar for other users

More about a DTD

- Controls the manipulation of data
  - Requires everyone to use the same set of tags the same way
  - Association with XML document:
    (External Reference)
    <!DOCTYPE clip SYSTEM "clipdef.dtd">

Document Type Definition - Example

```xml
<!DOCTYPE clip [
<!ELEMENT clip (title, size)>]
<!ELEMENT title (#PCDATA)>]
<!ELEMENT size (#PCDATA)>]
<!ATTLIST size unit CDATA #REQUIRED> ]
```
Internal vs. External DTD

- External DTD - usual case
  ```xml
  <!DOCTYPE greeting SYSTEM "hello.dtd">
  <greeting>Hello</greeting>
  ```

- Internal DTD
  ```xml
  <!DOCTYPE greeting [
    <!ELEMENT greeting (#PCDATA)>
  ]>
  <greeting>Hello</greeting>
  ```

DTD Notation Used for Element Content Declarations

- (expression) - expression treated as a unit
- (a, b) - sequence: a followed by b
- (a|b) - choice: a or b but not both
- a? - a or nothing
- a+ - one or more occurrences of a
- a* - zero or more occurrences of a

Example:
```
(title,author+,date,keywords?)`
```

Problem Namespaces Addresses

- Sometimes XML elements have the same name but mean different things
  ```xml
  <title>The Fish Slapping Dance</title>
  <title>Mr</title>
  ```

- Namespaces are a mechanism for solving elements and attributes name conflicts

Namespaces

- Collection of related XML elements and attributes identified by a URI reference
- NOTE: URIs are used to avoid collisions in namespace's names
- Provides unique names for elements and attributes by adding context to the tags
- Enables reuse of grammar specifications

XML Namespaces

- Makes XML truly extensible
- Enable developers to mix data described by multiple schema in one XML document instance
- Schema components are reusable

  ```xml
  <ClipInfo:title>The Fish Slapping Dance</title>
  <PersonInfo:title>Mr</title>
  ```

Namespaces - Declaration

- Default declaration
  ```xml
  <clip xmlns="urn:Clip.org:ClipInfo">
  <title>The Fish Slapping Dance</title>
  </clip>
  ```

- Explicit declaration
  ```xml
  <clip xmlns:clip="urn:Clip.org:ClipInfo"
       xmlns:money="urn:Finance:Money">
    <clip:title>The Fish Slapping Dance</clip:title>
    <clip:price money:currency="US dollar">14.76</clip:price>
  </clip>
  ```
Document Grammar Specifications

- XML Schema
  - Provides greater functionality than Document Type Definition
  - Developed later (still in WD form)
  - Aimed at Structured Data, not just Documents
  - Therefore, Full Data-type support
  - Uses XML syntax
  - Association with XML document:
    `<clip xmlns="x:schema:clipSchema.xml">

XML Schema

- Validation of documents with markup from different namespaces
- Extensibility
  - Schema authors can add their own elements and attributes to XML Schema documents
- Default element content
- Data types with possibility of constraint specifications
- User defined data types

XML Schema - MS XDR Example

```xml
<Schema xmlns="urn:schemas-microsoft-com:datatypes" xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <AttributeType name="unit" dt:Type="string" required="yes"/>
  <ElementType name="title" content="textOnly" dt:Type="string"/>
  <ElementType name="size" content="textOnly" dt:Type="int">
    <attribute type="unit"/>
  </ElementType>
  <ElementType name="clip" content="allOnly">
    <element type="title"/>
    <element type="size"/>
  </ElementType>
</Schema>
```

More on XML Schema

- Fortunately, XML Schema is actually notated in XML itself
- Elements
- Attributes
- Types
  - A type is a collection of constraints on element content and attribute values
  - A type may be either
    - simple, for constraining string values
    - complex, for constraining elements which contain other elements

The XML Schema worldview

- Validity and well-formedness are XML 1.0 concepts
  - They are defined over character sequences
- Namespace-compliant
  - It's defined over character sequences too
- XML Schema *Schema-validity* is layered on top of XML 1.0 well-formedness plus Namespaces
  - XML document Infosets = Validity + WF + NS

Valid and Well-Formed XML Documents

- Well-formed document:
  - contains one or more elements
  - there is precisely one root element
  - all other tags nest within each other correctly
- Valid document: document complies with DTD/Schema
  - content model validity: order and nesting
  - data type validity: correct type and other constraints satisfaction e.g. value range
**Why validate?**
- DTD/Schema guarantees an interface
- Producers validate to ensure they are providing what they promised
- Consumers validate to check up on producers
  - and to protect their applications
- Application authors validate to simplify their task
  - Leave error detection and analysis to the validating parser

**Document Object Model**
- Programming API for XML documents
- Describes logical structure of document and the way a document is accessed and manipulated
- Defines naming convention for document components
- Enables straightforward access to the document components
- Implemented by the tools that enable manipulation of XML documents

**Document Object Model - Example**
```java
root = doc.getDocumentElement();
//print tag name
System.out.println(root.getTagName());
//get first child element of the root
docElem = (Node) root.getFirstChild();
//print tag name
System.out.println(docElem.getNodeName());
//print element type
System.out.println(docElem.getNodeType());
//print node value
docElem1 = (Text) docElem.getFirstChild();
System.out.println(docElem1.getNodeValue());
```

**XML Data Islands**
- XML code embedded in an HTML page
- Enables integration of XML with HTML page (XML data can be accessed by scripts)
- Contains well-formed XML document within `<XML>` tags
```
<XML ID="clipXML"/>
<clip>
  <title>The Fish Slapping Dance</title>
  <size unit="KB">829</size>
</clip>
</XML>
```
clipXML.documentElement.childNodes.item(0).text

**Uses for XML**
- Specific Languages for Specific Purposes
- XML for Message Exchange
- Programming with XML
  - SAX vs. DOM Parsers
  - Validation
- Storing objects as XML documents
Languages Based on XML

- Resource Description Framework (RDF) standard for metadata exchange, enables better content searching on the Web
- Synchronized Multimedia Integration Language (SMIL) enables simple authoring of TV-like multimedia presentations such as training courses on the Web
- Scalable Vector Graphics (SVG) - a language for describing two-dimensional graphics in XML

Evolution of XML

- Many XML languages, optimised for different roles
  - MathML -- for mathematics
  - SMIL -- for synchronised multimedia
  - RDF -- for describing "things"
  - XUL -- for describing the Navigator 5 user interface

The XML Family Tree

SMIL - General Information

- Synchronized Multimedia Integration Language - allows integrating a set of multimedia objects into a synchronized multimedia presentation
- SMIL provides mechanisms for
  - description of temporal behavior of the presentation
  - description of layout of the presentation on the screen
  - association of hyperlinks with media objects

SMIL - Basic Concepts

- Layout: the layout of the visual clips can be defined, the clips can be assigned to the predefined separate regions
- Clip playback: the clips can be played from various sources and in different modes
- Clip temporal dependency: the clips can be played in parallel or sequential manner
- Hyperlinks: a clip can be connected to another clip or presentation

SMIL Example

```xml
<smil>
  <head>
    <meta name="title" content="Online Teaching Services promo"/>
    <meta name="author" content="Jay Moonah, CAT"/>
    <meta name="language" content="en"/>
  </head>
  <body>
    <p>Text for the slide</p>
    <img src="image.png" alt="Image description"/>
    <audio src="audio.mp3" controls="true"/>
    <video src="video.mp4" controls="true"/>
    <object data="object.swf" type="application/x-shockwave-flash"/>
    <button>Click me</button>
  </body>
</smil>
```
**SMIL - Example**

- The example shows a presentation created using SMIL
- The presentation is displayed using RealPlayer from RealNetworks

---

**MathML**

- Designed to express **semantics** of maths
- Also can express **layout**
- Cut & paste into Maple, Mathematica

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow><mrow><mi>x</mi><mo>^</mo><mn>2</mn><mo>+</mo><mn>4</mn><mi>x</mi><mo>+</mo><mn>4</mn></mrow><mo>=</mo><mn>0</mn></mrow>
</math>
```

---

**XHTML: NextGen HTML**

```xml
<?xml version="1.0" encoding="iso-8859-1" ?>
<html xmlns="http://www.w3.org/TR/xhtml1" >
<head>
<title>Title of text XHTML Document</title>
</head>
<body>
<div id="myDiv">
<h1>Heading of Page</h1>
<p>here is a paragraph of text. I will include inside this paragraph a bunch of wonky text so that it looks <em>funny</em>. </p>
<p>here is another paragraph with <em>important</em> emphasis</p>
<p>and absolutely no <b>onion</b> sense of humor.</p>
<p>And another paragraph, this one with an <img src="image.jpg" alt="waste of time" /></img> image, and a <br /> line break. </p>
</div>
</body>
</html>
```

---

**XHTML**

- Just like HTML, but based on XML rules
- Will support **integration** of different data into a single document

---

**Other Use: Data Abstraction**

- XML as a **universal format** for data interchange
- Machines exchange data as XML-format messages
- Eliminates proprietary data formats
- Lots of XML processing software available

---

**XML Messaging**

![XML Messaging Diagram]
Example Message

<xpartorders xmlns="http://myco.org/Spec/partorders.desc">
  <order ref="z3-2112-2342" date="25aug1999-12:34:23h">
    <desc> Gold sprocket grommets, with matching hamster</desc>
    <part number="23-23221-a12"/>
    <quantity units="gross"> 12 </quantity>
    <delivery-date date="27aug1999-12:00h">
      Order something else .....
    </delivery-date>
  </order>
</xpartorders>

XML Tools

1. Parsers, used for accessing, analyzing and transforming XML documents (Sun, Microsoft, IBM)
   1. Validating parsers
   1. Non-validating parsers
1. Editors, used for creating XML content (Microsoft, IBM)
1. Java APIs for XML (Sun)

XML Editor - Example

1. The example demonstrates abilities of XML Notepad

XML Parsers

1. Validation
1. Application Programming Interfaces
   1. Document Object Model (DOM) - tree based
     1. compiles an XML document into an internal tree structure
     1. allows application to navigate the tree
   1. Simple API for XML (SAX) - event based
     1. reports parsing events directly to the application through callbacks
     1. does not usually build an internal tree

SAX Parser

```java
import org.xml.sax.HandlerBase;
import org.xml.sax.AttributeList;
public class MyHandler extends HandlerBase {
  String tag = "outside";
  int indent = 0;

  public void startElement (String name, AttributeList atts)
      throw SAXException;
  )
  {
    indent = indent + 2;
    for (i = 0; i < indent; i++) {
      System.out.print(" ");
    }
    System.out.println("Start element: " + name);
    tag = "inside";
  }
```

Browsing Document with SAX Parser
Browsing Document with SAX Parser

```java
import org.xml.sax.Parser;
import org.xml.sax.DocumentHandler;
import org.xml.sax.helpers.ParserFactory;

public class XMLContent {
    static final String parserClass = "com.ibm.xml.parsers.SAXParser";
    public static void main (String args[]) throws Exception {
        Parser parser = ParserFactory.makeParser(parserClass);
        DocumentHandler handler = new MyHandler();
        parser.setContentHandler(handler);
        for (int i = 0; i < args.length; i++) {
            parser.parse(args[i]);
        }
    }
}
```

Browsing Document with DOM Parser

```java
import org.w3c.dom.*;
import com.ibm.xml.parsers.DOMParser;
public class DOMAccess {
    static final String parserClass = "com.ibm.xml.parsers.DOMParser";
    public static void main (String args[]) throws Exception {
        DOMParser parser = new DOMParser();
        Document document;
        NodeList publications;
        Element author;
        Element lastname;
        Text nameString;
        parser.parse(args[0]);
    }
}
```

Browsing Document with DOM Parser

```java
document = parser.getDocument();
root = document.getDocumentElement();
System.out.println("Node name: " + root.getNodeName());
publications = root.getElementsByTagName("publication");
publication = (Element) publications.item(0);
System.out.println("Node name: " + publication.getNodeName());
author = (Element) publication.getElementsByTagName("author").item(0);
System.out.println("Node name: " + author.getNodeName());
lastname = (Element) author.getElementsByTagName("lastname").item(0);
System.out.println("Node name: " + lastname.getNodeName());
namestring = (Text) lastname.getFirstChild();
System.out.println("Last Name: " + namestring.getData());
```

Browsing Document with DOM Parser - Results

```
Start element: doc
Start element: publication
Start element: title
Content: Collaborative Virtual Workspace
End element: title
Start element: author
Start element: lastname
Content: Spellman
End element: lastname
Start element: firstname
Content: Peter
End element: firstname
Start element: date
Content: 1997
End element: date
Start element: keywords
Start element: keyword
Content: collaboration framework
```

Browsing Document with SAX Parser - Results

```
Node name: doc
Node name: publication
Node name: author
Node name: lastname
Last Name: Spellman
```
Validation of Document

```java
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.helpers.XMLReaderFactory;

public class Validator {
    static final String parserClass =
        "com.ibm.xml.parsers.ValidatingSAXParser";
    
    public static void main(String[] args) throws Exception {
        Source source =
            SourceFactory.newInstance(sourceClass);
        SAXParser parser =
            SAXParserFactory.newInstance().newSAXParser();
        XMLReader reader =
            SAXParserFactory.newInstance().newSAXParser();
        reader.setFeature("http://xml.org/sax/features/validation", true);
        reader.setAttribute("http://xml.org/sax/properties/feature/validating", true);
        reader.setErrorHandler(new ErrorHandler()
        {
            public void warning(SAXParseException e)
            {
                System.out.println("Warning:");
            }
            public void error(SAXParseException e)
            {
                System.out.println("Error:");
                System.out.println("XML document has been validated.");
            }
            public void fatalError(SAXParseException e)
            {
                System.out.println("Fatal error:");
            }
        });
        reader.parse(source);
    }
}
```

Validation of Document - Modification

```
<doc>
  <publications number="xml">
    <title>Something More Interesting</title>
    <author>
      <lastName>Spellman</lastName>
    </author>
    <firstName>Peter</firstName>
    <date>1997</date>
  </publications>
</doc>
```

Validation of Document - Results

```
D:\doc\validate\xml\publications.xml:15:24: Attribute "number" is not declared in element "publications".
D:\doc\validate\xml\publications.xml:16:18: Element "publications" is not declared in the DTD.
D:\doc\validate\xml\publications.xml:56:7: Element "<doc>" is not valid because it does not follow the rule, "(publication)".
```

Objects as XML Documents

- Customizing Java serialization mechanism
  - `java.io.Serializable`
  - `readObject(java.io.ObjectInputStream)`
  - `writeObject(java.io.ObjectOutputStream)`
- Solution for Beans
  - Use of information gathered from `BeanInfo` class
  - Use of `set` methods for each object field

XSL Stylesheets

- Overview
- Process
- Result tree construction
- XSL template element
- XSL patterns
- Important XSL elements
- Displaying XML data in Web browsers
Extensible Stylesheet Language (XSL) Overview

- Enables display of XML by transforming XML into structure suitable for display, for example HTML
- XSL transformations can be executed on the server to provide HTML documents for older browsers
- Provides mechanisms for transformation of XML data from one schema to another
- Enables conversion of XML documents through querying, sorting, and filtering
- Association with XML document:
  
  ```xml
  <?xml-stylesheet type="text/xsl" href="mystyle.xsl"?>
  ```

XSL - Style Sheets

- Contain a template of the desired result structure
- Identify data in the source document to insert into the template
Example: Fragments of XSL document define how elements of XML document should be transformed into HTML document

XSL Process

- Construction of source tree from XML document
- Transformation of source tree to result tree using stylesheet in XSL document
- Application of style rules to each node of result tree
- Display of document by user agent using appropriate styling on a display, on paper or some other medium

XSL - Example

- The example illustrates how the XSL document is applied to XML document and displayed in the Web browser
- The example must be viewed using Internet Explorer 5.0
- URL:
  
  http://msdn.microsoft.com/xml/samples/transform-viewer/transform-viewer.htm

XSL Patterns

- Simple query language for identifying nodes in an XML document
- Identify nodes depending on:
  - type, name, and values
  - relationship of the node to other nodes in the document
  - clip
  - clip/title
  - clip/*
  - clip/priceinfo/reqprice
XSL Patterns - Example

- The example shows how the parts of the XML document can be identified using XSL patterns
- The example must be displayed in Internet Explorer 5.0
- URL: http://msdn.microsoft.com/xml/samples/authors/author-patterns.htm

Result Tree Construction

- Stylesheet - set of template rules
- Template rule
  - pattern - identifies the source node to which the processing is applied
  - template - the fragment to be instantiated to form a part of the result tree
- Creation of result tree: finding the template rule for the root node and instantiating its template

XSL Template Element

- Describes template rule
- match attribute - source node to which the rule applies
- Content - the template, may contain XSL formatting vocabulary
- Conflict resolution
  - most specific rule will be applied
  - priorities (priority attribute of the rule)
- Namespaces used to distinguish XSL instructions from other template content

XSL Patterns

- Matching by ID
  <xsl:template match="id(pnl)'">
- Matching by attribute
  <xsl:template match="publication[attribute(number)='pnl']">
- Matching by child
  <xsl:template match="publication[date]">
- Matching by position
  <xsl:template match="publication[first-of-type()]">

XSL Patterns

- Matching by name
  <xsl:template match="publication"></xsl:template>
- Matching by ancestry
  <xsl:template match="publication/title"></xsl:template>
- Matching several names
  <xsl:template match="title|keyword"></xsl:template>
- Matching the root
  <xsl:template match="/">
- Wildcard matches
  <xsl:template match="*">

Other Important Elements

- Applies template rules to the children of the node
  <xsl:value-of select="pattern"></xsl:value-of>
- Extracts value of element pattern
  <xsl:for-each select="pattern">
- Performs operation for each element described by pattern
  <xsl:sort select="key"></xsl:sort>
- Used in apply-template or for-each element, sorts children according to the key
XML Before XSL X-form

<?xml version="1.0"?>
<!DOCTYPE doc SYSTEM "pubgrammar.dtd">
<doc>
  <publication number="pnl">
    <title>Collaborative Virtual Workspace</title>
    <author>
      <lastname>Spelman</lastname>
      <firstname>Peter</firstname>
    </author>
    <date>1997</date>
    <keywords>
      <keyword>collaboration framework</keyword>
      <keyword>virtual environments</keyword>
    </keywords>
  </publication>
</doc>

XML Benefits

- 21st Century ASCII
- Validation
- Good representation of tree data
- Multiple views thru XSL
**XML Benefits**

- Supported by all major vendors, including Microsoft, IBM, Netscape, Sun
- Easy Client-side manipulation
  - Designed to be easy to parse
  - 26K of Java code (Aelfred)
  - 5K of JavaScript
- Free XML parsers available, even for commercial use

**Resources**

- More tutorials and information for developers:
  - http://www.xml.com
- Resources at NPAC:

**References**

- XML Applications by Frank Bomphe and al.
- XML Complete by Steven Holtzner
- Extensible Stylesheet Language (XSL) Version 1.0, W3C Working Draft 16-Dec-98
- Extensible Markup Language (XML) 1.0, W3C Recommendation 10-Feb-98
- SAX - http://www.megginson.com/SAX
- Various Info - www.xml.com