Experiment No.: 16

Title : Write XSLT stylesheet to convert XML document to HTML.

Objectives:

1. To understand structure and working of XSL stylesheet.
2. To understand elements of stylesheet.
3. To prepare a stylesheet and attach it to the xml document.

Theory:

XSL stands for E**X**tensible **S**tylesheet **L**anguage. The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.

The components of the XSL language:

The full XSL language logically consists of three component languages which are described in three W3C (World Wide Web Consortium) Recommendations:

XPath: XML Path Language--a language for referencing specific parts of an XML document.
XSLT: XSL Transformations--a language for describing how to transform one XML document (represented as a tree) into another.
XSL-FO: Extensible Stylesheet Language Formatting Objects--XSLT plus a description of a set of Formatting Objects and Formatting Properties

XML Stylesheet Language for Transformations (**XSLT**), defined by the W3C XSL Working Group, describes a language for transforming XML documents into other XML documents or into other formats.

For example, One might use XSLT to produce HTML, or a different XML structure. One could even use it to produce plain text or to put the information in some other document format. To perform the transformation, it is necessary to supply a stylesheet, which is written in the XML Stylesheet Language (XSL). The XSL style sheet specifies how the XML data will be displayed, and XSLT uses the formatting instructions in the style sheet to perform the transformation.

An XSL stylesheet:

An XSL stylesheet basically consists of a set of templates. Each template "matches" some set of elements in the source tree and then describes the contribution that the matched element makes to the result tree.

Generally, elements in a stylesheet in the "xsl" namespace are part of the XSLT language, and non-xsl elements within a template are what get put into the result tree

The Format of a Stylesheet

```
<?xml version= “1.0” ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
. . .
        match rules followed by action.

</xsl:stylesheet>
```

DTE CSE

Referring stylesheet in xml document

W3C defines the **xml:stylesheet** processing instruction to associate a xml document to XSL stylesheet. This instruction is added to xml file as follows:

<?xml:stylesheet href = " stylesheetname.xsl" type="text/xsl" ?>

Details of xslt:

- XSLT is rule-based:

XSLT is different from conventional programming languages because XSLT is based on template rules, which specify how XML documents should be processed. Although conventional programming languages are often sequential, template rules can be based in any order because XSLT is a declarative language. The stylesheet declares what output should be produced when a pattern in the XML document is matched.

XSLT Namespace

Every XSL file needs to specify the XSL namespace so that the parser knows which version of XSLT to use. The most current namespace is:

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

Versions of the Microsoft parser MSXML before version 3.0 used the temporary namespace http://www.w3.org/TR/WD-xsl" version="1.0". This is now outdated and doesn't conform to the latest W3C recommendation.

The namespace prefix **xsl:** is used in the rest of the XSL file to identify XSL processing statements. If a statement isn't prefixed with **xsl:**, then it's simply copied to the output without being processed. This is the way to add HTML statements to the output.

XSL top level elements:

1. xsl:template

xsl:template is the most important top level element in an XSL stylesheet. XSL transformation works by applying one or more template rules to an xml document. Template rule then work by looking for matches on xml elements and then exercising the remainder of the rule on the matching element or elements in the order they encountered. If many elements matches the rule, the rule is applied to each in the order they appear. In addition, a template rule might contain instruction to continue to match on the children of the current element. In this way, XSL transform work sequentially throughout a document and recursively through the children of the element.

All xsl:template rules follows the pattern:

```
<xsl:template match = " some match criteria" >
        some replacement text and/or additional rules
</xsl:template>
```

Match Patterns (Locating Elements)

One critical capability of a stylesheet language is to locate source elements to be styled. XSLT does it with "match patterns" defined by the XML Path Language (XPath)

DKTE CSE

XPath has an extensible string-based syntax. It describes "location paths" between parts of a document or documents. One inspiration for XPath was the common "path/file" file system syntax

Key points about XPath expressions:

- Pattern matching occurs in a context; XPath expressions and XSLT elements can change the current context and consequently the nodes which match.
- XPath is inclusive or greedy, it addresses all matching elements. You must use predicates to refine the set of nodes selected.

Pattern Examples

<xsl:template match= “para” > → Matches all <para> children in the current context.

<xsl:template match= “para/emphasis” > → Matches all <emphasis> elements that have a parent of <para>

<xsl:template match= “/”> → Matches the root of the document(root of xml document is different from document root element defined in xml document).

<xsl:template match= “para//emphasis”>

Matches all <emphasis> elements that have an *ancestor* of <para>.

<xsl:template match= “section/para[1]”>

Matches the first <para> child of all the <section> children in the current context.

<xsl:template match= “//title”>

Matches *all* <title> elements anywhere in the document
<xsl:template match= “.//title”>

Matches all <title> elements that are descendants of the current context.

More Complex Patterns

<xsl:template match= “section/*/note” >

Matches <note> elements that have <section> grandparents.

<xsl:template match= “stockquote[@symbol]”

Matches <stockquote> elements that have a "symbol" attribute
<xsl:template match= “stockquote[@symbol="XXXX"]” >

Matches <stockquote> elements that have a "symbol" attribute with the value "XXXX"

<xsl:template match= “emphasis|strong”

Matches <emphasis> or elements

DKTE CSE

Pattern Syntax

The XPath pattern syntax is described formally in the XPath specification.
An XPath pattern is a 'location path', a location path is absolute if it begins with a slash ("/") and relative otherwise. A relative location path consists of a series of steps, separated by slashes

A step is an axis specifier, a node test, and a predicate. The formal syntax is made more complicated by the fact that it must describe both the abbreviated and unabbreviated syntaxes

Node Tests

Node tests are most frequently element names, but other node tests are possible:

<xsl:template match= "*">

Matches any element node
<xsl:template match= "namespace:name" >

Matches <name> element nodes in the specified namespace
<xsl:template match= "namespace:*">

Matches any element node in the specified namespace
<xsl:template match= "comment()" >

Matches comment nodes
<xsl:template match= "text()" >

Matches text nodes

<xsl:template match= "processing-instruction()" >

Matches processing instructions

<xsl:template match= "processing-instruction('target')" >

Matches processing instructions with the specified target (<?target ...?>
<xsl:template match= "node()">

Matches any node

Predicates

Predicates occur after the node test in square brackets. A wide range of expressions is possible.

<xsl:template match= "nodetest[1]" >

Matches the first node

Most node tests return nodes in document order, only the tests which select ancestor or preceding elements return nodes in reverse document order. The practical result of this is

DKTE CSE

that the "first" node is almost always the one closest to the context node, although parenthesis can change the effective order.

Nodetest [position()=last()] -- Matches the last node
Nodetest [position() mod 2 = 0] ---Matches even nodes
element [@id='foo'] --- Matches the element(s) with id attributes whos value is "foo"
element [not(@id)] ----Matches elements that don't have an id attribute
author[firstname="Norman"] --- Match <author> elements that have<firstname>childrenwith the content "Norman".
author[normalize-space(firstname)="Norman"] -- Match "Norman" without regard to leading and trailing space.

2. xsl:apply-template

The apply-template element is always found inside a template body.
It defines a set of nodes to be processed. In this process, it then may find any 'sub' template rules to process (child elements of element in context).

If you want the apply-template instruction to only process certain child elements, you can define a select attribute, to only process specific nodes.

In the following example, we only want to process the 'name' and 'address' elements in the root document element:

```
<xsl:template match="/">
        <xsl:apply-templates select="person/name | person/address"/>
</xsl:template>
```

3. xsl:value-of

The xsl:value-of instruction is one of the ways of writing the value of text to the output of the node in context.

Example:
<xsl:value-of select="NAME"/>
passes contents of NAME element to the output.

Other ways to output text:
- writing the text in the stylesheet : Here is text
- **<xsl:copy>**
- **<xsl:copy-of>**

Key concepts : stylesheet, XSLT, XPATH, template.

Procedure:

1. prepare source xml file.
2. prepare xsl file with required templates.
3. open the xml file in Intenet Explorer 6.0.

DKTE CSE