Introduction to XSLT

XSL Languages

It started with XSL and ended up with XSLT, XPath, and XSL-FO.

1   It Started with XSL

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.

2   CSS - HTML Style Sheets

HTML uses predefined tags and the meanings of tags are **well understood**.

The <table> element defines a table and a browser knows **how to display it**.

Adding styles to HTML elements is also simple. Telling a browser to display an element in a special font or color, is easily done with CSS.

3   XSL - XML Style Sheets

XML does not use predefined tags (we can use any tags we like) and the meanings of these tags are **not well understood**.

The <table> could mean an HTML table or a piece of furniture, and a browser **does not know how to display it**.

There must be something in addition to the XML document that describes how the document should be displayed; and that is XSL!

4   XSL - More Than a Style Sheet Language

XSL consists of three parts:

- XSLT is a language for transforming XML documents
- XPath is a language for defining parts of an XML document
- XSL-FO is a language for formatting XML documents

Think of XSL as a set of languages that can **transform** XML into XHTML, **filter and sort** XML data, **define parts** of an XML document, **format** XML data based on the data value, like displaying negative numbers in red, and **output** XML data to different media, like screens, paper, or voice.

5   This Tutorial is About XSLT

The rest of this tutorial is about XSLT - the language for transforming XML documents.

XSLT is a language for transforming XML documents into other XML documents.

XPath is a language for defining parts of an XML document.

5.1   What You Should Already Know

Before you study XSLT you should have a basic understanding of XML and XML Namespaces.

If you want to study these subjects first, please read our XML Tutorial.

5.2   XSLT - XSL Transformations

XSLT is the most important part of the XSL Standards. It is the part of XSL that is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.

XSLT can also add new elements into the output file, or remove elements. It can rearrange and sort elements, and test and make decisions about which elements to display, and a lot more.

A common way to describe the transformation process is to say that XSLT transforms an XML **source tree** into an XML **result tree.**

5.3   XSLT Uses XPath

XSLT uses XPath to define the matching patterns for transformations.

If you want to study XPath first, please read our XPath Tutorial.

5.4   How does it Work?

In the transformation process, XSLT uses XPath to define parts of the source document that **match** one or more predefined **templates**. When a match is found, XSLT will **transform** the matching part of the **source** document into the **result** document. The parts of the source document that do not match a template will end up unmodified in the result document.

5.5   XSLT is a Web Standard

XSLT became a W3C Recommendation 16. November 1999.

XSLT Browsers

Not all Internet browsers have full support for XSLT.

5.6   Internet Explorer 5 is Bad

XSLT in Internet Explorer 5 is NOT compatible with the official W3C XSL Recommendation.

When Internet Explorer 5.0 was released in March 1999, the XSLT standard was still a W3C Working Draft.

Since the final W3C XSL Recommendation is different from the Working Draft, the support for XSL in IE 5 is not 100% compatible with the official XSLT Recommendation.

This restriction applies to both IE 5.0 and IE 5.5.

5.7   Internet Explorer 6 is Better

Internet Explorer 6 fully supports the official W3C XSLT Recommendation.

The XML Parser 3.0 - shipped with Internet Explorer 6.0 and Windows XP - is based on both the W3C XSLT 1.0 and the W3C XPath 1.0 Recommendations.

If you are serious about learning XSLT you should upgrade to Internet Explorer 6.0.

5.8   Netscape 6

Netscape 6 isn't fully supporting the official W3C XSLT Recommendation. However, most of the examples in this tutorial will also work in Netscape 6.

5.9   Netscape 7

Netscape 7 supports the official W3C XSLT Recommendation.

5.10 Internet Explorer MSXML Parser

MSXML Parser 2.0 is the XML parser that is shipped with IE 5.0.

MSXML Parser 2.5 is the XML parser that is shipped with Windows 2000 and IE 5.5.

MSXML Parser 3.0 is the XML parser that is shipped with IE 6.0 and Windows XP. According to Microsoft, the MSXML Parser 3.0 is 100% compatible with the official W3C XSLT Recommendation: "MSXML 3.0 offers a significant advancement over MSXML 2.5: server-safe HTTP access, complete implementation of XSLT and XPath, changes to SAX (Simple API for XML), higher conformance with W3C standards, and a number of bug fixes."

For more info: http://msdn.microsoft.com/xml/general/xmlparser.asp

XSLT - Transformation

Example study: How to transform XML into XHTML using XSLT.

The details of this example will be explained in the next chapter.

5.11 Correct Style Sheet Declaration

The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.

Note: <xsl:stylesheet> and <xsl:transform> are completely synonymous and either can be used!

The correct way to declare an XSL style sheet according to the W3C XSLT Recommendation is:

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

or:

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

Note: The xmlns:xsl="http://www.w3.org/1999/XSL/Transform" identifies the official W3C XSL recommendation namespace. If you use this namespace, you must also include the attribute version="1.0".

Note: If you are using IE 6.0 or Netscape 6 you should use one of the codes above.

5.12 Incorrect Style Sheet Declaration

This was the correct way to declare an XSL style sheet according to the W3C XSL Working Draft:

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
```

Note: The declaration above is OUTDATED, but if you are using IE 5 you should use the (incorrect) code above.

5.13 Start with your XML Document

We want to **transform** the following XML document ("cdcatalog.xml") into XHTML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

To view an XML / XSL document in IE 5.0 (and higher) and Netscape 7 you can click on a link, type the URL in the address bar, or double-click on the name of an XML file in a files folder.

To view an XML / XSL document in Netscape 6 you'll have to open the XML file and then right-click in XML file and select "View Page Source".

View XML file

5.14 Create an XSL Style Sheet

Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation **template:**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
    <tr bgcolor="#9acd32">
      <th align="left">Title</th>
      <th align="left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

View XSL file

5.15 Link the XSL Style Sheet to the XML Document

Finally, add an XSL Style Sheet reference to your XML document ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

If you have an XSLT compliant browser it will nicely **transform** your XML into XHTML!

View the result in IE 6 or Netscape 6 and 7

View the result in IE 5

5.16 Example Explained

The details of the example above will be explained in the next chapters!

The <xsl:template> Element

An XSL style sheet consists of a set of rules called templates.

Each <xsl:template> element contains rules to apply when a specified node is matched.

5.17 XSLT uses Templates

The <xsl:template> element contains rules to apply when a specified node is matched.

The **match** attribute is used to **associate** the template with an **XML element**. The match attribute can also be used to define a template for a whole branch of the XML document (i.e. match="/" defines the whole document).

The following XSL Style Sheet contains a template to output the XML CD Catalog from the previous chapter:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <h2>My CD Collection</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Title</th>
       <th>Artist</th>
     </tr>
     <tr>
       <td>.</td>
       <td>.</td>
     </tr>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Since the style sheet is an XML document itself, the document begins with an xml declaration: **<?xml version="1.0" encoding="ISO-8859-1"?>**.

The **<xsl:stylesheet>** tag defines the start of the style sheet.

The **<xsl:template>** tag defines the start of a template. The **match="/"** attribute associates (matches) the template to the root (/) of the XML source document.

The rest of the document contains the template itself, except for the last two lines that defines the end of the template and the end of the style sheet.

The result of the transformation will look (a little disappointing) like this:

5.18 My CD Collection

Title	Artist
.	.

If you have Netscape 6 or IE 5 or higher you can view: [the XML file](#), [the XSL file](#), and [the result](#)

The result from this example was a little disappointing, because no data was copied from the XML document to the output.

In the next chapter you will learn how to use the **<xsl:value-of>** element to select the value of an XML element.

The <xsl:value-of> Element

The <xsl:value-of> element extracts the value of a selected node.

5.19 The <xsl:value-of> Element

The <xsl:value-of> element can be used to select the value of an XML element and add it to the output stream of the transformation:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
 <body>
   <h2>My CD Collection</h2>
   <table border="1">
     <tr bgcolor="#9acd32">
       <th>Title</th>
       <th>Artist</th>
     </tr>
     <tr>
      <td><xsl:value-of select="catalog/cd/title"/></td>
      <td><xsl:value-of select="catalog/cd/artist"/></td>
     </tr>
   </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Note: The value of the required **select** attribute contains an **XPath expression.** It works like navigating a file system where a forward slash (/) selects subdirectories.

5.20 The Result

The result of the transformation will look like this:

5.21 My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan

If you have Netscape 6 or IE 5 or higher you can view the XML file and the XSL file

View the result in IE 6 or Netscape 6 and 7

View the result in IE 5

The result from this example was also a little disappointing, because only one line of data was copied from the XML document to the output.

In the next chapter you will learn how to use the **<xsl:for-each>** element to select the values of several XML elements, and add them to the output.

The <xsl:for-each> Element

The <xsl:for-each> element allows you to do looping in XSLT.

5.22 The <xsl:for-each> Element

The XSL **<xsl:for-each>** element can be used to select every XML element of a specified node set:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Note: The value of the required **select** attribute contains an **XPath expression.** It works like navigating a file system where a forward slash (/) selects subdirectories.

5.23 The Result

The result of the transformation will look like this:

5.24 My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary More
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart

Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T`Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red
Red	The Communards
Unchain my heart	Joe Cocker

If you have Netscape 6 or IE 5 or higher you can view: the XML file and the XSL file

View the result with Netscape 6 or IE 6

View the result with IE 5

5.25 Filtering the Output

We can filter the output from an XML file by adding a criterion to the **select** attribute in the <xsl:for-each> element.

<xsl:for-each select="catalog/cd[artist='Bob Dylan']">

Legal filter operators are:

- = (equal)
- != (not equal)
- < less than
- > greater than

Take a look at the adjusted XSL style sheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
 <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
   <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
   </tr>
   <xsl:for-each select="catalog/cd[artist='Bob Dylan']">
   <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
   </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

5.26 The Result

The result of the transformation will look like this:

5.27 My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan

If you have Netscape 6 or IE 5 or higher you can view: the XML file and the XSL file.

View the result with Netscape 6 or IE 6

View the result with IE 5

The <xsl:sort> Element

The <xsl:sort> element is used to sort the output.

5.28 Where to put the Sort Information

To output the XML file as an XHTML file, and sort it at the same time, simply add a sort element inside the for-each element in your XSL file:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:sort select="artist"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

The **select** attribute indicates what XML element to sort on.

5.29 The Result

The result of the transformation will look like this:

5.30 My CD Collection

Title	Artist
Romanza	Andrea Bocelli
One night only	Bee Gees
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
The very best of	Cat Stevens
Greatest Hits	Dolly Parton
Sylvias Mother	Dr.Hook
Eros	Eros Ramazzotti

Still got the blues	Gary Moore
Unchain my heart	Joe Cocker
Soulsville	Jorn Hoel
For the good times	Kenny Rogers
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
1999 Grammy Nominees	Many
The dock of the bay	Otis Redding
When a man loves a woman	Percy Sledge
Maggie May	Rod Stewart
Stop	Sam Brown
Black angel	Savage Rose
Picture book	Simply Red
Bridge of Spies	T`Pau
Red	The Communards
Private Dancer	Tina Turner
Tupelo Honey	Van Morrison
Big Willie style	Will Smith

If you have Netscape 6 or IE 5 or higher you can view: the XML file and the XSL file.

View the result with Netscape 6 or IE 6

Note: Unable to view the result in IE 5, because the "http://www.w3.org/TR/WD-xsl" namespace does not understand the <xsl:sort> element.

The <xsl:if> Element

The <xsl:if> element contains a template that will be applied only if a specified condition is true.

5.31 Where to put the IF condition

To put a conditional if test against the content of the file, simply add an <xsl:if> element to your XSL document like this:

<xsl:if test="price &gt; 10">
  some output ...
</xsl:if>

The value of the required **test** attribute contains the expression to be evaluated.

Take a look at the adjusted XSL style sheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

The code above only selects the title and artist IF the price of the cd is higher than 10.

5.32 The Result

The result of the transformation will look like this:

5.33 My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Still got the blues	Gary Moore
One night only	Bee Gees
Romanza	Andrea Bocelli
Black Angel	Savage Rose
1999 Grammy Nominees	Many

If you have Netscape 6 or IE 5 or higher you can view: the XML file and the XSL file.

View the result with Netscape 6 or IE 6

Note: Unable to view the result in IE 5, because the "http://www.w3.org/TR/WD-xsl" namespace does not understand the <xsl:if> element.

The <xsl:choose> Element

The <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.

5.34 Where to put the Choose Condition

To insert a conditional choose test against the content of the XML file, simply add the <xsl:choose>, <xsl:when>, and <xsl:otherwise> elements to your XSL document like this:

<xsl:choose>
  <xsl:when test="price &gt; 10">
    ... some code ...
  </xsl:when>
  <xsl:otherwise>
    ... some code ....
  </xsl:otherwise>
</xsl:choose>

Look at the adjusted XSL style sheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <xsl:choose>
          <xsl:when test="price &gt; 10">
            <td bgcolor="#ff00ff">
            <xsl:value-of select="artist"/></td>
          </xsl:when>
          <xsl:otherwise>
            <td><xsl:value-of select="artist"/></td>
          </xsl:otherwise>
        </xsl:choose>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

The code above will add a pink background-color to the artist column WHEN the price of the cd is higher than 10.

5.35 The Result

The result of the transformation will look like this:

5.36 My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T`Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red
Red	The Communards
Unchain my heart	Joe Cocker

If you have Netscape 6 or IE 5 or higher you can view: the XML file and the XSL file.

View the result with Netscape 6 or IE 6

Note: Unable to view the result in IE 5, because the "http://www.w3.org/TR/WD-xsl" namespace does not understand the <xsl:choose> element.

5.37 Another Example

Here is another example that contains several <xsl:when> elements:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <xsl:choose>
          <xsl:when test="price &gt; 10">
            <td bgcolor="#ff00ff">
            <xsl:value-of select="artist"/></td>
          </xsl:when>
          <xsl:when test="price &gt; 9 and price &lt;= 10">
            <td bgcolor="#cccccc">
            <xsl:value-of select="artist"/></td>
          </xsl:when>
          <xsl:otherwise>
            <td><xsl:value-of select="artist"/></td>
          </xsl:otherwise>
        </xsl:choose>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

The code above will add a pink background color to the artist column WHEN the price of the cd is higher than 10, and a grey background-color WHEN the price of the cd is higher than 9 and lower or equal to 10.

5.38 The Result

The result of the transformation will look like this:

5.39 My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore

Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T`Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red
Red	The Communards
Unchain my heart	Joe Cocker

If you have Netscape 6 or IE 5 or higher you can view: the XML file and the XSL file.

View the result with Netscape 6 or IE 6

Note: Unable to view the result in IE 5, because the "http://www.w3.org/TR/WD-xsl" namespace does not understand the <xsl:choose> element.

The \<xsl:apply-templates\> Element

The \<xsl:apply-templates\> element applies a template rule to the current element or to the current element's child nodes.

5.40 The \<xsl:apply-templates\> Element

The \<xsl:apply-templates\> element applies a template rule to the current element or to the current element's child nodes.

If we add a select attribute to the \<xsl:apply-templates\> element it will process only the child element that matches the value of the attribute. We can use the select attribute to specify the order in which the child nodes are processed.

Look at the following XSL style sheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<xsl:apply-templates/>
</body>
</html>
</xsl:template>
<xsl:template match="cd">
<p>
<xsl:apply-templates select="title"/>
<xsl:apply-templates select="artist"/>
</p>
</xsl:template>
<xsl:template match="title">
Title: <span style="color:#ff0000">
<xsl:value-of select="."/></span>
<br />
</xsl:template>
<xsl:template match="artist">
Artist: <span style="color:#00ff00">
<xsl:value-of select="."/></span>
<br />
</xsl:template>
</xsl:stylesheet>
```

5.41 The Result

The result of the transformation will look like this:

5.42 My CD Collection

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Title: Still got the blues
Artist: Gary Moore

Title: Eros
Artist: Eros Ramazzotti

Title: One night only
Artist: Bee Gees

Title: Sylvias Mother
Artist: Dr.Hook

Title: Maggie May
Artist: Rod Stewart

Title: Romanza
Artist: Andrea Bocelli

Title: When a man loves a woman
Artist: Percy Sledge

Title: Black angel
Artist: Savage Rose

Title: 1999 Grammy Nominees
Artist: Many

Title: For the good times
Artist: Kenny Rogers

Title: Big Willie style
Artist: Will Smith

Title: Tupelo Honey
Artist: Van Morrison

Title: Soulsville
Artist: Jorn Hoel

Title: The very best of
Artist: Cat Stevens

Title: Stop
Artist: Sam Brown

Title: Bridge of Spies
Artist: T`Pau

Title: Private Dancer
Artist: Tina Turner

Title: Midt om natten
Artist: Kim Larsen

Title: Pavarotti Gala Concert
Artist: Luciano Pavarotti

Title: The dock of the bay
Artist: Otis Redding

Title: Picture book
Artist: Simply Red

Title: Red
Artist: The Communards

Title: Unchain my heart
Artist: Joe Cocker

If you have Netscape 6 or IE 5 or higher you can view: the XML file and the XSL file.

View the result with Netscape 6 or IE 6

Note: Unable to view the result in IE 5, because the "http://www.w3.org/TR/WD-xsl" namespace does not understand the <xsl:apply-template> element.

XSLT - On the Client

If your browser supports it, XSLT can be used to transform the document to XHTML in your browser.

5.43 A JavaScript Solution

In the previous chapter we explained how XSLT can be used to transform a document from XML to XHTML. We added an XSL style sheet to the XML file and let the browser do the transformation.

Even if this works fine, it is not always desirable to include a style sheet reference in an XML file (i.e. it will not work in a non XSLT aware browser.)

A more versatile solution would be to use a JavaScript to do the XML to XHTML transformation.

By using JavaScript, we can:

- do browser-specific testing
- use different style sheets according to browser and user needs

That's the beauty of XSLT. One of the design goals for XSLT was to make it possible to transform data from one format to another, supporting different browsers and different user needs.

XSLT transformation on the client side is bound to be a major part of the browsers work tasks in the future, as we will see a growth in the specialized browser market (Braille, aural browsers, Web printers, handheld devices, etc.)

5.44 The XML file and the XSL file

Take a new look at the XML document that you saw in the previous chapters:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

If you have Netscape 6 or IE 5 or higher you can view the XML file.

And the accompanying XSL style sheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th align="left">Title</th>
        <th align="left">Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title" /></td>
        <td><xsl:value-of select="artist" /></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

If you have Netscape 6 or IE 5 or higher you can view the XSL file.

Note: Be sure to notice that the XML file does not have a reference to the XSL file.

IMPORTANT: The above sentence indicates that an XML file could be transformed using many different XSL files.

5.45 Transforming XML to XHTML in Your Browser

Here is the source code needed to transform the XML file to XHTML on the client:

```
<html>
<body>
<script type="text/javascript">
// Load XML
var xml = new ActiveXObject("Microsoft.XMLDOM")
xml.async = false
xml.load("cdcatalog.xml")

// Load XSL
var xsl = new ActiveXObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load("cdcatalog.xsl")

// Transform
document.write(xml.transformNode(xsl))
</script>

</body>
</html>
```

Tip: If you don't know how to write JavaScript, you can study our JavaScript tutorial.

The first block of code creates an instance of the Microsoft XML parser (XMLDOM), and loads the XML document into memory. The second block of code creates another instance of the parser and loads the XSL document into memory. The last line of code transforms the XML document using the XSL document, and writes the result to the XHTML document. Nice!

If you have IE 6.0: See how it works.

If you have IE 5.0: See how it works.

XSLT - On the Server

Since not all browsers support XSLT, one solution is to transform the XML to XHTML on the server.

5.46 A Cross Browser Solution

In the previous chapter we explained how XSLT can be used to transform a document from XML to XHTML in the browser. We let a JavaScript use an XML parser to do the transformation. This solution will not work with a browser that doesn't support an XML parser.

To make XML data available to all kinds of browsers, we have to transform the XML document on the SERVER and send it as pure XHTML to the BROWSER.

That's another beauty of XSLT. One of the design goals for XSLT was to make it possible to transform data from one format to another on a server, returning readable data to all kinds of future browsers.

XSLT transformation on the server is bound to be a major part of the Internet Information Server work tasks in the future, as we will see a growth in the specialized browser market (Braille, aural browsers, Web printers, handheld devices, etc.)

5.47 The XML file and the XSLT file

Take a new look at the XML document that you saw in the previous chapters:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

If you have Netscape 6 or IE 5 or higher you can view the XML file.

And the accompanying XSL style sheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th align="left">Title</th>
        <th align="left">Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title" /></td>
        <td><xsl:value-of select="artist" /></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

If you have Netscape 6 or IE 5 or higher you can view the XSL file.
Note: Be sure that the XML file does not have a reference to the XSL file.
IMPORTANT: The above sentence indicates that an XML file on the server could be transformed using many different XSL files.

5.48 Transforming XML to XHTML on the Server

Here is the source code needed to transform the XML file to XHTML on the server:

```
<%
'Load XML
set xml = Server.CreateObject("Microsoft.XMLDOM")
xml.async = false
xml.load(Server.MapPath("cdcatalog.xml"))

'Load XSL
set xsl = Server.CreateObject("Microsoft.XMLDOM")
xsl.async = false
xsl.load(Server.MapPath("cdcatalog.xsl"))

'Transform file
Response.Write(xml.transformNode(xsl))
%>
```

Tip: If you don't know how to write ASP, you can study our ASP tutorial.

The first block of code creates an instance of the Microsoft XML parser (XMLDOM), and loads the XML file into memory. The second block of code creates another instance of the parser and loads the XSL document into memory. The last line of code transforms the XML document using the XSL document, and returns the result to the browser. Nice!

See how it works.

XML Editors

If you are serious about XML, you will benefit from using a professional XML Editor.

5.49 XML is Text Based

XML is a text based markup language.

One great thing about XML is that XML files can be created and edited using a simple text editor like Notepad.

However, when you start working with XML, you will soon find that it is better to edit XML documents using a professional XML editor.

5.50 Why Not Notepad?

Many "hardcore" web developers use Notepad to edit both HTML and XML documents because Notepad is free and simple to use, and personally I often use Notepad for quick editing of simple HTML, CSS, and XML files.

But, if you use Notepad for XML editing, you will soon run into problems.

Notepad does not know that you are writing XML, so it will not be able to assist you. You will create many errors, and as your XML documents grow larger you will lose control.

5.51 Why an XML Editor?

Today XML is an important technology, and everyday we can see XML playing a more and more critical role in new web development.

New development projects use XML based technologies like:

- XML Schema to define XML structures and data types
- XSLT to transform XML data
- SOAP to exchange XML data between applications
- WSDL to describe web services
- RDF to describe web resources
- XPath and XQuery to access XML data
- SMIL to define graphics... and much more

To be able to write XML documents for all your new development projects, you will need an intelligent editor to help you write error free XML documents.

5.52 XML Editors

Good XML editors will help you to write error free XML documents, validate your text against a DTD or a schema, and force you to stick to a valid XML structure.

A good XML editor should be able to:

- Add closing tags to your opening tags automatically
- Force you to write valid XML
- Verify your XML against a DTD
- Verify your XML against a Schema
- Color code your XML syntax

5.53 XMLSPY

At W3Schools we have been using XMLSPY for many years. XMLSPY is our favorite XML editor. These are some of the features we specially like:

- Easy to use
- Syntax coloring
- Automatic tag completion
- Automatic well-formed check
- Easy switching between text view and grid view
- Built in DTD and / or Schema validation
- Built in graphical XML Schema designer
- Powerful conversion utilities
- Database import and export
- Built in templates for most XML document types
- Built in XPath analyzer
- Full SOAP and WSDL capabilities
- Powerful project management

Read more about XMLSPY

XSLT Elements Reference

The XSLT elements from the W3C Recommendation (XSLT Version 1.0).

5.54 XSLT Elements

The links in the "Element" columns point to attributes and more useful information about the specific element.

- **NN**: indicates the earliest version of Netscape that supports the tag
- **IE**: indicates the earliest version of Internet Explorer that supports the tag

Note: All elements supported in IE 5.X may have NON-standard behavior, because IE 5.X was released before XSLT became a W3C Recommendation!

Element	Description	IE	NN
apply-imports	Applies a template rule from an imported style sheet	6.0	
apply-templates	Applies a template rule to the current element or to the current element's child nodes	5.0	6.0
attribute	Adds an attribute	5.0	6.0
attribute-set	Defines a named set of attributes	6.0	6.0
call-template	Calls a named template	6.0	6.0
choose	Used in conjunction with <when> and <otherwise> to express multiple conditional tests	5.0	6.0
comment	Creates a comment node in the result tree	5.0	6.0
copy	Creates a copy of the current node (without child nodes and attributes)	5.0	6.0
copy-of	Creates a copy of the current node (with child nodes and attributes)	6.0	6.0
decimal-format	Defines the characters and symbols to be used when converting numbers into strings, with the format-number() function	6.0	
element	Creates an element node in the output document	5.0	6.0
fallback	Specifies an alternate code to run if the processor does not support an XSLT element	6.0	
for-each	Loops through each node in a specified node set	5.0	6.0
if	Contains a template that will be applied only if a specified condition is true	5.0	6.0
import	Imports the contents of one style sheet into another. **Note:** An imported style sheet has lower precedence than the importing style sheet	6.0	6.0
include	Includes the contents of one style sheet into another. **Note:** An included style sheet has the same precedence as the including style sheet	6.0	6.0
key	Declares a named key that can be used in the style	6.0	6.0

	sheet with the key() function		
message	Writes a message to the output (used to report errors)	6.0	6.0
namespace-alias	Replaces a namespace in the style sheet to a different namespace in the output	6.0	
number	Determines the integer position of the current node and formats a number	6.0	6.0
otherwise	Specifies a default action for the <choose> element	5.0	6.0
output	Defines the format of the output document	6.0	6.0
param	Declares a local or global parameter	6.0	6.0
preserve-space	Defines the elements for which white space should be preserved	6.0	6.0
processing-instruction	Writes a processing instruction to the output	5.0	6.0
sort	Sorts the output	6.0	6.0
strip-space	Defines the elements for which white space should be removed	6.0	6.0
stylesheet	Defines the root element of a style sheet	5.0	6.0
template	Rules to apply when a specified node is matched	5.0	6.0
text	Writes literal text to the output	5.0	6.0
transform	Defines the root element of a style sheet	6.0	6.0
value-of	Extracts the value of a selected node	5.0	6.0
variable	Declares a local or global variable	6.0	6.0
when	Specifies an action for the <choose> element	5.0	6.0
with-param	Defines the value of a parameter to be passed into a template	6.0	6.0

XSLT Functions

XSLT supplies a number of functions.

5.55 XSLT Functions

Name	Description
current()	Returns the current node
document()	Used to access the nodes in an external XML document
element-available()	Tests whether the element specified is supported by the XSLT processor
format-number()	Converts a number into a string
function-available()	Tests whether the function specified is supported by the XSLT processor
generate-id()	Returns a string value that uniquely identifies a specified node
key()	Returns a node-set using the index specified by an <xsl:key> element
system-property()	Returns the value of the system properties
unparsed-entity-uri()	Returns the URI of an unparsed entity