

## Experiment No. 6

**Problem: Implement a program to display XML data using XSL.**

### **Display XML data using XSL**

XSLT (Extensible Stylesheet Language Transformations) is a language for transforming XML documents into other XML documents, or other formats such as HTML for web pages, plain text or XSL Formatting Objects, which may subsequently be converted to other formats, such as PDF, PostScript and PNG.[2] XSLT 1.0 is widely supported in modern web browsers.

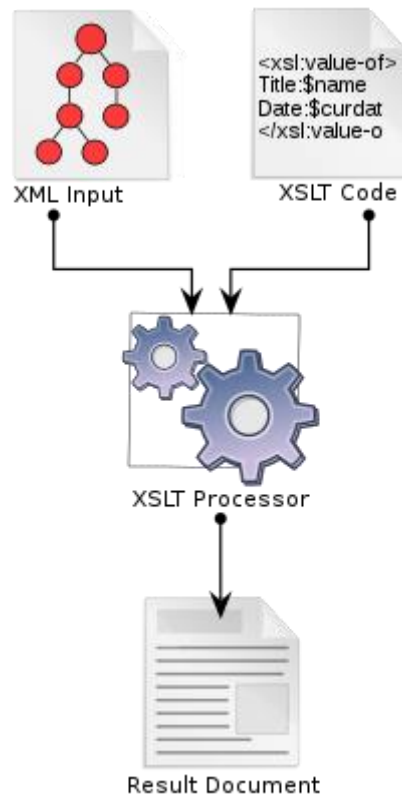
The original document is not changed; rather, a new document is created based on the content of an existing one. Typically, input documents are XML files, but anything from which the processor can build an XQuery and XPath Data Model can be used, such as relational database tables or geographical information systems.

XSLT is designed as a special-purpose language for XML transformation.

### **Design and processing model**

The XSLT processor takes one or more XML source documents, plus one or more XSLT stylesheets, and processes them to produce an output document. In contrast to widely implemented imperative programming languages like C, XSLT is declarative. The basic processing paradigm is pattern matching. Rather than listing an imperative sequence of actions to perform in a stateful environment, template rules only define how to handle a node matching a particular XPath-like pattern, if the processor should happen to encounter one, and the contents of the templates effectively comprise functional expressions that directly represent their evaluated form: the result tree, which is the basis of the processor's output.

A typical processor behaves as follows. First, assuming a stylesheet has already been read and prepared, the processor builds a source tree from the input XML document. It then processes the source tree's root node, finds the best-matching template for that node in the stylesheet, and evaluates the template's contents. Instructions in each template generally direct the processor to either create nodes in the result tree, or to process more nodes in the source tree in the same way as the root node. Finally the result tree is serialized as XML or HTML text.



**Diagram of the basic elements and process flow of eXtensible Stylesheet Language Transformations.**

## **XPath**

XSLT uses XPath to identify subsets of the source document tree and perform calculations. XPath also provides a range of functions, which XSLT itself further augments.

XSLT 1.0 uses XPath 1.0, while XSLT 2.0 uses XPath 2.0. XSLT 3.0 will work with either XPath 3.0 or 3.1. In the case of 1.0 and 2.0, the XSLT and XPath specifications were published on the same date. With 3.0, however, they were no longer synchronized; XPath 3.0 became a Recommendation in April 2014, followed by XPath 3.1 in February 2017; XSLT 3.0 followed in June 2017.

## **How to display XML elements in a HTML table using XSLT?**

XSL (eXtensible Stylesheet Language) is a styling language for XML.

XSLT stands for XSL Transformations.

These examples will teach you how to use XSLT to transform XML documents into other formats (like transforming XML into HTML).

1. Create an XML document.
2. Create an XSL stylesheet.
3. To bind the XML elements to a HTML table, the `<for-each>` XSL template must appear before each table row tag. This ensures that a new row is created for each `<book>` element.
4. The `<value-of>` template will output the selected text of each child element into a separate table.

### A. Example 1

```
<?xml version="1.0"?>
```

```
<?xml-stylesheet type="text/xsl" href="books04.xsl"?>
```

```
<books>
```

```
<book>
```

```
<title>OpenGL Programming Guide Fourth Edition</title>
```

```
<location>107</location>
```

```
<publisher>Addison-Wesley</publisher>
```

```
<year>2004</year>
```

```
</book>
```

```
<book>
```

```
<title>Curves and Surfaces for CAGD: A Practical Guide</title>
```

```
<location>116</location>
```

```
<publisher>Academic Press</publisher>
```

```
<year>2002</year>
```

```
</book>
```

```
<book>
```

```

    <title>An Introduction to NURBS: With Historical Perspective</title>

    <location>120</location>

    <publisher>Academic Press</publisher>

    <year>2001</year>

</book>

<book>

    <title>NURBS: From Projective Geometry to Practical Use</title>

    <location>126</location>

    <publisher>A K Peters</publisher>

    <year>1999</year>

</book>

</books>

```

XSL:

```

<?xml version="1.0"?>

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

<xsl:output method="html" encoding="UTF-8"/>

<xsl:template match="/">

<html>

<head><title>Books</title>

</head>

<body>

<table width="100%" border="1">

<THEAD>

```

```
<TR>

  <TD width="35%"><B>Title</B></TD>

  <TD width="15%"><B>Location</B></TD>

  <TD width="10%"><B>Publisher</B></TD>

  <TD width="10%"><B>Year</B></TD>

</TR>

</THEAD>

<TBODY>

  <xsl:for-each select="books/book">

    <TR>

      <TD width="35%"><xsl:value-of select="title" /></TD>

      <TD width="15%"><xsl:value-of select="location" /></TD>

      <TD width="10%"><xsl:value-of select="publisher" /></TD>

      <TD width="10%"><xsl:value-of select="year" /></TD>

    </TR>

  </xsl:for-each>

</TBODY>

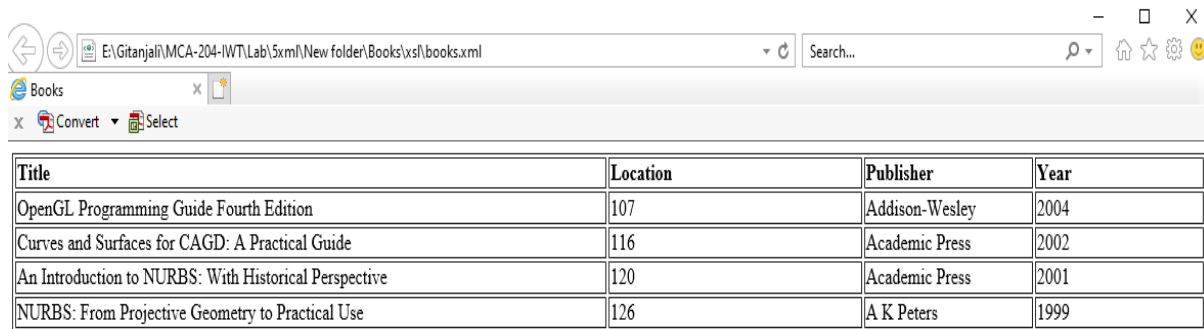
</table>

</body>

</html>

</xsl:template>

</xsl:stylesheet>
```



Title	Location	Publisher	Year
OpenGL Programming Guide Fourth Edition	107	Addison-Wesley	2004
Curves and Surfaces for CAGD: A Practical Guide	116	Academic Press	2002
An Introduction to NURBS: With Historical Perspective	120	Academic Press	2001
NURBS: From Projective Geometry to Practical Use	126	A K Peters	1999

## B. Example 2

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "sports.xsl"?>
<!-- Fig. 14.20: sports.xml -->
<!-- Sports Database -->
<sports>
  <game id = "783">
    <name>Cricket</name>
    <paragraph>
      More popular among commonwealth nations.
    </paragraph>
  </game>

  <game id = "239">
    <name>Baseball</name>
    <paragraph>
      More popular in America.
    </paragraph>
  </game>

  <game id = "418">
    <name>Soccer (Futbol)</name>
    <paragraph>
      Most popular sport in the world.
    </paragraph>
  </game>
</sports>
```

**XML document that describes various sports**

```

<?xml version = "1.0"?>
<!-- Fig. 14.21: sports.xml -->
<!-- A simple XSLT transformation -->

<!-- reference XSL style sheet URI -->
<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

  <xsl:output method = "html" omit-xml-declaration = "no"
    doctype-system =
      "http://www.w3c.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
    doctype-public = "-//W3C//DTD XHTML 1.0 Strict//EN"/>

  <xsl:template match = "/"> <!-- match root element -->

    <html xmlns = "http://www.w3.org/1999/xhtml">
      <head>
        <title>Sports</title>
      </head>

      <body>
        <table border = "1" bgcolor = "wheat">
          <thead>
            <tr>
              <th>ID</th>
              <th>Sport</th>
              <th>Information</th>
            </tr>
          </thead>

          <!-- insert each name and paragraph element value -->
          <!-- into a table row. -->
          <xsl:for-each select = "/sports/game">
            <tr>
              <td><xsl:value-of select = "@id"/></td>
              <td><xsl:value-of select = "name"/></td>
              <td><xsl:value-of select = "paragraph"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>

  </xsl:template>
</xsl:stylesheet>

```

XSLT that creates elements and attributes in an XHTML document

### C. Example 3

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

<!-- Fig. 14.22: sorting.xml -->
<!-- XML document containing book information -->
<book isbn = "999-99999-9-X">
  <title>Deitel&apos;s XML Primer</title>

  <author>
    <firstName>Jane</firstName>
    <lastName>Blue</lastName>
  </author>

  <chapters>
    <frontMatter>
      <preface pages = "2" />
      <contents pages = "5" />
      <illustrations pages = "4" />
    </frontMatter>

    <chapter number = "3" pages = "44">Advanced XML</chapter>
    <chapter number = "2" pages = "35">Intermediate XML</chapter>
    <appendix number = "B" pages = "26">Parsers and Tools</appendix>
    <appendix number = "A" pages = "7">Entities</appendix>
    <chapter number = "1" pages = "28">XML Fundamentals</chapter>
  </chapters>

  <media type = "CD" />
</book>
```

XML document containing book information.



```

<?xml version = "1.0"?>

<!-- Fig. 14.23: sorting.xml -->
<!-- Transformation of book information into XHTML -->
<xsl:stylesheet version = "1.0"
  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">

  <!-- write XML declaration and DOCTYPE DTD information -->
  <xsl:output method = "html" omit-xml-declaration = "no"
    doctype-system = "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"
    doctype-public = "-//W3C//DTD XHTML 1.1//EN"/>

  <!-- match document root -->
  <xsl:template match = "/">
    <html xmlns = "http://www.w3.org/1999/xhtml">
      <xsl:apply-templates/>
    </html>
  </xsl:template>

  <!-- match book -->
  <xsl:template match = "book">
    <head>
      <title>ISBN <xsl:value-of select = "@isbn"/> -
        <xsl:value-of select = "title"/></title>
    </head>

    <body>
      <h1 style = "color: blue"><xsl:value-of select = "title"/></h1>
      <h2 style = "color: blue">by
        <xsl:value-of select = "author/lastName"/>,
        <xsl:value-of select = "author/firstName"/></h2>

      <table style = "border-style: groove; background-color: wheat">

        <xsl:for-each select = "chapters/frontMatter/*">
          <tr>
            <td style = "text-align: right">
              <xsl:value-of select = "name()"/>
            </td>

            <td>
              ( <xsl:value-of select = "@pages"/> pages )
            </td>
          </tr>
        </xsl:for-each>

        <xsl:for-each select = "chapters/chapter">
          <xsl:sort select = "@number" data-type = "number"
            order = "ascending"/>
          <tr>
            <td style = "text-align: right">
              Chapter <xsl:value-of select = "@number"/>
            </td>

```

```

        <td>
            <xsl:value-of select = "text()"/>
            ( <xsl:value-of select = "@pages"/> pages )
        </td>
    </tr>
</xsl:for-each>

<xsl:for-each select = "chapters/appendix">
    <xsl:sort select = "@number" data-type = "text"
        order = "ascending"/>
    <tr>
        <td style = "text-align: right">
            Appendix <xsl:value-of select = "@number"/>
        </td>

        <td>
            <xsl:value-of select = "text()"/>
            ( <xsl:value-of select = "@pages"/> pages )
        </td>
    </tr>
</xsl:for-each>
</table>

<br /><p style = "color: blue">Pages:
    <xsl:variable name = "pagecount"
        select = "sum(chapters//*/@pages)"/>
    <xsl:value-of select = "$pagecount"/>
<br />Media Type: <xsl:value-of select = "media/@type"/></p>
</body>
</xsl:template>
</xsl:stylesheet>

```



XSL document that transforms sorting.xml into XHTML.

#### D. Example 4

```

<?xml version="1.0" ?>
<persons>
    <person username="JS1">
        <name>John</name>
        <family-name>Smith</family-name>
    </person>
    <person username="MI1">
        <name>Morka</name>
    </person>
</persons>

```

```

    <family-name>Ismincius</family-name>
  </person>
</persons>

```

```

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

  <xsl:template match="/persons">
    <root>
      <xsl:apply-templates select="person"/>
    </root>
  </xsl:template>

  <xsl:template match="person">
    <name username="{@username}">
      <xsl:value-of select="name" />
    </name>
  </xsl:template>
</xsl:stylesheet>

```

Its evaluation results in a new XML document, having another structure:

```

<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name username="JS1">John</name>
  <name username="MI1">Morka</name>
</root>

```

Processing the following example XSLT file

```

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

<xsl:stylesheet

version="1.0"

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

xmlns="http://www.w3.org/1999/xhtml">

```

```
<xsl:output method="xml" indent="yes" encoding="UTF-8"/>
```

```
<xsl:template match="/persons">
```

```
  <html>
```

```
    <head> <title>Testing XML Example</title> </head>
```

```
    <body>
```

```
      <h1>Persons</h1>
```

```
      <ul>
```

```
        <xsl:apply-templates select="person">
```

```
          <xsl:sort select="family-name" />
```

```
        </xsl:apply-templates>
```

```
      </ul>
```

```
    </body>
```

```
  </html>
```

```
</xsl:template>
```

```
<xsl:template match="person">
```

```
  <li>
```

```
    <xsl:value-of          select="family-name"/><xsl:text>,      </xsl:text><xsl:value-of
select="name"/>
```

```
  </li>
```

```
</xsl:template>
```

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

with the XML input file shown above results in the following XHTML (whitespace has been adjusted here for clarity):

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

<html xmlns="http://www.w3.org/1999/xhtml">

  <head> <title>Testing XML Example</title> </head>

  <body>

    <h1>Persons</h1>

    <ul>

      <li>Ismincius, Morka</li>

      <li>Smith, John</li>

    </ul>

  </body>

</html>
```

**This XHTML generates the output below when rendered in a web browser:**

# Persons

- Ismincius, Morka
- Smith, John

Rendered XHTML generated from an XML input file and an XSLT transformation.

In order for a web browser to be able to apply an XSL transformation to an XML document on display, an XML stylesheet processing instruction can be inserted into XML. So, for example, if the stylesheet in Example 2 above were available as "example2.xsl", the following instruction could be added to the original incoming XML:

```
<?xml-stylesheet href="example2.xsl" type="text/xsl" ?>
```

In this example, text/xsl is technically incorrect according to the W3C specifications (which say the type should be application/xslt+xml), but it is the only media type that is widely supported across browsers as of 2009.

## **Important Tasks:**

Transforming an XML document using XSLT

Locating Data in XML Documents with XPath

Traversing an XML document using the XML DOM