eXtensible Stylesheet Language Transforms (XSLT)

An Introduction

Prof. David Bernstein
James Madison University

Computer Science Department
bernstdh@jmu.edu

XML Basics

- What is it?

  A subset of the Standard Generalized Markup Language (SGML)

  A meta-language ((i.e., it is used to define other languages)

  XML is a way of "marking up" hierarchical data (and papers/articles/books each contain hierarchical data)

- Comparison with HTML:

  HTML is an application of SGML/XML (i.e, a language defined using SGML/XML)

XML Elements with Components

- Syntax:

  < *tag* [*attribute* **="** *value* **"**] ... **>** [*component* ...] **</** *tag* **>**

- An Example:

  <Employee job="Engineer">John Smith</Employee>

- Note:

  Tags and attributes are case-sensitive

XML Elements without Components

- Syntax:

  < *tag* [*attribute* **="** *value* **"**] ... **/>**

- An Example:

  <Caboose id="857-931" type="limited" />

An Example

A Gradebook

```
<?xml version="1.0"?>
<GRADES>
  <STUDENT>
     <NAME>Kim Maschino</NAME>
     <ID>321-00-0021</ID>
     <GRADE>A+</GRADE>
  </STUDENT>
  <STUDENT>
     <NAME>Doug Sideler</NAME>
     <ID>001-31-9998</ID>
     <GRADE>B</GRADE>
  </STUDENT>
  <STUDENT>
     <NAME>Wayne Tromer</NAME>
     <ID>100-23-4167</ID>
     <GRADE>C</GRADE>
  </STUDENT>
</GRADES>
```

A Brief History of XSL

- The developers of XSL were interested in creating a method of defining the formatting/presentation of XML documents
- The original XSL proposal was submitted to the W3C in August of 1997
- In early 1999 it became apparent that it would be better to think of XSL as two distinct components, XSL Transformations (XSLT) and XSL Formatting Objects (XSL-FO)

  XSL-FO is similar in spirit to other types of stylesheets (including CSS and DSSSL)

  XSLT is a declarative language for selecting and processing the nodes in a document

Overview

- XSLT is a Declarative Language:

  A program consists of a set of template rules, each of which describes how a particular element type should be processed

  Rules can appear in any order

- Syntax:

  Rules in an XSLT program are described using XML

  All XSL tags are in the **`xsl`** namespace

Overview (cont.)

- Side-Effects:

  XSLT is free of side-effects

- Input:

  Any XML document

- Output:

  The most common output is HTML

  XML

  Text

A Simple Example

The Input

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="simple.xsl" ?>
<text>I Love JMU</text>
```

A Simple Example (cont.)

When the root element is encountered, we want to build the skeleton of the HTML document.

```
<xsl:template match="/">
  <HTML>
    <BODY>
      <xsl:apply-templates />
    </BODY>
  </HTML>
</xsl:template>
```

A Simple Example (cont.)

When a **`text`** elements is encountered, we want to insert a **P** element into the HTML skeleton, and insert the contents of the **`text`** element in the **P**.

```
<xsl:template match="text">
  <P>
    <xsl:value-of select="." />
  </P>
</xsl:template>
```

A Simple Example (cont.)

The Complete Program

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
                result-ns="html">

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



<xsl:template match="text">
  <P>
    <xsl:value-of select="." />
  </P>
</xsl:template>



</xsl:stylesheet>
```

The `xsl:stylesheet` Element

- Purpose:

  The root element of all XSLT programs

- The Version for IE:

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

- The General Version:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:msxsl="urn:schemas-microsoft-com:xslt"
                xmlns:user="http://jmu.edu/bernstdh"
                version="1.0">

<xsl:output method="html" indent="yes"/>
```

The `xsl:template` Element

- Purpose:

  Defines a template (or rule) for producing output

- Attributes:

  `match` The *pattern* that determines which nodes will be processed by this template.

The `xsl:apply-templates` Element

- Purpose:

  Indicates that the processor should search for and apply matching templates

- Attributes:

  `select` The *pattern* that determines which nodes will be processed. If this attribute is omitted, all nodes will be processed.

The **`xsl:value-of`** Element

- Purpose:

  Returns an XML node

- Attributes:

  `select` The *pattern* that determines which nodes will be processed. If this attribute is omitted, the current node will be returned. If more than one node is matched, only the first will be returned.

An Example

The Input

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

<bibliography>

<reference>
  <author>
    <firstname>C.E.</firstname>
    <lastname>White</lastname>
  </author>
  <author>
    <firstname>D.</firstname>
    <lastname>Bernstein</lastname>
  </author>
  <author>
    <firstname>A.L.</firstname>
    <lastname>Kornhauser</lastname>
  </author>
  <year>2000</year>
  <title>Some Map-Matching Algorithms for Personal Navigation Assistants</title>
  <journal>
  Transportation Research C
  </journal>
</reference>


<reference>
  <author>
    <firstname>D.</firstname>
    <lastname>Bernstein</lastname>
  </author>
  <author>
    <firstname>I.</firstname>
    <lastname>El Sanhouri</lastname>
  </author>
  <year>1999</year>
  <title>The Roles of Driver Information and Congestion Pricing Systems</title>
  <book>
    Behavioral and Network Impacts of Driver Information Systems
    <editor>
```

```
      <firstname>P.</firstname>
      <lastname> Nijkamp</lastname>
    </editor>
    <editor>
      <firstname>R.</firstname>
      <lastname>Emerink</lastname>
    </editor>
    <publisher>John Wiley and Sons</publisher>
    <pages>371-392</pages>
  </book>
</reference>



<reference>
  <author>
    <firstname>J.</firstname>
    <lastname>Padmos</lastname>
  </author>
  <author>
    <firstname>D.</firstname>
    <lastname>Bernstein</lastname>
  </author>
  <year>1997</year>
  <title>Personal Travel Assistants and the World Wide Web</title>
  <journal>
    Transportation Research Record
    <volume>1573</volume>
    <pages>52-56</pages>
  </journal>
</reference>


<reference>
  <author>
    <firstname>R.</firstname>
    <lastname>Guensler</lastname>
  </author>
  <author>
    <firstname>D.</firstname>
    <lastname>Bernstein</lastname>
  </author>
  <year>1996</year>
  <title>Transportation Resources on the Internet</title>
  <journal>
    ITE Journal
    <volume>66</volume>
    <pages>42-47</pages>
  </journal>
</reference>



<reference>
  <author>
    <firstname>I.</firstname>
    <lastname>El Sanhouri</lastname>
  </author>
  <author>
    <firstname>D.</firstname>
    <lastname>Bernstein</lastname>
  </author>
  <year>1994</year>
  <title>Integrating Driver Information Systems with Facility-Based Congestion Pri
```

```
  <journal>
    Transportation Research Record
    <volume> 1450</volume>
    <pages>44-52</pages>
  </journal>
</reference>


<reference>
  <author>
    <firstname>D.</firstname>
    <lastname>Bernstein</lastname>
  </author>
  <author>
    <firstname>A.</firstname>
    <lastname>Kanaan</lastname>
  </author>
  <year>1993</year>
  <title>Automatic Vehicle Identification Technologies and Functionalities</title>
  <journal>
    IVHS Journal
    <volume> 1</volume>
    <pages>191-204</pages>
  </journal>
</reference>

</bibliography>
```

The Program

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
                  result-ns="html">

<!-- reference.xsl v1.0 -->



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




<!-- author -->
<xsl:template match="author">
  <xsl:apply-templates select="firstname"/>
  <xsl:apply-templates select="lastname"/>,
</xsl:template>



<!-- bibliography -->
<xsl:template match="bibliography">
  <xsl:apply-templates />
</xsl:template>
```

```
<!-- book -->
<!-- Note the use of the built-in text() function -->
<!-- that prevents the children from being selected -->
<xsl:template match="book">
  <xsl:value-of select="text()" />,
  <xsl:apply-templates select="editor"/>
  <xsl:apply-templates select="publisher"/>
  <xsl:apply-templates select="pages"/>.
</xsl:template>



<!-- editor -->
<xsl:template match="editor">
  <xsl:apply-templates select="firstname"/>
  <xsl:apply-templates select="lastname"/>,
</xsl:template>




<!-- firstname -->
<xsl:template match="firstname">
  <xsl:value-of select="." />
</xsl:template>






<!-- journal -->
<!-- Note the use of the built-in text() function -->
<!-- that prevents the children from being selected -->
<xsl:template match="journal">
  <xsl:value-of select="text()" />,
  <xsl:apply-templates select="volume"/>
  <xsl:apply-templates select="pages"/>.
</xsl:template>




<!-- lastname -->
<xsl:template match="lastname">
  <xsl:value-of select="." />
</xsl:template>






<!-- pages -->
<xsl:template match="pages">
  pp. <xsl:value-of select="." />
</xsl:template>



<!-- publisher -->
<xsl:template match="publisher">
```

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



<!-- reference -->
<xsl:template match="reference">

    <xsl:apply-templates select="author"/>
    <xsl:apply-templates select="year"/>
    <xsl:apply-templates select="title"/>
    <xsl:apply-templates select="journal"/>
    <xsl:apply-templates select="book"/>

</xsl:template>




<!-- title -->
<xsl:template match="title">
  &quot;<xsl:value-of select="." />&quot;,
</xsl:template>




<!-- volume -->
<xsl:template match="volume">
  Vol. <xsl:value-of select="." />,
</xsl:template>



<!-- year -->
<xsl:template match="year">
  (<xsl:value-of select="." />)
</xsl:template>



</xsl:stylesheet>
```

Path Expressions

- Purpose:

  Describe a sequence of "branches" in a document tree (i.e., a way of "finding" a node in the document tree.

- Operators:

`/`	The *child* operator. It selects elements that are children of the specified node. For example, **`reference/year`** refers to the **`year`** element that is a child of the **`reference`** element.
`.`	The current node operator. For example, **`./year`** refers to the **`year`** element that is a child of the current element.
`@`	The attribute operator. For example, **`train/@number`** refers to the **`number`** attribute of the **`train`** element.
`*`	The wildcard operator. For example, **`timetable/*`** refers to all children of the **`timetable`** element.
`[ ]`	The index operator. For example, **`stop/time[0]`** refers to the first **`time`** element that is a child of the **`stop`** element. As another example, **`stop/time[end()]`** refers to the last **`time`** element that is a child of the **`stop`** element. (**Note:** In more recent versions of XSLT you should use **`last()`** and not **`end()`**.)

Path Expressions: An Example

The Input

```
<?xml version="1.0"?>
<!DOCTYPE timetable SYSTEM "timetable.dtd">

<?xml-stylesheet type="text/xsl"
                 href="/bernstdh/web/common/lectures/xsltexamples/timetable/timeta

<timetable title="Boston...Providence...New London...New York"
     subtitle="Through services to Philadelphia...Washington...Newport News">
  <train number="95" normaldays="Mo-Fr" reservations="YES" businessclass="YES">
    <stop>
      <station>Boston, MA-South Station</station>
      <time>6:20AM</time>
    </stop>

    <stop>
      <station>Boston, MA-Back Bay</station>
      <time>6:25AM</time>
    </stop>

    <stop>
      <station>Route 128, MA</station>
      <time>7:07AM</time>
    </stop>

    <stop>
      <station>Providence, RI</station>
      <time>7:31AM</time>
    </stop>

    <stop>
      <station>Kingston, RI</station>
      <time></time>
```

```
</stop>

<stop>
  <station>Westerly, RI</station>
  <time></time>
</stop>

<stop>
  <station>Mystic, CT</station>
  <time></time>
</stop>

<stop>
  <station>New London, CT</station>
  <time>8:06AM</time>
</stop>

<stop>
  <station>Old Saybrook, CT</station>
  <time></time>
</stop>

<stop>
  <station>New Haven, CT</station>
  <time status="Ar">9:00AM</time>
  <time status="Dp">9:15AM</time>
</stop>

<stop>
  <station>Bridgeport, CT</station>
  <time></time>
</stop>

<stop>
  <station>Stamford, CT</station>
  <time></time>
</stop>

<stop>
  <station>New Rochelle, NY</station>
  <time></time>
</stop>

<stop>
  <station>New York, NY</station>
  <time status="Ar">10:50AM</time>
</stop>

<stop>
  <station>Newark, NJ</station>
  <time>11:26AM</time>
</stop>

<stop>
  <station>Metropark, NJ</station>
  <time>11:40AM</time>
</stop>

<stop>
  <station>Trenton, NJ</station>
  <time>12:04AM</time>
</stop>

<stop>
```

```
    <station>Philadelphia, PA</station>
    <time>12:32PM</time>
  </stop>

  <stop>
    <station>Wilmington, DE</station>
    <time>12:58PM</time>
  </stop>

  <stop>
    <station>Baltimore, MD</station>
    <time>1:49PM</time>
  </stop>

  <stop>
    <station>BWI Airport Rail Sta., MD</station>
    <time>2:02PM</time>
  </stop>

  <stop>
    <station>New Carrollton, MD</station>
    <time>2:17PM</time>
  </stop>

  <stop>
    <station>Washington, DC</station>
    <time>2:35PM</time>
  </stop>

  <stop>
    <station>Richmond, VA</station>
    <time>5:19PM</time>
  </stop>

  <stop>
    <station>Newport News, VA</station>
    <time status="Ar">7:07PM</time>
  </stop>
</train>



<train number="191" normaldays="Sa"  reservations="YES" businessclass="YES">
  <stop>
    <station>Boston, MA-South Station</station>
    <time>6:20AM</time>
  </stop>

  <stop>
    <station>Boston, MA-Back Bay</station>
    <time>6:25AM</time>
  </stop>

  <stop>
    <station>Route 128, MA</station>
    <time>7:07AM</time>
  </stop>

  <stop>
    <station>Providence, RI</station>
    <time>7:31AM</time>
  </stop>
```

```
<stop>
  <station>Kingston, RI</station>
  <time></time>
</stop>

<stop>
  <station>Westerly, RI</station>
  <time></time>
</stop>

<stop>
  <station>Mystic, CT</station>
  <time></time>
</stop>

<stop>
  <station>New London, CT</station>
  <time>8:06AM</time>
</stop>

<stop>
  <station>Old Saybrook, CT</station>
  <time></time>
</stop>

<stop>
  <station>New Haven, CT</station>
  <time status="Ar">9:00AM</time>
  <time status="Dp">9:15AM</time>
</stop>

<stop>
  <station>Bridgeport, CT</station>
  <time></time>
</stop>

<stop>
  <station>Stamford, CT</station>
  <time></time>
</stop>

<stop>
  <station>New Rochelle, NY</station>
  <time></time>
</stop>

<stop>
  <station>New York, NY</station>
  <time status="Ar">10:50AM</time>
</stop>

<stop>
  <station>Newark, NJ</station>
  <time>11:26AM</time>
</stop>

<stop>
  <station>Metropark, NJ</station>
  <time>11:40AM</time>
</stop>

<stop>
  <station>Trenton, NJ</station>
  <time>12:04AM</time>
```

```
  </stop>

  <stop>
    <station>Philadelphia, PA</station>
    <time>12:32PM</time>
  </stop>

  <stop>
    <station>Wilmington, DE</station>
    <time>12:58PM</time>
  </stop>

  <stop>
    <station>Baltimore, MD</station>
    <time>1:49PM</time>
  </stop>

  <stop>
    <station>BWI Airport Rail Sta., MD</station>
    <time>2:02PM</time>
  </stop>

  <stop>
    <station>New Carrollton, MD</station>
    <time>2:17PM</time>
  </stop>

  <stop>
    <station>Washington, DC</station>
    <time>2:35PM</time>
  </stop>

  <stop>
    <station>Richmond, VA</station>
    <time></time>
  </stop>

  <stop>
    <station>Newport News, VA</station>
    <time></time>
  </stop>
</train>



<train number="171" normaldays="Daily" businessclass="YES">
  <stop>
    <station>Boston, MA-South Station</station>
    <time>7:22AM</time>
  </stop>

  <stop>
    <station>Boston, MA-Back Bay</station>
    <time>7:27AM</time>
  </stop>

  <stop>
    <station>Route 128, MA</station>
    <time>7:42AM</time>
  </stop>

  <stop>
    <station>Providence, RI</station>
```

```
    <time>8:17AM</time>
  </stop>

  <stop>
    <station>Kingston, RI</station>
    <time>8:43AM</time>
  </stop>

  <stop>
    <station>Westerly, RI</station>
    <time>8:59AM</time>
  </stop>

  <stop>
    <station>Mystic, CT</station>
    <time>9:10AM</time>
  </stop>

  <stop>
    <station>New London, CT</station>
    <time>9:25AM</time>
  </stop>

  <stop>
    <station>Old Saybrook, CT</station>
    <time>9:47AM</time>
  </stop>

  <stop>
    <station>New Haven, CT</station>
    <time status="Ar">10:25AM</time>
    <time status="Dp">10:35AM</time>
  </stop>

  <stop>
    <station>Bridgeport, CT</station>
    <time>10:57AM</time>
  </stop>

  <stop>
    <station>Stamford, CT</station>
    <time>11:23AM</time>
  </stop>

  <stop>
    <station>New Rochelle, NY</station>
    <time>11:43AM</time>
  </stop>

  <stop>
    <station>New York, NY</station>
    <time status="Ar">12:20PM</time>
  </stop>

  <stop>
    <station>Newark, NJ</station>
    <time>12:56PM</time>
  </stop>

  <stop>
    <station>Metropark, NJ</station>
    <time></time>
  </stop>
```

```
    <stop>
      <station>Trenton, NJ</station>
      <time>1:34PM</time>
    </stop>

    <stop>
      <station>Philadelphia, PA</station>
      <time>2:02PM</time>
    </stop>

    <stop>
      <station>Wilmington, DE</station>
      <time>2:43PM</time>
    </stop>

    <stop>
      <station>Baltimore, MD</station>
      <time>3:42PM</time>
    </stop>

    <stop>
      <station>BWI Airport Rail Sta., MD</station>
      <time>3:55PM</time>
    </stop>

    <stop>
      <station>New Carrollton, MD</station>
      <time>4:10PM</time>
    </stop>

    <stop>
      <station>Washington, DC</station>
      <time>4:30PM</time>
    </stop>

    <stop>
      <station>Richmond, VA</station>
      <time></time>
    </stop>

    <stop>
      <station>Newport News, VA</station>
      <time></time>
    </stop>
  </train>



</timetable>
```

The Program

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
                result-ns="html"
                language="JScript">


  <!-- Root -->
  <xsl:template match="/">
    <HTML>
```

```
    <HEAD>
    </HEAD>

    <BODY BGCOLOR="#FFFFFF">
      <xsl:apply-templates select="timetable"/>
    </BODY>
  </HTML>
</xsl:template>



<!-- Station -->
<xsl:template match="station">
  <B>
    <xsl:value-of select="text()" />
  </B>
</xsl:template>



<!-- Stop -->
<xsl:template match="stop">
  <TD>
    <xsl:apply-templates select="station" />
  </TD>
</xsl:template>




<!-- Time -->
<xsl:template match="time">
  <TD>
    <xsl:value-of select="text()" />
  </TD>
</xsl:template>


<!-- Timetable -->
<!-- Note the use of the attribute operator -->
<xsl:template match="timetable">
  <H2>
    <xsl:value-of select="@title"/>
  </H2>
  <H3>
    <xsl:value-of select="@subtitle"/>
  </H3>
  <TABLE>
    <xsl:apply-templates select="train"/>
  </TABLE>
</xsl:template>




<!-- Train -->
<!-- Note the use of the attribute and child operators -->
<xsl:template match="train">
    <TR>
      <TD><xsl:value-of select="@number" /></TD>
      <TD><xsl:value-of select="@normaldays" /></TD>
      <xsl:apply-templates select="stop/time" />
    </TR>
</xsl:template>
```

```
  <!-- Note the use of the index, attribute, and child operators -->
  <xsl:template match="train[0]">
      <TR>
        <TD><B>Train Number</B></TD>
        <TD><B>Normal Days</B></TD>
        <xsl:apply-templates select="stop" />
      </TR>
      <TR>
        <TD><xsl:value-of select="@number" /></TD>
        <TD><xsl:value-of select="@normaldays" /></TD>
        <xsl:apply-templates select="stop/time" />
      </TR>
  </xsl:template>




</xsl:stylesheet>
```

Filter Operators

- Purpose:

  Allow elements to be selected

- Some Selection Criteria:

  The existence of child nodes

  The value of a node

  The existence of an attribute

  The value of an attribute

- Syntax:

  [*operator pattern*]

  where *operator* denotes a filter operator and *pattern* denotes a filter pattern

Filter Operators (cont.)

= The equal value operator. For example, **`time[@status = 'Ar']`** refers to a **`time`** element with a **`status`** attribute equal to "Ar".

> The greater than value operator.

< The less than value operator.

>= The greater than or equal to value operator.

<= The less than or equal to value operator.

!= The not equal to value operator.

The `xsl:if` Element

- Purpose:

  Conditional processing

- Attributes:

  `match` The *pattern* that determines the value of the test condition.

  `test` Evaluate to *true* if the given element exists

The `xsl:if` Element (cont.)

An Example

```
<xsl:if test="pages">
  ,<xsl:apply-templates select="pages"/>
</xsl:if>
```

The `xsl:choose` Element

- Purpose:

Conditional processing

- Contains:

  One or more **`xsl:when`** elements

  At most one **`xsl:otherwise`** element

- Attributes:

  Like **`xsl:if`**

The **`xsl:choose`** Element (cont.)

An Example

```
<!-- author -->
<!-- v2.0 Fixes first author and last author -->
<xsl:template match="author">
  <xsl:choose>
    <xsl:when match="author[0]">
      <xsl:apply-templates select="lastname"/>,
      <xsl:apply-templates select="firstname"/>
    </xsl:when>
    <xsl:when match="author[end()]">
      and
      <xsl:apply-templates select="firstname"/>
      <xsl:apply-templates select="lastname"/>
    </xsl:when>
    <xsl:otherwise>
      ,
      <xsl:apply-templates select="firstname"/>
      <xsl:apply-templates select="lastname"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

A Complete Example

The Program

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
                  result-ns="html">

<!-- reference.xsl v1.0 -->



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

```
</xsl:template>



<!-- author -->
<xsl:template match="author">
  <xsl:apply-templates select="firstname"/>
  <xsl:apply-templates select="lastname"/>,
</xsl:template>



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



<!-- book -->
<!-- Note the use of the built-in text() function -->
<!-- that prevents the children from being selected -->
<xsl:template match="book">
  <xsl:value-of select="text()" />,
  <xsl:apply-templates select="editor"/>
  <xsl:apply-templates select="publisher"/>
  <xsl:apply-templates select="pages"/>.
</xsl:template>



<!-- editor -->
<xsl:template match="editor">
  <xsl:apply-templates select="firstname"/>
  <xsl:apply-templates select="lastname"/>,
</xsl:template>



<!-- firstname -->
<xsl:template match="firstname">
  <xsl:value-of select="." />
</xsl:template>



<!-- journal -->
<!-- Note the use of the built-in text() function -->
<!-- that prevents the children from being selected -->
<xsl:template match="journal">
  <xsl:value-of select="text()" />,
  <xsl:apply-templates select="volume"/>
  <xsl:apply-templates select="pages"/>.
</xsl:template>
```

```
<!-- lastname -->
<xsl:template match="lastname">
  <xsl:value-of select="." />
</xsl:template>




<!-- pages -->
<xsl:template match="pages">
  pp. <xsl:value-of select="." />
</xsl:template>



<!-- publisher -->
<xsl:template match="publisher">
  <xsl:value-of select="." />,
</xsl:template>




<!-- reference -->
<xsl:template match="reference">

    <xsl:apply-templates select="author"/>
    <xsl:apply-templates select="year"/>
    <xsl:apply-templates select="title"/>
    <xsl:apply-templates select="journal"/>
    <xsl:apply-templates select="book"/>

</xsl:template>




<!-- title -->
<xsl:template match="title">
  &amp;quot;<xsl:value-of select="." />&amp;quot;,
</xsl:template>




<!-- volume -->
<xsl:template match="volume">
  Vol. <xsl:value-of select="." />,
</xsl:template>



<!-- year -->
<xsl:template match="year">
  (<xsl:value-of select="." />)
</xsl:template>
```

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

The Timetable Example with CSS Formatting

For Those Who Are Interested

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl"
                result-ns="html"
                language="JScript">


  <!-- Root -->

  <xsl:template match="/">
    <HTML>
      <HEAD>
      <STYLE>

        SPAN.title   {font-family: Arial; font-size: 16pt;
                      font-style: bold}

        SPAN.subtitle {font-family: Arial; font-size: 14pt;
                      font-style: bold}


        SPAN.trainnumber {font-family: Arial; font-size: 12pt;
                      font-style: bold}

        TABLE    {border-left-style: none; border-right-style: solid;
                  border-top-style: solid; border-bottom-style: solid;
                  border-top-width: 1px; border-bottom-width: 2px;
                  border-right-width: 1px; border-color: black;
                  font-family: Arial; font-size: 8pt;}

        TD       {border-top-style: solid;  border-bottom-style: none;
                  border-right-style: solid;
                  border-width: 1px}

        TD.head  {border-bottom-style: solid;}

        TR       {height: 14pt;}
        TR.trainnumber {height: 18pt;}

      </STYLE>
      </HEAD>

      <BODY BGCOLOR="#FFFFFF">
        <xsl:apply-templates />
      </BODY>
    </HTML>
  </xsl:template>



  <!-- Station -->

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

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




<!-- Stop -->

<xsl:template match="stop">
  <TR>
    <TD ALIGN="LEFT" VALIGN="TOP">
      <xsl:if test="ancestor(train[0])">
        <xsl:apply-templates select="station"/>
      </xsl:if>
    </TD>
    <TD ALIGN="RIGHT" VALIGN="TOP">
      <xsl:apply-templates select="time"/>
    </TD>
  </TR>
</xsl:template>



<!-- Text  -->

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



<!-- Time -->

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



<!-- Timetable -->

<xsl:template match="timetable">
  <SPAN CLASS="title">
    <xsl:value-of select="./@title"/>
  </SPAN>
  <BR />
  <SPAN CLASS="subtitle">
    <xsl:value-of select="./@subtitle"/>
  </SPAN>
  <BR />
  <xsl:apply-templates />
</xsl:template>



<!-- Train -->

<xsl:template match="train">
  <TABLE STYLE="{display:inline}"  CELLPADDING="0" CELLSPACING="0">
```

```
<COLGROUP>
</COLGROUP>
<COLGROUP>
  <xsl:if test="@reservations">
    <xsl:attribute name="STYLE">
       {background-color: orange}
    </xsl:attribute>
  </xsl:if>
</COLGROUP>

<TR CLASS="trainnumber">
  <TD CLASS="head" ALIGN="LEFT">
    <xsl:if match="train[0]">
      <B>Train Number</B>
    </xsl:if>
  </TD>
  <TD CLASS="head" ALIGN="CENTER">
    <SPAN CLASS="trainnumber">
      <xsl:value-of select="./@number" />
    </SPAN>
  </TD>
</TR>

<TR>
  <TD CLASS="head" ALIGN="LEFT">
    <xsl:if match="train[0]">
      <B>Normal Days of Operation</B>
    </xsl:if>
  </TD>
  <TD CLASS="head" ALIGN="CENTER">
    <xsl:value-of select="./@normaldays" />
  </TD>
</TR>

<TR>
  <TD CLASS="head" ALIGN="LEFT">
    <xsl:if match="train[0]">
      <B>   Will Also Operate</B>
    </xsl:if>
  </TD>
  <TD CLASS="head" ALIGN="CENTER">
    <xsl:value-of select="./@willalsooperate" /> 
  </TD>
</TR>

<TR>
  <TD CLASS="head" ALIGN="LEFT">
    <xsl:if match="train[0]">
      <B>   Will Not Operate</B>
    </xsl:if>
  </TD>
  <TD CLASS="head" ALIGN="CENTER">
    <xsl:value-of select="./@willnotoperate" /> 
  </TD>
</TR>

<TR>
  <TD CLASS="head" ALIGN="LEFT">
    <xsl:if match="train[0]">
      <B>On Board Service</B>
    </xsl:if>
  </TD>
  <TD CLASS="head" ALIGN="CENTER">
    <xsl:if test="@businessclass">
```

```
            <IMG SRC="businessclass.gif" />
          </xsl:if>
          <xsl:if test="@reservations">
            <IMG SRC="reservations.gif" />
          </xsl:if>
          <xsl:if test="@sleepingcar">
            <IMG SRC="sleepingcar.gif" />
          </xsl:if>
        </TD>
      </TR>


      <xsl:apply-templates select="stop"/>
    </TABLE>
  </xsl:template>









  <!-- SCRIPTS -->

  <xsl:script><![CDATA[



  ]]></xsl:script>


</xsl:stylesheet>
```