

This courseware and the data contained herein is the property of eXcelon Corporation., located at 25 Mall Road, Burlington, MA 01803, and its licensors. Any use, disclosure, reproduction, modification, display or transfer of the data and/or courseware is prohibited, except by the express written authorization of eXcelon Corporation. This courseware contains proprietary eXcelon Corporation, Inc., information and is licensed for use pursuant to eXcelon Corporation's standard software license agreement.

If you are an entity of the U.S. government, you agree that the program(s) and data referred to in this courseware are Commercial Computer Software, as defined in the F.A.R., and the DoD F.A.R. Suppl., and are delivered with only those rights set forth in eXcelon Corporation's standard software license agreement.

The program(s) referred to in this courseware are not specifically developed, or licensed, for use in any nuclear, aviation, mass transit, or medical application or in any other inherently dangerous applications, and any such use shall be construed as a misuse of the program(s). eXcelon Corporation. shall not be liable for any claims or damages arising from such use of the program(s) for any such applications.

Data contained herein are proprietary to eXcelon Corporation , or its licensors, and may not be used, disclosed, reproduced, modified, performed or displayed without the prior written approval of eXcelon Corporation.

Copyright (c) 1989-2000 by eXcelon Corporation, Inc. All rights reserved. eXcelon Corporation, ObjectStore, Leadership by Design, Object Exchange, and the eXcelon logo are registered trademarks of eXcelon Corporation. Cache-Forward, ObjectForms, Object Manager, eXcelon and Javlin are trademarks of eXcelon Corporation. All other trademarks are the property of their respective owners.

THE INFORMATION IN THIS COURSEWARE IS SUBJECT TO CHANGE WITHOUT NOTICE. EXCELON CORPORATION MAKES NO WARRANTY OF ANY KIND REGARDING THIS MATERIAL AND, ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS COURSEWARE.

eXcelon Corporation •

The logo for eXcelon Corporation, featuring the word "eXcelon" in a bold, red, sans-serif font, with a trademark symbol (TM) to the upper right of the "n". The word "corp." follows in a smaller, red, sans-serif font.

Twenty Five Mall Road • Burlington, MA 01803 • [www.exceloncorp.com](http://www.exceloncorp.com)



## Table of Contents

|                                |   |
|--------------------------------|---|
| Module 1: Introduction to XSLT | 2 |
| 1-1: What Is XSLT?             | 2 |
| 1-2: What Does XSLT Look Like? | 3 |
| 1-3: How Does XSLT Work?       | 6 |
| 1-4: How Is XSLT Being Used?   | 7 |
| 1-5: XSL Tools                 | 8 |
| 1-6: Invoking XSLT Engines     | 9 |

# 1. Introduction to XSLT

|                                      |   |
|--------------------------------------|---|
| Topic 1-1: What Is XSLT?             | 2 |
| Topic 1-2: What Does XSLT Look Like? | 3 |
| Topic 1-3: How Does XSLT Work?       | 6 |
| Topic 1-4: How Is XSLT Being Used?   | 7 |
| Topic 1-5: XSL Tools                 | 8 |
| Topic 1-6: Invoking XSLT Engines     | 9 |

|                             |  |
|-----------------------------|--|
| Objective                   | <p>At the end of this Module, you will:</p> <ul style="list-style-type: none"> <li>• Explain the relationship of XSLT to other W3C standards</li> <li>• Identify the basic components of an XSLT</li> <li>• Describe the XSLTs processing model</li> <li>• List XSLT related tools</li> <li>• Explain three architectures in which XSLT is employed</li> </ul>   |
| References                  | <p>Chapters 1 and 10 of <u>XSLT</u>, Michael Kay, Wrox Publishing, 2000</p> <p><a href="http://www.xslt.com">http://www.xslt.com</a></p> <p><a href="http://www.xml.com">http://www.xml.com</a></p> <p><a href="http://www.w3.org/TR/XSLT">http://www.w3.org/TR/XSLT</a></p> <p><a href="http://ibm.com/developer/xml">http://ibm.com/developer/xml</a></p> <p><a href="http://www.sun.com/software/xml/developers">http://www.sun.com/software/xml/developers</a></p> <p><a href="http://www.oasis-open.org/cover/xsl.html">http://www.oasis-open.org/cover/xsl.html</a></p>  |
| Topic 1-1:<br>What Is XSLT? | <p><i>XSL (Extensible Stylesheet Language)</i> is an XML-based language used to process XML data. XML data processing can be divided into two categories:</p> <ul style="list-style-type: none"> <li>• Transformation</li> <li>• Formatting</li> </ul> <p><i>XSLT (XSL-Transformation)</i> is a W3C Recommendation. It is used to transform XML documents to other text-based formats, including text, HTML, and XML. A transformation could be as simple as rearranging the elements of an XML document, or as complicated as creating a linked set of HTML files with embedded scripting. Most of this course will focus on XML-to-HTML transformations.</p> <p><i>XSL-FO (XSL-Formatting Objects)</i> is a W3C Recommendation that specifies how to turn XML data into a set of 'formatting objects' (which is just an XML dialect) for print production. Formatting objects are just what they sound like; a set of objects on a page that specify information such as font pitch, font family, margins, gutter widths, etc. XSL-FO has not been implemented as widely as XSLT, partially because of need, and partially because it is a more recent standard.</p> |

*XPath* is a W3C Recommendation that grew out of the W3C's XSLT and XPointer[1] activities. Simply stated, XPath provides a standardized way of pointing to parts of an XML document. These 'parts' of a document are known as nodes. XSL relies heavily on XPath, as we will see in this course.

*DOM (Document Object Model)* is another W3C Recommendation. It provides a standard set of interfaces that programmers used to manipulate XML data. DOM tools will load an entire XML document into a tree representation in memory, and then provide methods to access the tree nodes. Many vendors provide APIs that conform to the DOM specification. You can find DOM implementations in Java, C++, python, and many other languages. We will use some utilities in this course that employ DOM. We will not spend much time discussing the details of DOM, as this can be a varied and complex topic.

*SAX (Simple API for XML)* provides a stream based interface into XML. It accepts XML in streams and reacts based on 'events' (a particular element, a node of a particular type, etc.). We will not spend a great deal of time on SAX in this course.

---

## Topic 1-2: What Does XSLT Look Like?

Here is a very simple XSLT file

### Example 1: XSLT stylesheet

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

<xsl:template match="*" /><xsl:apply-templates/></xsl:template>

<xsl:template match="text()|@"><xsl:value-of
select="."/></xsl:template>

<xsl:template match="bookstore">
  <html><div>
    <h1>Jim Bob's Bookstore</h1>
    <hr/>
    <table STYLE="font:10pt. Verdana" border="2">
      <tr>
        <th>Author</th>
        <th>Title</th>
        <th>Year</th>
        <th>Price</th></tr>

        <xsl:apply-templates select="book"/>
      </table>
    </div></html>
</xsl:template>
```

```

<xsl:template match="book">
  <tr>
<xsl:for-each select="/*">
  <td><xsl:apply-templates select="."/></td>
</xsl:for-each>
  </tr>
</xsl:template>

</xsl:stylesheet>

```

First, notice that an XSLT is just an XML document. This will prove to be convenient at times and frustrating at other times. You can actually look at the XML DTD and/or Schema that describes the structure of an XSLT.

Second, notice that this file is just comprised of templates ( `xsl:template`). Templates are simply a set of tags, instructions, and text that are instantiated upon some processing event. Usually, the event will be the occurrence of an element. For instance, the third template is instantiated when the processor encounters a bookstore element. In this case, we simply have some HTML tags, some static content (see the `th` tags), and some other XSLT instructions (any element with the `xsl:` namespace).

Finally, let's look at the output. If we provide this stylesheet along with the following XML document, we will get HTML that, when loaded into a browser, will appear as below.

## Example 2: XML Source

```

<?xml version="1.0"?>
<bookstore>
  <book>
    <author>W. Shakespeare</author>
    <title>Hamlet</title>
    <published>1997</published>
    <price>2.95</price>
  </book>
  <book>
    <author>W. Shakespeare</author>
    <title>Macbeth</title>
    <published>1989</published>
    <price>9.95</price>
  </book>
  <book>
    <author>D. Alighieri</author>
    <title>The Divine Comedy</title>
    <published>1321</published>
  </book>
</bookstore>

```

```

        <price>5.95</price>
    </book>
</bookstore>

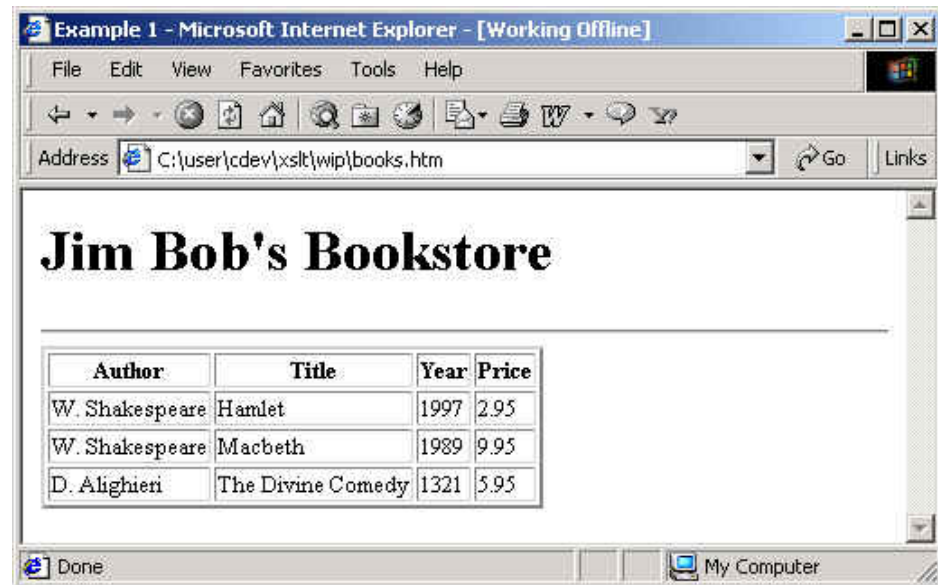
```

### Example 3: HTML Output

```

<html>
<div>
<h1>Jim Bob's Bookstore</h1>
<hr>
<table STYLE="font:10pt. Verdana" border="2">
<tr>
<th>Author</th><th>Title</th><th>Year</th><th>Price</th>
</tr>
<tr>
<td>W. Shakespeare</td><td>Hamlet</td><td>1997</td><td>2.95</td>
</tr>
<tr>
<td>W. Shakespeare</td><td>Macbeth</td><td>1989</td><td>9.95</td>
</tr>
<tr>
<td>D. Alighieri</td><td>The Divine
Comedy</td><td>1321</td><td>5.95</td>
</tr>
</table>
</div>
</html>

```



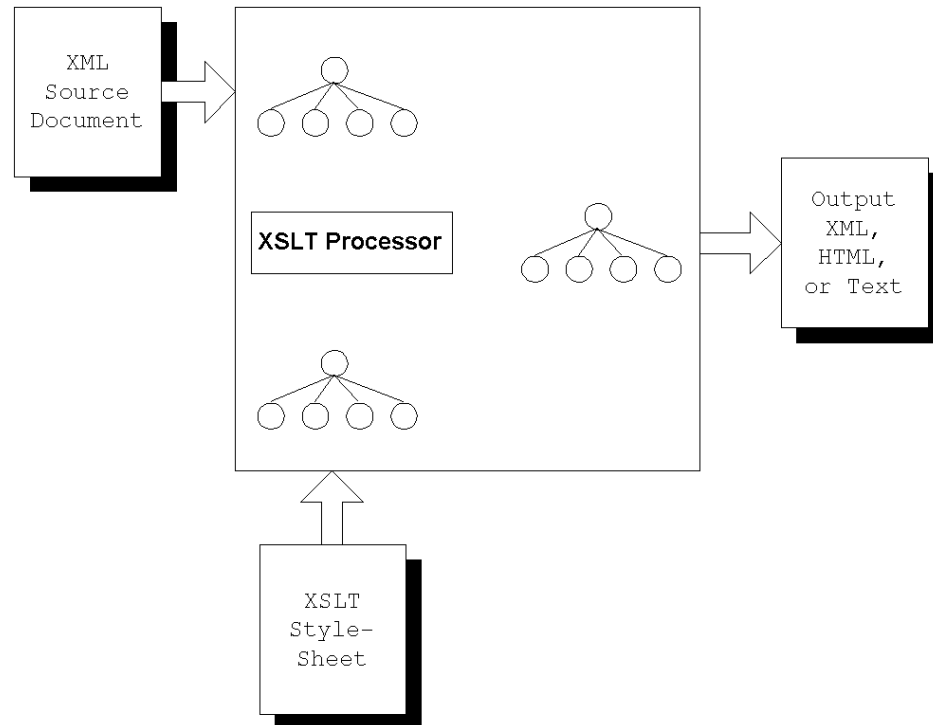
---

Topic 1-3:  
How Does XSLT  
Work?

XML is a wonderful way to describe data, but it requires processing to be anything useful. There are a number of tools that can manipulate, manage, format, or process XML data. XSLT processors are easy to come by these days, and because XSLT is standardized, we can make certain assumptions about any XSLT processor.

XSLT processing is fairly simple.





When you work with a stylesheet, three documents are involved:

- XML source document
- Result document (HTML, XML, or text)
- XSL stylesheet (also an XML document)

When the processor receives a request, it expects to have a stylesheet and some XML data handed to it. The very first thing that the processor does is turn both of these documents into a tree structure in memory (similar to, but not exactly like a DOM tree).

The processor then begins to 'walk' the XML source tree. As it encounters nodes in the tree, it looks for corresponding templates in the XSLT tree. When it finds a match, it performs the specified action; namely, placing content (tags, source data, and static data) into the output tree. When the entire source tree has been processed, the processor will generate an output file based on the constructed output tree.

---

#### Topic 1-4: How Is XSLT Being Used?

There is a lot of excitement and hoopla surrounding the use of XML as the 'lingua franca' of the Internet. XSLT is part of the reason for that, and there are in fact many interesting uses of XML today.

#### **Multiple outputs**

Many companies are realizing their investment in XML by providing content to customers when and where they want it. Because XML is a rich data source, it is generally easy to derive other formats from it. Obviously, there is HTML. But

consider the many other targets of XML data. WML (Wireless Markup Language) is part of a specification that provides content to handheld devices. As the buzz surrounding the wireless web increases, WML will no doubt be an important target for XML data. XSLT, of course, is ideal for this sort of problem.

### ***Business Document Translation***

Another hot topic is online trading communities and inter business process integration. The foundation on which these ideas rest is the sharing of data. As a result, hundreds, perhaps thousands of XML dialects have surfaced that allow companies to exchange information on anything from purchase orders to chemical formulas. Ironically (and perhaps predictably) organizations do not always agree which standard to adopt for cross enterprise integration. This is another situation where XSLT can be very helpful. XSLT can act as the translator from one XML dialect to another.

### ***Document Production***

XML's roots are in SGML (Standard Generalized Markup Language). SGML was the international standard that for years was used in complex publishing systems. XML is now frequently being used in these types of systems. XSLT is again a helpful utility in transforming data into other formats for distribution.

There are undoubtedly many other uses of XSLT in production today.

---

## Topic 1-5: XSL Tools

### ***XSLT Engines***

XSLT engines can perform in a number of situations. Many times, they act as stand alone programs that are invoked on the command line, or in a window. Other times, they are part of an API call.

*XT* is one of, if not the most complete XSLT engine. Created by James Clark ( [www.jclark.com](http://www.jclark.com)

), this engine is available as an executable, or it can be implemented as a servlet.

*Xalan* a product of the Apache Open Source Software Project. It is distributed in Java or C++.

*MSXML* is Microsoft's implementation of XSLT. Actually, it is an entire XML processor that includes an XSLT engine. It is used in Microsoft Internet Explorer to render XML. It also can be invoked via an MSXML DOM call, as we will soon see.

*Sun's XSLT compiler* XSLT is often invoked by a client request over the internet. If there is significant traffic, loading a stylesheet for each request can be resource consuming. As a result, Sun Microsystems has developed a tool that 'compiles' XSLTs into a java class, known as translets. These translets are

much more efficient.

*eXcelon Stylus* is actually an entire XSLT editing environment. It provides a three pane view of source data, XSLT templates, and real time browser output. It has built in knowledge of XSLT and guides you through the XSLT development process.

---

#### Topic 1-6: Invoking XSLT Engines

As the adoption of XSLT as a core technology spreads, there will be more demands on how and where an XSLT engine is invoked. Today, there are three primary methods:

- Command Line Executables
- API Calls
- In-Browser Processing

#### *Processing on the Command Line*

In production, XSLT engines generally are not run as command line executables, however, these tools can be quite useful for development, testing, and debugging.

**XT** may be run via the command line. The syntax is quite simple:

Example 4: XT on the Command Line

```
xt source.xml stylesheet.xsl result.htm param=value
```

**Xalan** has an optional toolkit that you can download. Included in that toolkit is an executable that invokes the XSLT engine.

Example 5: Xalan on the Command Line

```
xalanxslt source.xml stylesheet.xsl param=value
```

#### *API Calls*

This is the most common way to invoke XSLT engines in production. The engine could be called from any type of custom application, including a web servlet, JSP, ASP, or JavaScript. XT, Xalan, MSXML3, and many other engines may be invoked via API call. We will see some examples of these API calls later in the course. Until then, please refer to product specific documentation.

Depending on the environment and needs of an application, API calls can be

made on the client or the server side.

### *In-Browser Processing*

Microsoft Internet Explorer has built in support for XSLT. When an XML file is sent to IE with a specific XML Processing Instruction, the browser applies the stylesheet to the source XML and displays the result. Here is an example of an XML file with the Processing Instruction for the browser:

#### Example 6: IE Processing Instruction

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

<!-- ==== This tells IE about the Stylesheet ==== -->
<?xml-stylesheet type="text/xsl" href="books.xsl"?>
<!-- ===== -->

<bookstore>
  <book>
    <author>W. Shakespeare</author>
    <title>Hamlet</title>
    <published>1997</published>
    <price>2.95</price>
  </book>
  <book>
    <author>W. Shakespeare</author>
    <title>Macbeth</title>
    <published>1989</published>
    <price>9.95</price>
  </book>
  <book>
    <author>D. Alighieri</author>
    <title>The Divine Comedy</title>
    <published>1321</published>
    <price>5.95</price>
  </book>
</bookstore>
```

Netscape has recently increased its support for XML in its latest release (Netscape 6).

---

#### Questions

1. What does the 'T' in XSLT stand for?
2. XML processing can be divided into what two broad categories?
3. What is the purpose of XPath?
4. Is an XSLT stylesheet an XML document? Do you think that this is good or bad?

5. Refer to Example 1. How many templates are in this stylesheet?
6. What is the first thing that an XSLT engine does?
7. What triggers the instantiation of a template?
8. How will you use XSLT?
9. Xalan is an example of an XSLT \_\_\_\_\_?

---

|          |  |
|----------|--|
| Activity | <p>In this activity, you will invoke an XSLT processor in three different ways:</p> <ol style="list-style-type: none"> <li>1. On the command line</li> <li>2. From an API call</li> <li>3. Directly in the browser</li> </ol> <p>Lab Directory:</p> <pre>\$lab_home\01overview</pre> <p>Files:</p> <p><code>books.xml</code> This is the source XML file.</p> <p><code>books.xsl</code> This stylesheet transforms the booklist into an HTML table.</p> <p><code>run-ie.htm</code> This HTML file makes DOM calls using the MSXML3 parser via JavaScript.</p> <p>Complete the following:</p> <ol style="list-style-type: none"> <li>1. Apply <code>books.xsl</code> to <code>books.xml</code> by typing in the following command:<br/> <pre>\$xt_home\xt books.xml books.xsl xt-out.htm</pre> View the HTML using Internet Explorer.</li> <li>2. Open <code>run-ie.htm</code> in Internet Explorer. Examine the source HTML.<br/> <b>Note:</b> This will only work properly if you have <code>msxml3.dll</code> properly installed in <i>replace</i> mode. See installation instructions for more details.</li> <li>3. Apply <code>books.xsl</code> to <code>books.xml</code> using Internet Explorer. You will have to add the appropriate Processing Instruction to the XML file. View the Source from the browser.</li> </ol> |
|----------|--|

---

|         |  |
|---------|--|
| Summary | <p>In this Module, we covered the landscape of XSL, including:</p> <ol style="list-style-type: none"> <li>1. Terms: XSL, XSLT, XSL-FO, XPath, and DOM.</li> <li>2. XSLT components: Stylsheets are usually made up of templates.</li> <li>3. XSLT uses: multiple outputs, business document exchange, document production.</li> <li>4. XSLT tools, including engines and editors.</li> </ol> |
|---------|--|

---