XSLT

extensible stylesheet language transformation

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

EXtensible Stylesheet Language Transformation, 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.

Audience

This tutorial has been prepared for beginners to help them in understanding the basic concepts related to XSLT. This tutorial will give you enough understanding on XSLT from where you can take yourself to a higher level of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic knowledge of XML, HTML, and JavaScript.

Disclaimer & Copyright

Table of Contents

1. XSLT – OVERVIEW

XLS

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 XLS

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.

Following are the main parts of XSL:

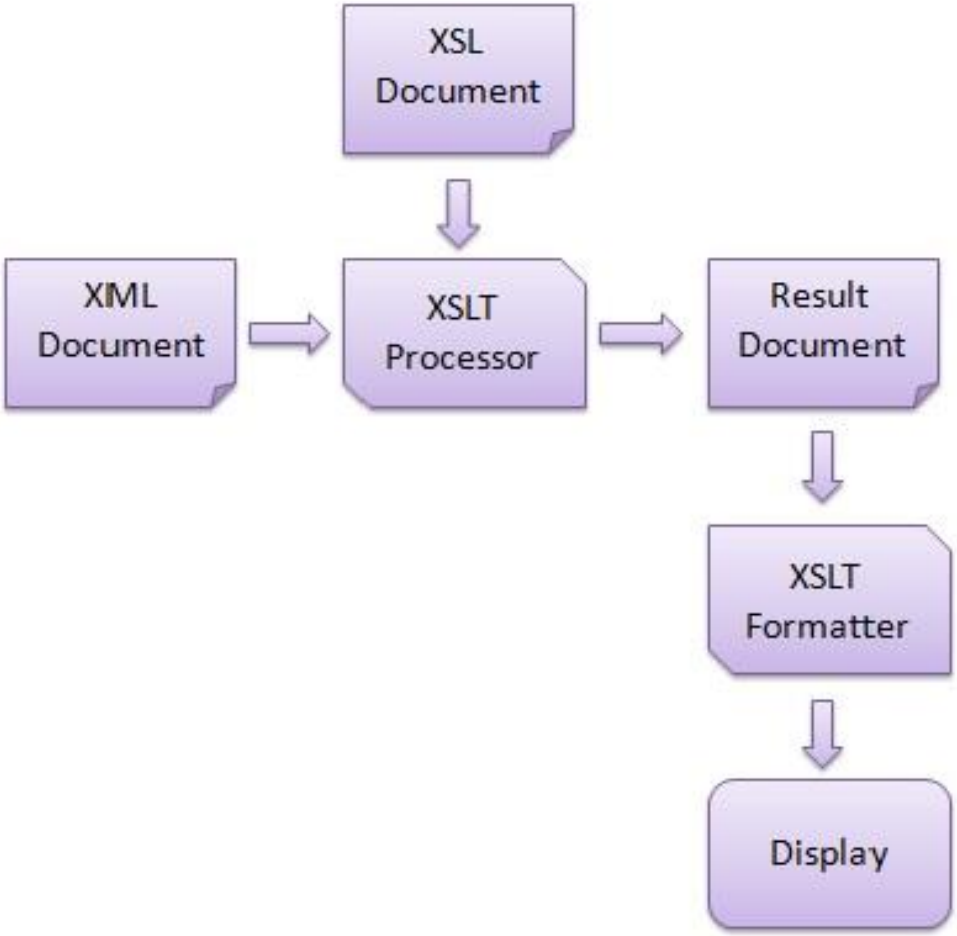
- **XSLT** - used to transform XML document into various other types of document.
- **XPath** - used to navigate XML document.
- **XSL-FO** - used to format XML document.

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.

2. XSLT SYNTAX

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

students.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.

students.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>
```

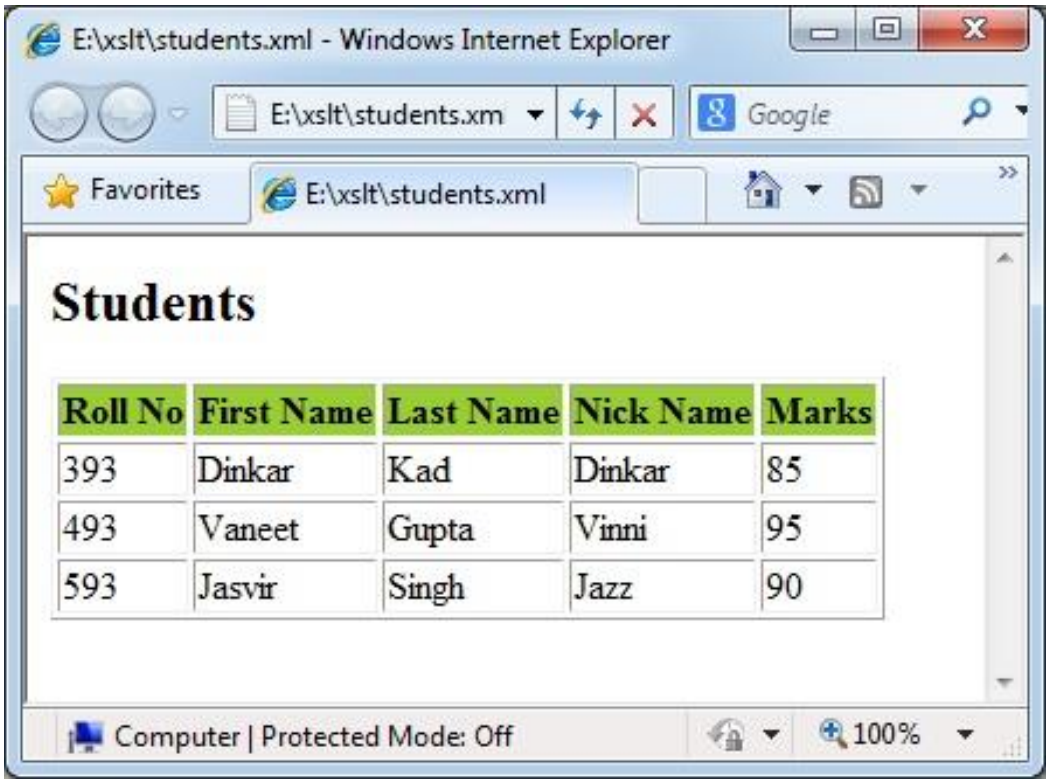
Step 3: View the XML Document in Internet Explorer

students.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>
```

Output



3. XSLT 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 **<xsl:template>** element.

```
<xsl:template
   name= Qname
   match = Pattern
   priority = number
   mode = QName >
</xsl:template>
```

Attributes

Name	Description
name	Name of the element on which template is to be applied.
match	Pattern which signifies the element(s) on which template is to be applied.
priority	Priority number of a template. Matching template with low priority is not considered in from in front of high priority template.
mode	Allows element to be processed multiple times to produce a different result each time.

Elements

Number of occurrences	Unlimited
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

Demo Example

This template rule has a pattern that identifies <student> elements and produces an output in a tabular format.

students.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>
```

students_imports.xsl

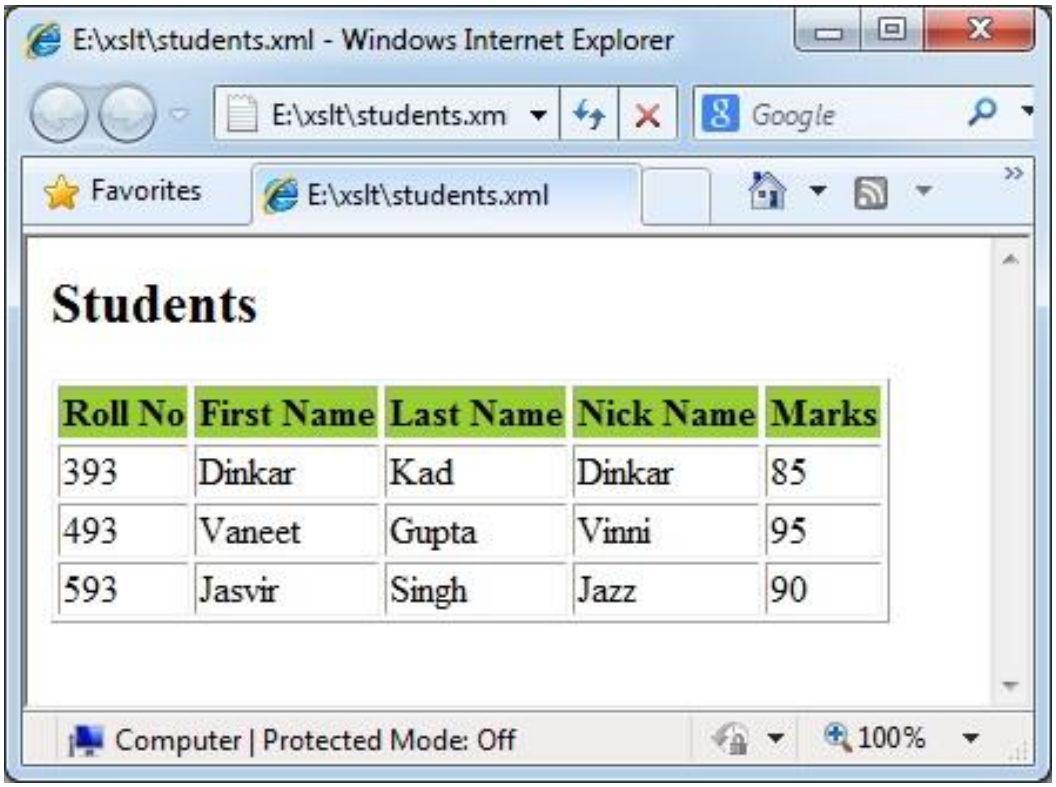
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Students</h2>
  <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="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>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Output



4. XSLT <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

Name	Description
Select	XPath Expression to be evaluated in current context.
disable-output-escaping	Default-"no". If "yes", output text will not escape xml characters from text.

Elements

Number of occurrences	Unlimited
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

Demo Example

This example creates a table of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks>.

students.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>
```

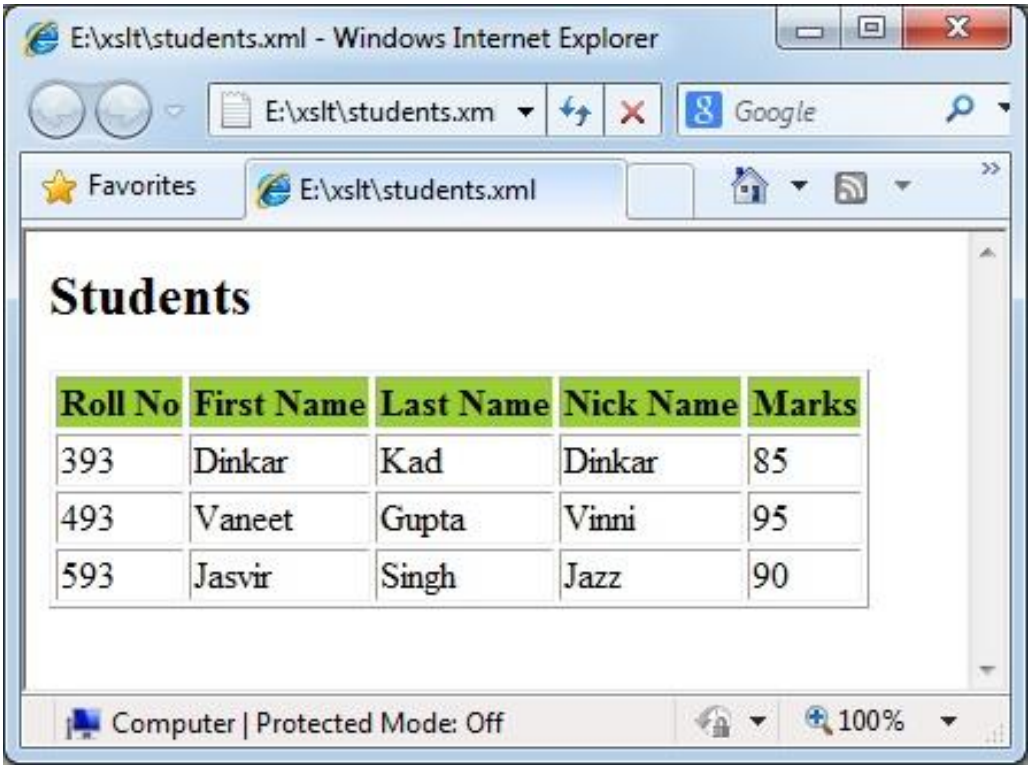
students.xsl

```
<?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>
  <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="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>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Verify the output



5. XSLT <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

Name	Description
select	XPath Expression to be evaluated in current context to determine the set of nodes to be iterated.

Elements

Number of occurrences	Unlimited
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-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

Demo Example

This example creates a table of <student> element with its attribute rollno and its child <firstname>, <lastname>, <nickname> and <marks> by iterating over each student.

students.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>
```

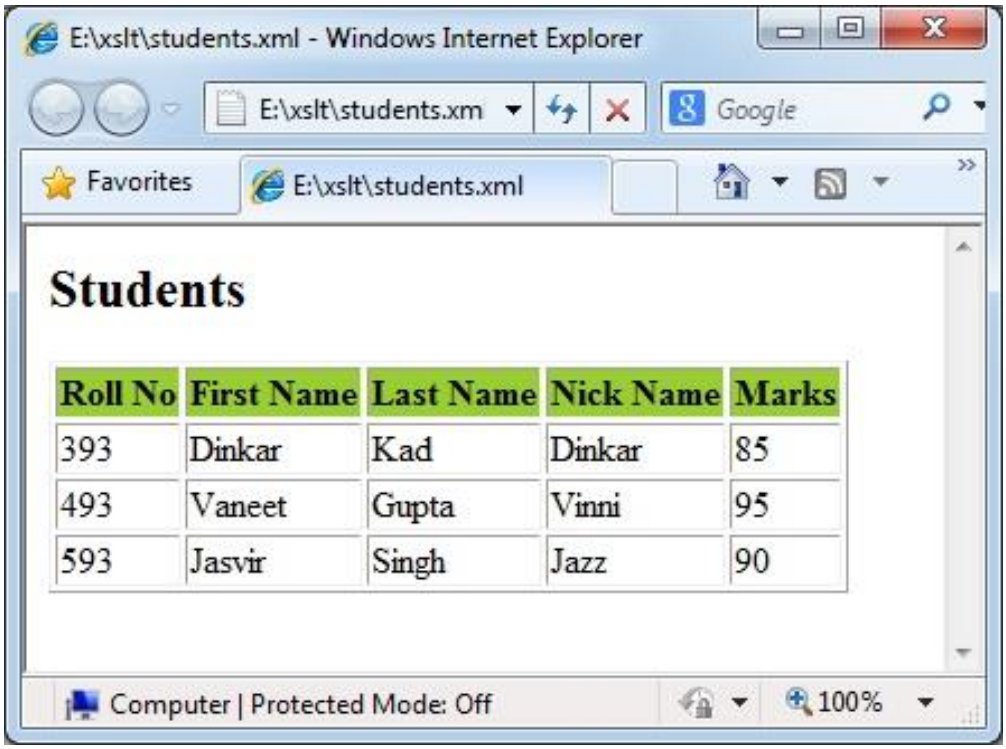
students.xsl

```
<?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>
  <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="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>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Output


E:\xslt\students.xml - Windows Internet Explorer
E:\xslt\students.xm
Google
Favorites
E:\xslt\students.xml
Students
Roll No | First Name | Last Name | Nick Name | Marks
393 | Dinkar | Kad | Dinkar | 85
493 | Vaneet | Gupta | Vinni | 95
593 | Jasvir | Singh | Jazz | 90
Computer | Protected Mode: Off
100%

6. XSLT <SORT>

<xsl:sort> tag specifies a 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

Name	Description
select	Sorting key of the node.
lang	Language alphabet used to determine sort order.
data-type	Data type of the text.
order	Sorting order. Default is "ascending"
case-order	Sorting order of string by capitalization. Default is "upper-first".

Elements

Number of occurrences	Unlimited
Parent elements	xsl:apply-templates, xsl:for-each
Child elements	None

Demo Example

This example creates a table of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks> by iterating over each student sort them by first name.

students.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>
```

students.xsl

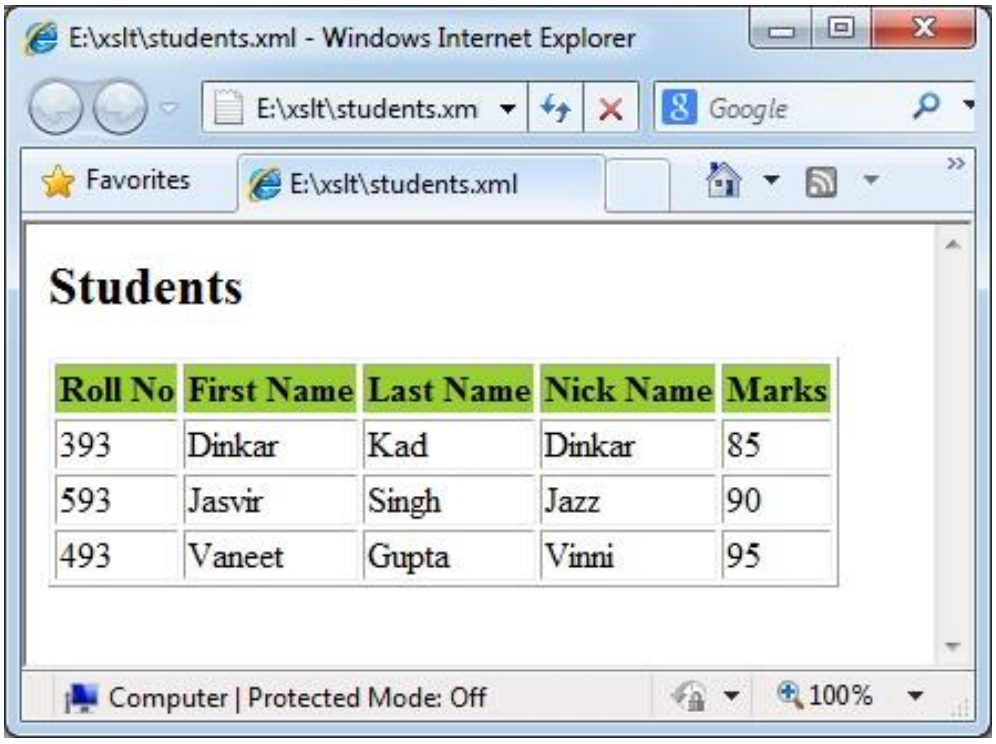
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Students</h2>
  <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="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>
  </table>
```

```
  </body>
  </html>
</xsl:template>


</xsl:stylesheet>
```

Output



7. XSLT <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

Name	Description
test	The condition in the xml data to test.

Elements

Number of occurrences	Unlimited
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

Demo Example

This example creates a table of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks> by iterating over each student. It checks marks to be greater than 90 and then prints the student(s) details.

students.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>
```

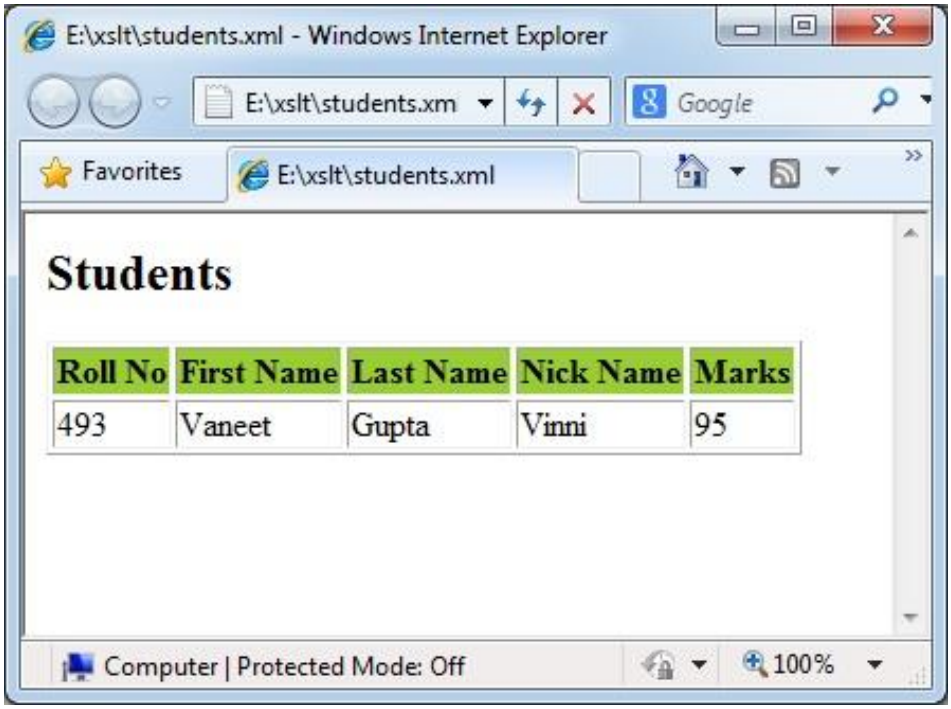
students.xsl

```
<?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>
  <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="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>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Output


E:\xslt\students.xml - Windows Internet Explorer
E:\xslt\students.xm
Google
Favorites
E:\xslt\students.xml
Students
Roll No
First Name
Last Name
Nick Name
Marks
493
Vaneet
Gupta
Vinni
95
Computer | Protected Mode: Off
100%

8. XSLT <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

Number of occurrences	Unlimited
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

Demo Example

This example creates a table of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks> by iterating over each student. It checks and then prints the grade details.

students.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>
```

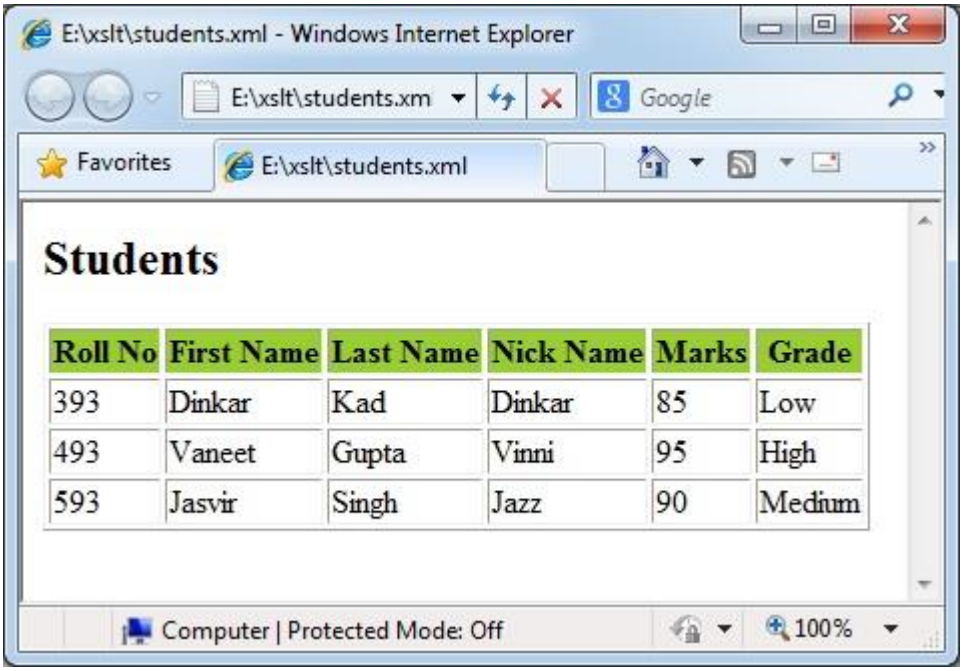
students.xsl

```
<?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>
  <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>
```

```
      <th>Grade</th>
    </tr>
   <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>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Output



Roll No	First Name	Last Name	Nick Name	Marks	Grade
393	Dinkar	Kad	Dinkar	85	Low
493	Vaneet	Gupta	Vinni	95	High
593	Jasvir	Singh	Jazz	90	Medium

9. XSLT <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

Name	Description
Name	Name of the key to be used.
Match	Patterns used to identify a node that holds this key.
Use	XPath expression to identify the value of the nodes of xml document.

Elements

Number of occurrences	Unlimited
Parent elements	xsl:stylesheet
Child elements	None

Demo Example

This example creates a table of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks> by iterating over each student. It checks key as firstname to be one of the student's name and then prints the student details.

students.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>
```

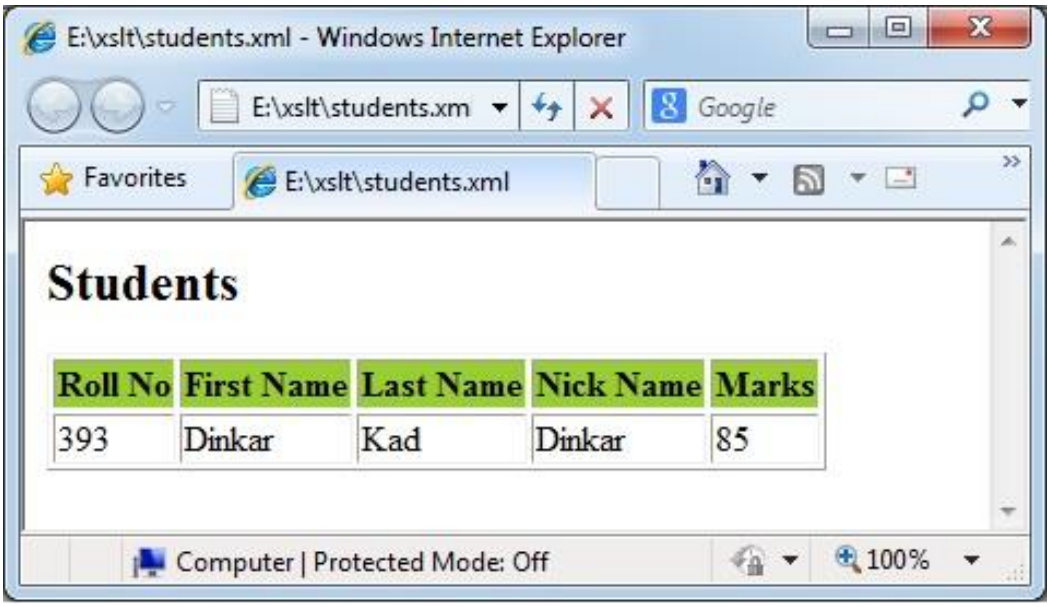
students.xsl

```
<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>
```

Output


E:\xslt\students.xml - Windows Internet Explorer
E:\xslt\students.xm
Google
Favorites
E:\xslt\students.xml
Students
Roll No
First Name
Last Name
Nick Name
Marks
393
Dinkar
Kad
Dinkar
85
Computer | Protected Mode: Off
100%

10. XSLT <MESSAGE>

<message> tag element helps to debug an XSLT processing. It is similar to javascript alerts. <xsl:> tag buffers a message to XLST 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

Name	Description
terminate	It specifies whether the transformation should terminate upon executing this instruction or not. Default is "yes".

Elements

Number of occurrences	Unlimited
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

Demo Example

This example creates a table of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks> by iterating over each student. It checks key as firstname to be present and then prints the student details, otherwise displays an error message.

students.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></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>
```

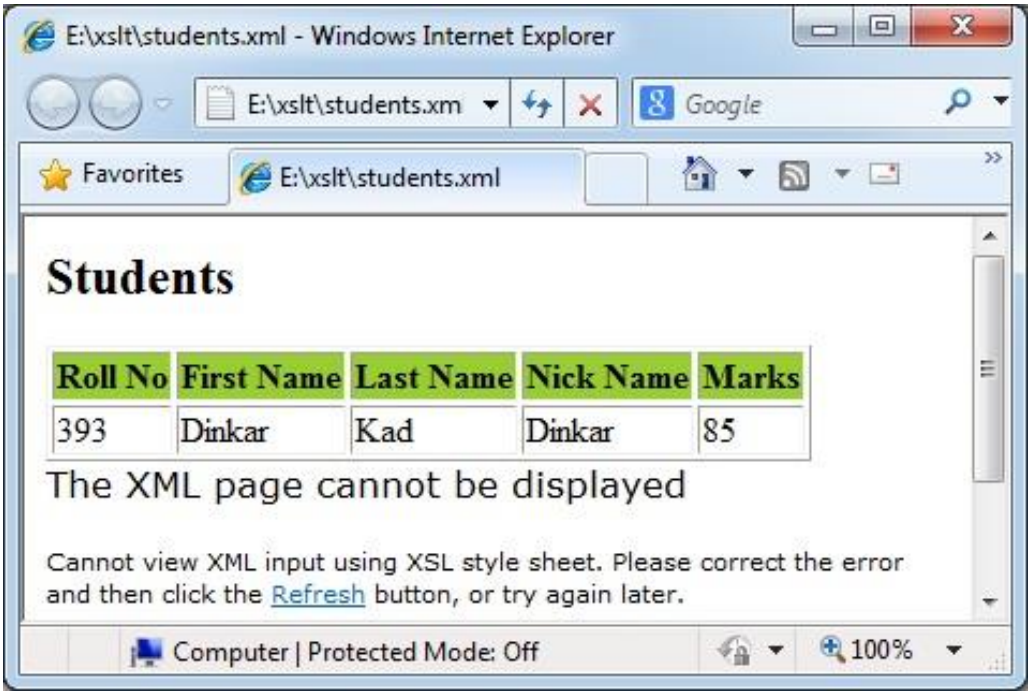
students.xsl

```
<?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>
  <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="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>
  </table>
  </body>
  </html>
</xsl:template>
```

```
</xsl:stylesheet>
```

Output



11. XSLT <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

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

Elements

Number of occurrences	Unlimited
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:sort, xsl:with-param

Demo Example

This example creates a list of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks> by iterating over each student.

students.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>
```

students.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">
```

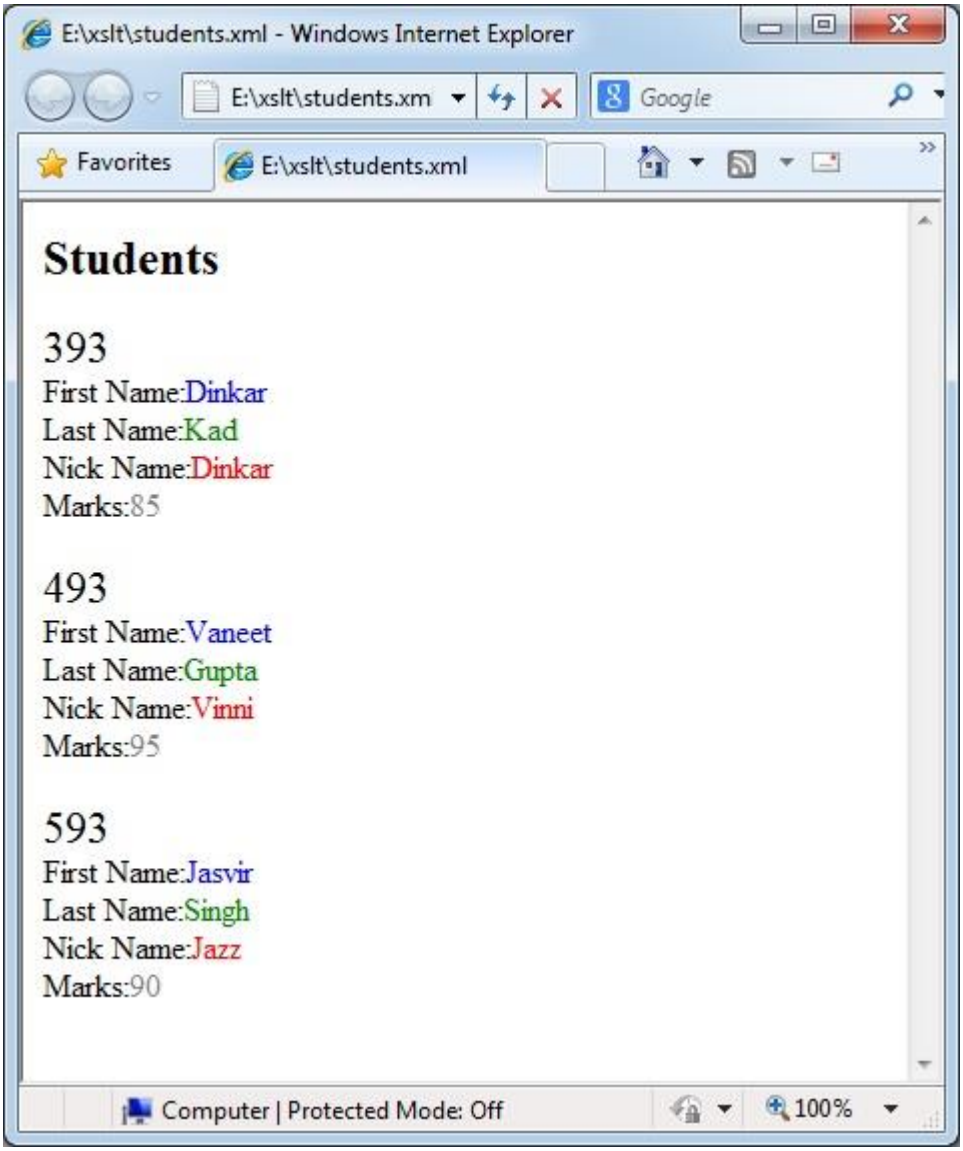
```
Last Name:<span style="color:green;">
<xsl:value-of select="." />
</span>
<br />
</xsl:template>

<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>
```

Output


E:\xslt\students.xml - Windows Internet Explorer
E:\xslt\students.xm
Google
Favorites
E:\xslt\students.xml
Students
393
First Name:Dinkar
Last Name:Kad
Nick Name:Dinkar
Marks:85
493
First Name:Vaneet
Last Name:Gupta
Nick Name:Vinni
Marks:95
593
First Name:Jasvir
Last Name:Singh
Nick Name:Jazz
Marks:90
Computer | Protected Mode: Off
100%

12. XSLT <IMPORT>

<xsl:import> tag imports the contents of one stylesheet into another. Importing a style sheet has higher precedence over imported stylesheet.

Declaration

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

```
<xsl:import href="uri">
</xsl:import>
```

Attributes

Name	Description
href	Used to pass the path of XLST stylesheet to be imported.

Elements

Number of occurrences	Unlimited
Parent elements	xsl:stylesheet, xsl:transform
Child elements	None

Demo Example

This example creates a list of <student> element with its attribute **rollno** and its child <firstname>, <lastname>, <nickname>, and <marks> by iterating over each student. Here we have created two xsl stylesheets where students_imports.xsl stylesheet imports students.xsl and students.xml is linked to students_imports.xsl.

students.xml

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="students_imports.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>
```

students.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>
  <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="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>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

students_imports.xsl

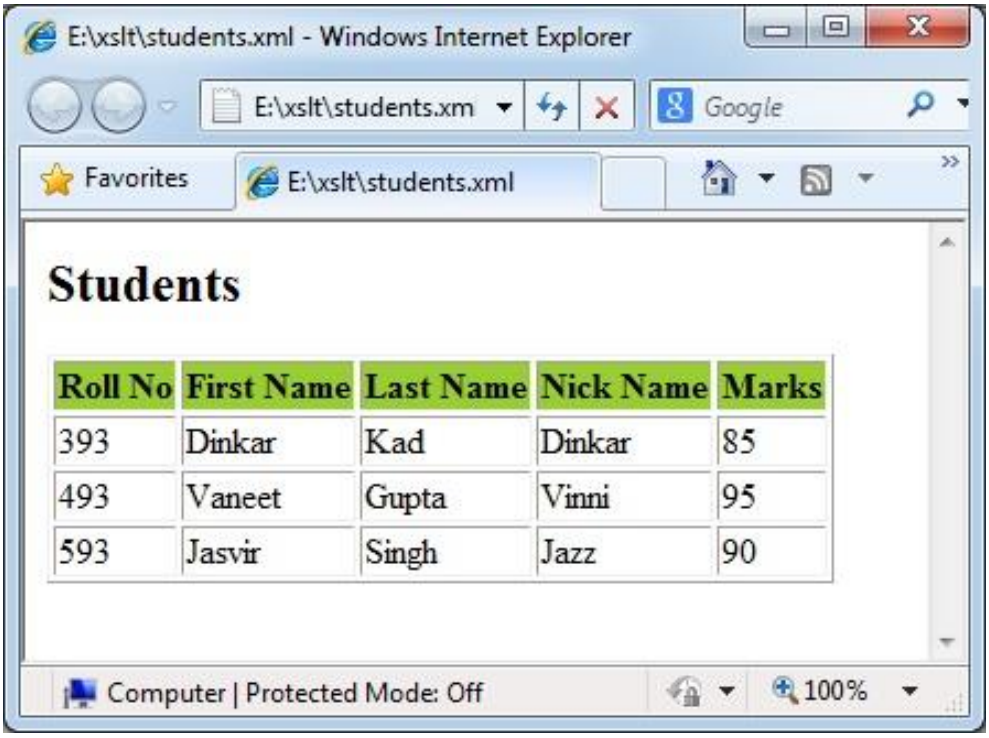
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:import href="students.xsl"/>

<xsl:template match="/">
  <xsl:apply-imports/>
</xsl:template>

</xsl:stylesheet>
```

Output


E:\xslt\students.xml - Windows Internet Explorer
E:\xslt\students.xm
Google
Favorites
E:\xslt\students.xml
Students
Roll No | First Name | Last Name | Nick Name | Marks
393 | Dinkar | Kad | Dinkar | 85
493 | Vaneet | Gupta | Vinni | 95
593 | Jasvir | Singh | Jazz | 90
Computer | Protected Mode: Off
100%