Enterprise Systems and Architecture

CMPU4025

XSL

WORKING WITH XSL

- W3C developed the **Extensible Style sheet Language (XSL)**
- XSL is composed of three parts:
  – XSL-FO (Extensible Style sheet Language – Formatting Objects)
  – **XSLT (Extensible Style sheet Language Transformations)**
  – **XPath**

INTRODUCING XSL-FO, XSLT, AND XPATH

- **XSLT** is used to **transform** XML content from one XML format to another
- **XPath** is used to locate information from an XML document and **perform operations and calculations** upon that content

3

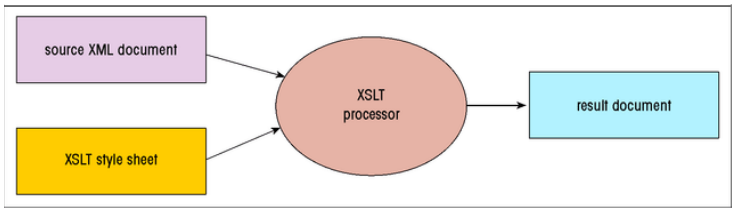
INTRODUCING XSLT STYLE SHEETS AND PROCESSORS

- An XSLT style sheet contains **instructions for transforming** the contents of an XML document into another format
- An **XSLT style sheet** document **is** itself an **XML** document
- An XSLT style sheet document has an extension ***.xsl***

4

GENERATING A RESULT DOCUMENT

- An XSLT style sheet **converts a source document** of XML content **into a result document** by using the **XSLT processor**



5

INTRODUCING XSLT STYLE SHEETS AND PROCESSORS

- The **transformation** can be performed by a **server or a client**
- In a **server-side transformation**, the server receives a request from a client, applies the style sheet to the source document, and returns the result document to the client
- In a **client-side transformation**, a client requests retrieval of both the source document and the style sheet from the server, then performs the transformation, and generates the result document

6

CREATING AN XSLT STYLE SHEET

- To create an XSLT style sheet, the general structure:

```
<?xml version ="1.0"?>
 <xsl:stylesheet version = 1.0 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      Content of the style sheet
</xsl:stylesheet>
```

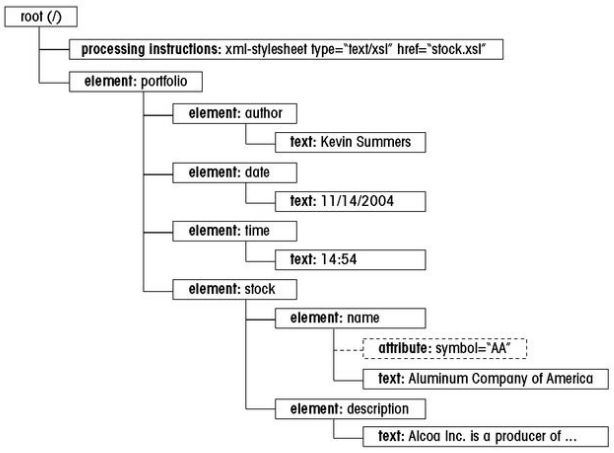
7

WORKING WITH DOCUMENT NODES

- Under *XPath*, each component in the document is referred to as a **node**, and the entire structure of the document is a **node tree**
- The node tree consists of the following objects:
  - the XML document itself
  - Comments
  - Processing Instructions
  - Namespaces
  - Elements
  - Element text
  - Element attributes

8

NODE TREE EXAMPLE



9

WORKING WITH DOCUMENT NODES

- At the **top** of the node tree is the **root node**
- A node that contains other nodes is called a **parent node**, and the nodes contained in the parent are called **child nodes**
- Nodes that share a common parent are called **sibling nodes**
- Any node below another node is referred to as a **descendant** of that node

10

WORKING WITH DOCUMENT NODES

- Nodes are distinguished based on the object they refer to in the document
- A node for an element is called an **element node**
- The node that stores element attributes is called an **attribute node**

11

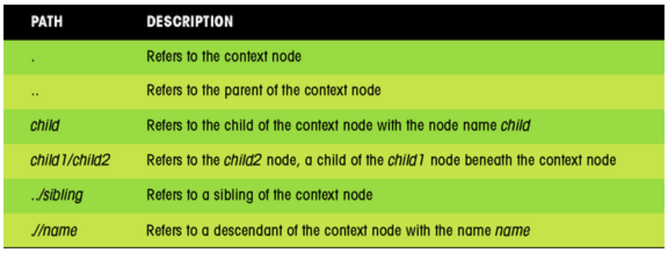
USING XPATH TO REFERENCE A NODE

- **XPath** provides the **syntax** to refer to the various nodes in the node tree
- The **location** of a node can be expressed in either **absolute or relative** terms
- XPath also does **data extraction**

12

RELATIVE PATHS

- With a relative path, the location of the node is indicated relative to a specific node in the tree called the context node



PATH	DESCRIPTION
.	Refers to the context node
..	Refers to the parent of the context node
child	Refers to the child of the context node with the node name *child*
child1/child2	Refers to the *child2* node, a child of the *child1* node beneath the context node
../sibling	Refers to a sibling of the context node
//name	Refers to a descendant of the context node with the name *name*

13

USING XPATH TO REFERENCE A NODE

- For **absolute path**, XPath begins with the root node, identified by a forward slash and proceeds down the levels of the node tree
  - An absolute path: /child1/child2/child3/…

- To reference an element **without regard to its location** in the node tree, use a double forward slash with the name of the descendant node
  - A relative path : //descendant

14

REFERENCING GROUPS OF ELEMENTS

- XPath allows you to refer to groups of nodes by using the wildcard character (*)
- To select all of the nodes in the node tree, you can use the path:

  //*

- The (*) symbol matches any node, and the (//)symbol matches any level of the node tree
  - **Example**: /portfolio/stock/*

15

REFERENCING ATTRIBUTE NODES

- XPath uses different notation to refer to attribute nodes
- The syntax for attribute node is:

  @attribute

  where *attribute* is the name of the attribute

Example: /portfolio/stock/name/@symbol

16

WORKING WITH TEXT NODES

- The text contained in an element node is treated as a text node
- The syntax for selecting a text node is:

  @text()
- To match all text nodes in the document, use:

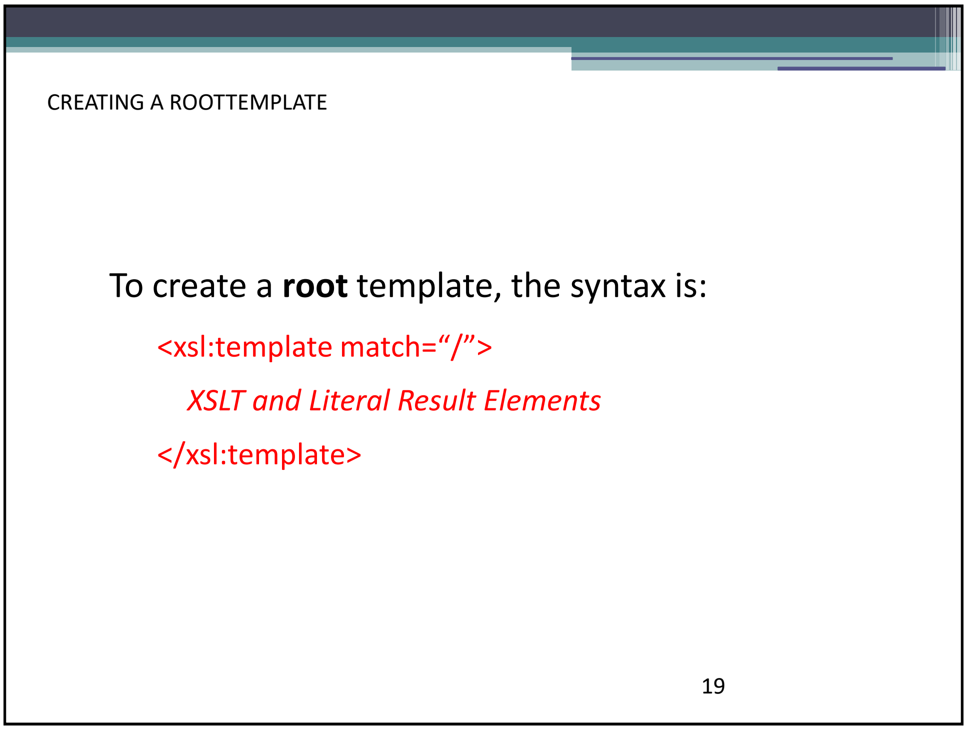
  //text()

17

CREATING THE ROOT TEMPLATE

- A **template** is a collection of elements that define how a particular section of the source document should be transformed in the result document
- The **root template** sets up the initial code for the result document

18

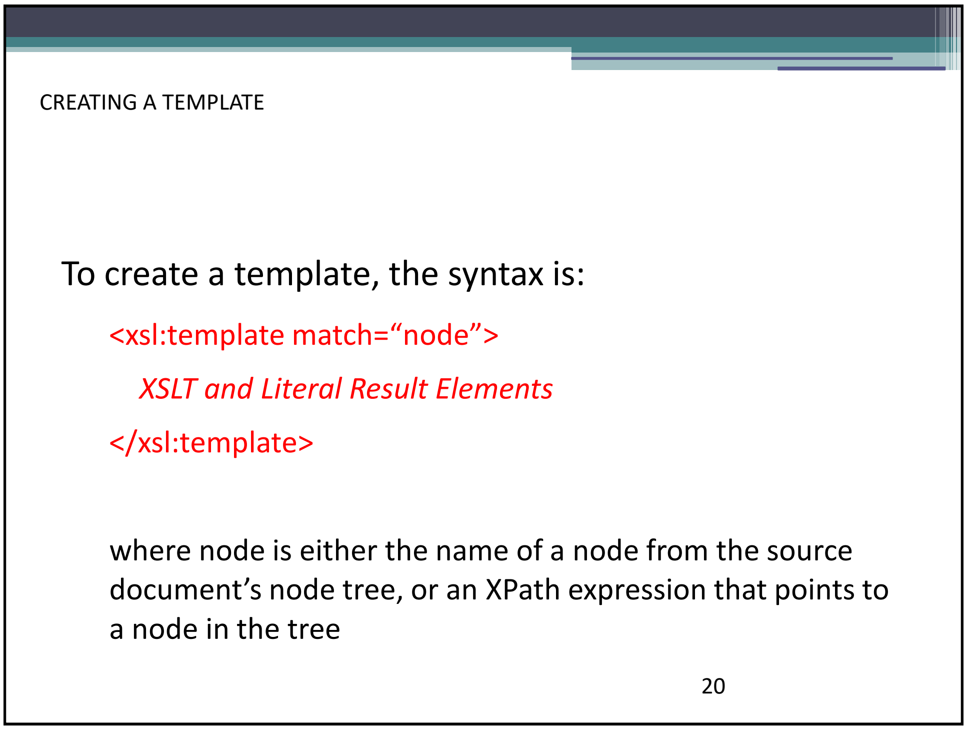


CREATING A ROOTTEMPLATE

To create a **root** template, the syntax is:

```
<xsl:template match="/">
   XSLT and Literal Result Elements
</xsl:template>
```

19



CREATING A TEMPLATE

To create a template, the syntax is:

```
<xsl:template match="node">
   XSLT and Literal Result Elements
</xsl:template>
```

where node is either the name of a node from the source document’s node tree, or an XPath expression that points to a node in the tree

20

CREATING THE ROOT TEMPLATE

- A template contains two types of content: XSLT elements and literal result elements
  - **XSLT elements** are those elements that are part of the XSLT namespace and are used to send commands to the XSLT processor
  - A **literal** result element is text sent to the result document, but not acted upon by the XSLT processor

21

CREATING THE ROOT TEMPLATE EXAMPLE

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

<xsl:template match="/">
<html>
<head>
<title>Stock Information</title>
<link href="stock.css" rel="stylesheet" type="text/css" />
</head>
<body>
<h1 class="title">Hardin Financial</h1>
<h2 class="title">Stock Information</h2>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

22

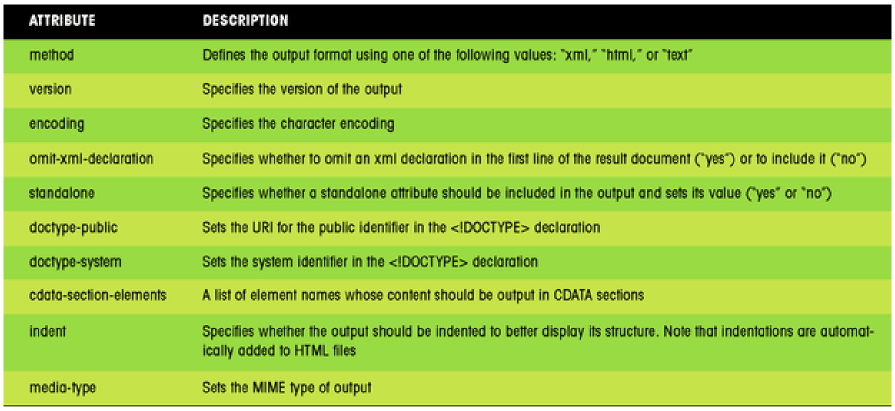
SPECIFYING THE OUTPUT METHOD

- By default, the **XSLT processor** will render the **result** document as an **XML file**
- To control how the processor formats the source document, you can specify the output method using the

  <xsl:output/>

  element

ATTRIBUTS OF THE <XSL:OUTPUT/> ELEMENT



ATTRIBUTE	DESCRIPTION
method	Defines the output format using one of the following values: "xml," "html," or "text"
version	Specifies the version of the output
encoding	Specifies the character encoding
omit-xml-declaration	Specifies whether to omit an xml declaration in the first line of the result document ("yes") or to include it ("no")
standalone	Specifies whether a standalone attribute should be included in the output and sets its value ("yes" or "no")
doctype-public	Sets the URI for the public identifier in the <!DOCTYPE> declaration
doctype-system	Sets the system identifier in the <!DOCTYPE> declaration
cdata-section-elements	A list of element names whose content should be output in CDATA sections
indent	Specifies whether the output should be indented to better display its structure. Note that indentations are automatically added to HTML files
media-type	Sets the MIME type of output

TRANSFORMING A DOCUMENT

- A **browser** with a **built-in XSLT** processor allows you to view the result document
- Most XSLT processors provide the capability to create the result document as a separate file
- An XSLT processor could transform an XML file into a HTML file.

25

CREATING AN HTML FILE

- One advantage of creating a separate HTML file is that it can be viewed in any Web browser
- You have to regenerate the HTML file every time you make a change to the source document, or the style sheet
- The XSLT processor adds one extra line to the document that provides additional information to the browser about the content of the document and its encoding

26

INSERTING A NODE VALUE

- To insert a node's value into the result document, the syntax is:
  - <xsl:value-of> select="*XPath Expression*" />
  - where *XPath Expression* is an expression that identifies the node from the source document's node tree
- If the node contains child elements in addition to text content, the text in those child nodes appears as well

27

INSERTING A NODE VALUE EXAMPLE

```
<xsl:template match="/">
<html>
<head>
<title>Stock Information</title>
<link href="stock.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="datetime"><b>Last Updated: </b>
   <xsl:value-of select="portfolio/date" /> at
   <xsl:value-of select="portfolio/time" />
</div>
<h1 class="title">Hardin Financial</h1>
<h2 class="title">Stock Information</h2>
</body>
</html>
</xsl:template>
```

28

PROCESSING A BATCH OF NODES

To process a batch of nodes, the syntax is:

<xsl:for-each select=“*XPath Expression*” />

XSLT and Literal Elements

</xsl:for-each>

where *XPath Expression* is an expression that defines the group of nodes to which the XSLT and literal result elements are applied

29

PROCESSING A BATCH OF NODES EXAMPLE

```
<xsl:template match="/">
<html>
<head>
<title>Stock Information</title>
<link href="stock.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="datetime"><b>Last Updated: </b>
   <xsl:value-of select="portfolio/date" /> at
   <xsl:value-of select="portfolio/time" />
</div>
<h1 class="title">Hardin Financial</h1>
<h2 class="title">Stock Information</h2>
<xsl:for-each select="portfolio/stock">
<h3 class="name">
   <xsl:value-of select="name" />
</h3>
</xsl:for-each>
</body>
</html>
</xsl:template>
```

30