**XSLT**

E**X**tensible **S**tylesheet **L**anguage **T**ransformation commonly known as XSLT is a way to transform the XML document into other formats such as XHTML. This tutorial explains the basics of XSLT. It contains chapters discussing all the basic components of XSLT with suitable examples.

**XSLT Overview**

**XSL**

Before learning XSLT, we should first understand XSL which stands for E**X**tensible **S**tylesheet **L**anguage. It is similar to XML as CSS is to HTML.

**Need for XSL**

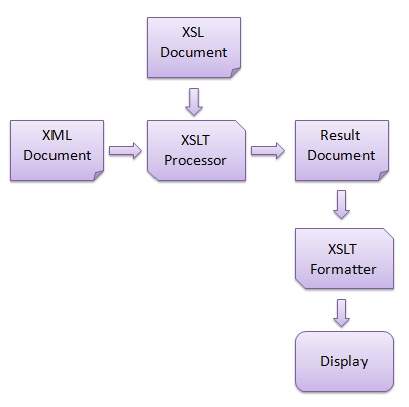
In case of HTML document, tags are predefined such as table, div, and span; and the browser knows how to add style to them and display those using CSS styles. But in case of XML documents, tags are not predefined. In order to understand and style an XML document, World Wide Web Consortium (W3C) developed XSL which can act as XML based Stylesheet Language. An XSL document specifies how a browser should render an XML document.

**What is XSLT**

XSLT, Extensible Stylesheet Language Transformations, provides the ability to transform XML data from one format to another automatically.

**How XSLT Works**

An XSLT stylesheet is used to define the transformation rules to be applied on the target XML document. XSLT stylesheet is written in XML format. XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.



Advantages

Here are the advantages of using XSLT −

* Independent of programming. Transformations are written in a separate xsl file which is again an XML document.
* Output can be altered by simply modifying the transformations in xsl file. No need to change any code. So Web designers can edit the stylesheet and can see the change in the output quickly.

Let’s suppose we have the following sample XML file, students.xml, which is required to be transformed into a well-formatted HTML document.

**student.xml**

<?xml version = "1.0"?>

<class>

<student rollno = "393">

<firstname>Dinkar</firstname>

<lastname>Kad</lastname>

<nickname>Dinkar</nickname>

<marks>85</marks>

</student>

<student rollno = "493">

<firstname>Vaneet</firstname>

<lastname>Gupta</lastname>

<nickname>Vinni</nickname>

<marks>95</marks>

</student>

<student rollno = "593">

<firstname>Jasvir</firstname>

<lastname>Singh</lastname>

<nickname>Jazz</nickname>

<marks>90</marks>

</student>

</class>

We need to define an XSLT style sheet document for the above XML document to meet the following criteria −

* Page should have a title **Students**.
* Page should have a table of student details.
* Columns should have following headers: Roll No, First Name, Last Name, Nick Name, Marks
* Table must contain details of the students accordingly.

**Step 1:** Create XSLT document

Create an XSLT document to meet the above requirements, name it as students.xsl and save it in the same location where students.xml lies.

**student.xsl**

<?xml version = "1.0" encoding = "UTF-8"?>

<!-- xsl stylesheet declaration with xsl namespace:

Namespace tells the xlst processor about which

element is to be processed and which is used for output purpose only

-->

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

<!-- xsl template declaration:

template tells the xlst processor about the section of xml

document which is to be formatted. It takes an XPath expression.

In our case, it is matching document root element and will

tell processor to process the entire document with this template.

-->

<xsl:template match = "/">

<!-- HTML tags

Used for formatting purpose. Processor will skip them and browser

will simply render them.

-->

<html>

<body>

<h2>Students</h2>

<table border = "1">

<tr bgcolor = "#9acd32">

<th>Roll No</th>

<th>First Name</th>

<th>Last Name</th>

<th>Nick Name</th>

<th>Marks</th>

</tr>

<!-- for-each processing instruction

Looks for each element matching the XPath expression

-->

<xsl:for-each select="class/student">

<tr>

<td>

<!-- value-of processing instruction

process the value of the element matching the XPath expression

-->

<xsl:value-of select = "@rollno"/>

</td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

Step 2: Link the XSLT Document to the XML Document

Update **student.xml** document with the following xml-stylesheet tag. Set **href** value to students.xsl

<?xml version = "1.0"?>

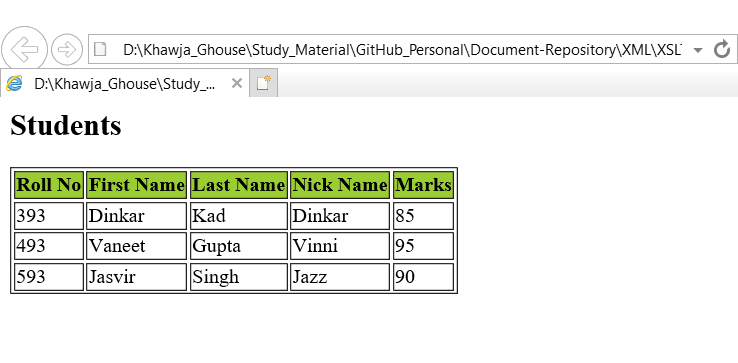
<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>

<class>

...

</class>

Step 3: View the XML Document in Internet Explorer



1. **<xsl:template>**

***<xsl:template>*** defines a way to reuse templates in order to generate the desired output for nodes of a particular type/context.

**Declaration**

Following is the syntax declaration of the **<xsl:template>** element.

<xsl:template

name = Qname

match = Pattern

priority = number

mode = QName >

</xsl:template>

**Attributes:**

1. name: Name of the element on which template is used to applied.
2. match: Pattern which signifies the elements on which template is used to be applied.
3. priority: Priority number of a template. Matching template with low priority is not considered in from in front of high priority template.
4. mode: Allows element to be processed multiple times to produce a different result each time.

**Elements:**

**Parent Elements**: xsl:stylesheet, xsl:transform

**Child Elements**: xsl:apply-imports,xsl:apply-templates,xsl:attribute, xsl:call-template, xsl:choose, xsl:comment, xsl:copy, xsl:copy-of, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:number, xsl:param, xsl:processing-instruction, xsl:text, xsl:value- of, xsl:variable, output elements

1. **<value-of>:**

***<xsl:value-of>*** tag puts the value of the selected node as per XPath expression, as text.

**Declaration**

Following is the syntax declaration of **<xsl:value-of>** element.

<xsl:value-of

select = Expression

disable-output-escaping = "yes" | "no" >

</xsl:value-of>

**Attributes:**

**Select**: XPath Expression to be evaluated in the current context.

**disable-outputescaping:** Default-"no". If "yes", output text will not escape xml characters from text.

**Elements:**

**Parent Elements:**

xsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing instruction, xsl:template, xsl:variable, xsl:when, xsl:with-param, output elements.

**Child Elements**: None.

Example:

<xsl:for-each select = "class/student">

<tr>

<td><xsl:value-of select = "@rollno"/></td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

</tr>

</xsl:for-each>

1. **<for-each>**

***<xsl:for-each>*** tag applies a template repeatedly for each node.

**Declaration**

Following is the syntax declaration of **<xsl:for-each>** element.

<xsl:for-each

select = Expression >

</xsl:for-each>

**Attributes**:

Select: XPath Expression to be evaluated in the current context to determine the set of nodes to be iterated.

**Elements**:

**Parent Elements**: xsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:foreach, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processinginstruction, xsl:template, xsl:variable, xsl:when, xsl:with-param, output elements.

**Child Elements**: xsl:apply-imports, xsl:apply-templates, xsl:attribute, xsl:call-template, xsl:choose, xsl:comment, xsl:copy, xsl:copy-of, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:number, xsl:processing-instruction, xsl:sort, xsl:text, xsl:value-of, xsl:variable

1. **<sort>**

***<xsl:sort>*** tag specifies sort criteria on the nodes.

**Declaration**

Following is the syntax declaration of **<xsl:sort>** element.

<xsl:sort

select = string-expression

lang = { nmtoken }

data-type = { "text" | "number" | QName }

order = { "ascending" | "descending" }

case-order = { "upper-first" | "lower-first" } >

</xsl:sort>

Attributes:

* 1. **Select:** The sorting key of the node.
  2. **Lang:** Language alphabet used to determine sort order.
  3. **data-type:** Data type of the text.
  4. **Order:** Sorting order. Default is "ascending".
  5. **case-order:** Sorting order of string by capitalization. Default is "upper-first"

**Parent Elements**: xsl:apply-templates, xsl:for-each.

**Child Elements**: None.

Example:

<xsl:for-each select = "class/student">

<xsl:sort select = "firstname"/>

<tr>

<td><xsl:value-of select = "@rollno"/></td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

</tr>

</xsl:for-each>

1. **<if>**

***<xsl:if>*** tag specifies a conditional test against the content of nodes.

**Declaration:**

Following is the syntax declaration of **<xsl:if>** element.

<xsl:if

test = boolean-expression >

</xsl:if>

**Attributes**:

test: The condition in the xml data to test.

**Parent Elements**: xsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instruction, xsl:template, xsl:variable, xsl:when, xsl:with-param, output elements.

**Child Elements**: xsl:apply-templates, xsl:attribute, xsl:call-template, xsl:choose, xsl:comment, xsl:copy, xsl:copy-of, xsl:element, xsl:for-each, xsl:if, xsl:processing-instruction, xsl:text, xsl:value-of, xsl:variable, output elements.

Example:

<xsl:for-each select = "class/student">

**<xsl:if test = "marks > 90">**

<tr>

<td><xsl:value-of select = "@rollno"/></td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

</tr>

</xsl:if>

</xsl:for-each>

1. **<choose>**

***<xsl:choose>*** tag specifies a multiple conditional tests against the content of nodes in conjunction with the <xsl:otherwise> and <xsl:when> elements.

**Declaration**

Following is the syntax declaration of **<xsl:choose>** element.

<xsl:choose >

</xsl:choose>

**Elements:**

**Parent Elements**: xsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:for-each, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processing-instruction, xsl:template, xsl:variable, xsl:when, xsl:with-param, output elements.

**Child Elements**: xsl:otherwise, xsl:when.

Example:

<xsl:for-each select = "class/student">

<tr>

<td><xsl:value-of select = "@rollno"/></td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

<td>

<xsl:choose>

<xsl:when test = "marks > 90">

High

</xsl:when>

<xsl:when test = "marks > 85">

Medium

</xsl:when>

<xsl:otherwise>

Low

</xsl:otherwise>

</xsl:choose>

</td>

</tr>

</xsl:for-each>

1. **<key>**

***<xsl:key>*** tag element specifies a named name-value pair assigned to a specific element in an XML document. This key is used with the key() function in XPath expressions to access the assigned elements in an XML document.

**Declaration**

Following is the syntax declaration of **<xsl:key>** element.

<xsl:key

name = QName

match = Pattern

use = Expression >

</xsl:key>

**Attributes**:

* 1. Name: Name of the key to be used.
  2. Match: Patterns used to identify a node that holds this key.
  3. Use: XPath expression to identify the value of the nodes of xml document.

**Parent Element:** xsl:stylesheet.

**Child Element:** None.

Example:

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

<xsl:key name = "firstname-search" match = "student" use = "firstname"/>

<xsl:template match = "/">

<html>

<body>

<h2>Students</h2>

<table border = "1">

<tr bgcolor = "#9acd32">

<th>Roll No</th>

<th>First Name</th>

<th>Last Name</th>

<th>Nick Name</th>

<th>Marks</th>

</tr>

<xsl:for-each select = "key('firstname-search', 'Dinkar')">

<tr>

<td><xsl:value-of select = "@rollno"/></td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>

1. **<message>**

***<message>*** tag element helps to debug an XSLT processing. It is similar to javascript alerts. <xsl:> tag buffers a message to XSLT processor which terminates the processing and sends a message to the caller application to display the error message.

**Declaration**

Following is the syntax declaration of **<xsl:message>** element.

<xsl:message

terminate = "yes" | "no" >

</xsl:message>

**Attributes**

* 1. terminate: It specifies whether the transformation should terminate upon executing this instruction or not. Default is "yes".

Example:

<xsl:for-each select = "class/student">

<xsl:if test = "firstname = ''">

<xsl:message terminate = "yes">A first name field is empty.

</xsl:message>

</xsl:if>

<tr>

<td><xsl:value-of select = "@rollno"/></td>

<td><xsl:value-of select = "firstname"/></td>

<td><xsl:value-of select = "lastname"/></td>

<td><xsl:value-of select = "nickname"/></td>

<td><xsl:value-of select = "marks"/></td>

</tr>

</xsl:for-each>

1. **<apply-template>**

***<xsl:apply-template>*** tag signals the XSLT processor to find the appropriate template to apply, based on the type and context of each selected node.

**Declaration**

Following is the syntax declaration of **<xsl:apply-template>** element.

<xsl:apply-template

select = Expression

mode = QName >

</xsl:apply-template>

**Attributes**:

* 1. Select: Used to process nodes selected by an XPath expression, instead of processing all the children.
  2. Mode: Allows an element as specified by its Qualified Names to be processed multiple times, each time producing a different result.

**Parent Elements**: xsl:attribute, xsl:comment, xsl:copy, xsl:element, xsl:fallback, xsl:foreach, xsl:if, xsl:message, xsl:otherwise, xsl:param, xsl:processinginstruction, xsl:template, xsl:variable, xsl:when, xsl:with-param, output elements.

**Child Elements**: xsl:sort, xsl:with-param.

Example:

**Student.xml**

<?xml version = "1.0"?>

<?xml-stylesheet type = "text/xsl" href = "students.xsl"?>

<class>

<student rollno = "393">

<firstname>Dinkar</firstname>

<lastname>Kad</lastname>

<nickname>Dinkar</nickname>

<marks>85</marks>

</student>

<student rollno = "493">

<firstname>Vaneet</firstname>

<lastname>Gupta</lastname>

<nickname>Vinni</nickname>

<marks>95</marks>

</student>

<student rollno = "593">

<firstname>Jasvir</firstname>

<lastname>Singh</lastname>

<nickname>Jazz</nickname>

<marks>90</marks>

</student>

</class>

Student.xsl

<?xml version = "1.0" encoding = "UTF-8"?>

<xsl:stylesheet version = "1.0"

xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

<xsl:template match = "/">

<html>

<body>

<h2>Students</h2>

<xsl:apply-templates select = "class/student" />

</body>

</html>

</xsl:template>

<xsl:template match = "class/student">

<xsl:apply-templates select = "@rollno" />

<xsl:apply-templates select = "firstname" />

<xsl:apply-templates select = "lastname" />

<xsl:apply-templates select = "nickname" />

<xsl:apply-templates select = "marks" />

<br />

</xsl:template>

<xsl:template match = "@rollno">

<span style = "font-size = 22px;">

<xsl:value-of select = "." />

</span>

<br />

</xsl:template>

<xsl:template match = "firstname">

First Name:<span style = "color:blue;">

<xsl:value-of select = "." />

</span>

<br />

</xsl:template>

<xsl:template match = "lastname">

Last Name:<span style = "color:green;">

<xsl:value-of select = "." />

</span>

<br />

</xsl:template>

<xsl:template match = "nickname">

Nick Name:<span style = "color:red;">

<xsl:value-of select = "." />

</span>

<br />

</xsl:template>

<xsl:template match = "marks">

Marks:<span style = "color:gray;">

<xsl:value-of select = "." />

</span>

<br />

</xsl:template>

</xsl:stylesheet>

