XSL Introduction

Reference Text: New Perspectives on XML, Comprehensive, 3rd Edition
Patrick Carey, Sasha Vodnik, 2015

1

XSL

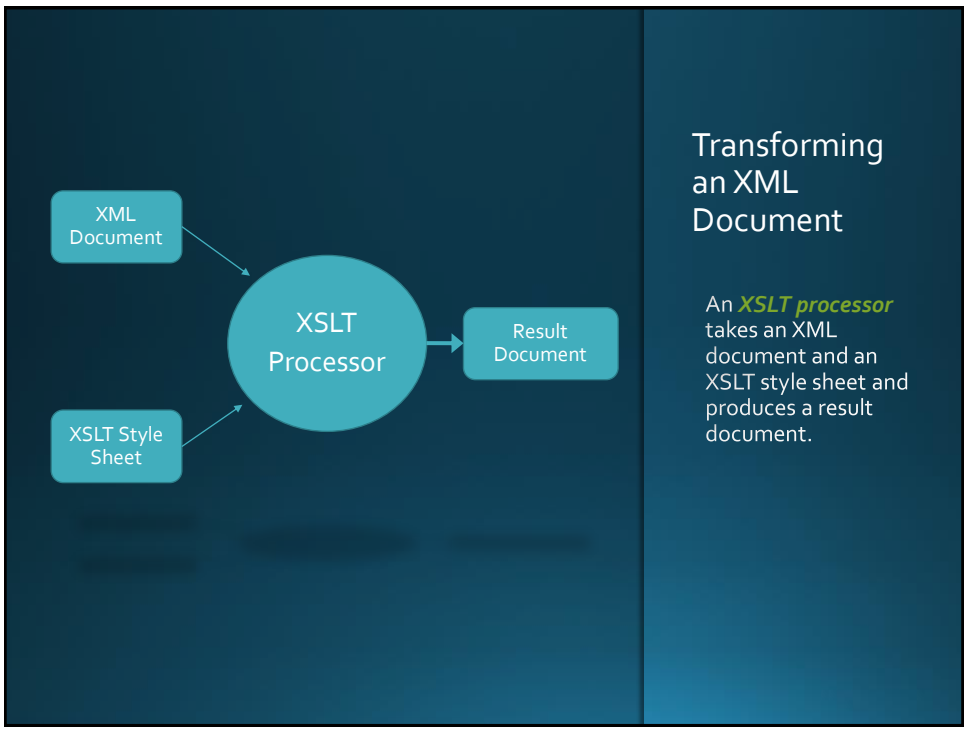
- **EXtensible Style sheet Language**
  - Developed by the W3C
- **XSLT**
  - **EXtensible Style sheet Language Transformation**
  - Used to ***transform XML*** content from one XML format to another
- **Xpath**
  - A declarative language used to ***locate*** information from an XML document and ***perform operations and calculations*** upon that content

2

XSLT Style Sheet

- An XSLT Style Sheet document
  - has a file extension ***.xsl***
  - is an **XML** document
  - contains **instructions for transforming** the contents of an XML document into another format

3



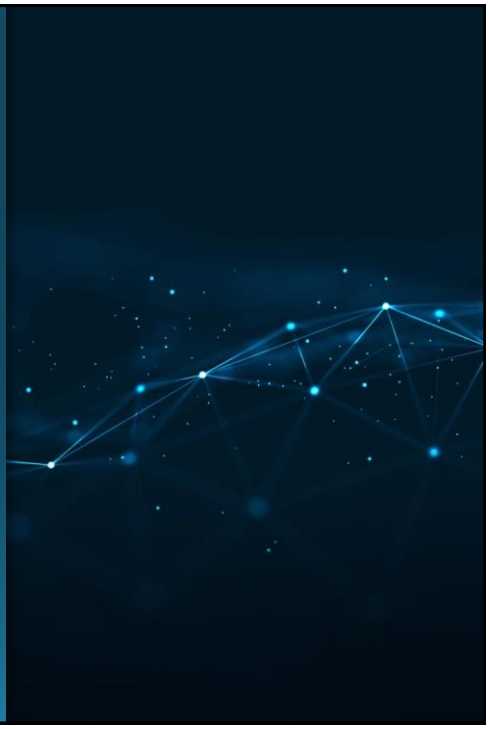
Transforming an XML Document

An ***XSLT processor*** takes an XML document and an XSLT style sheet and produces a result document.

4

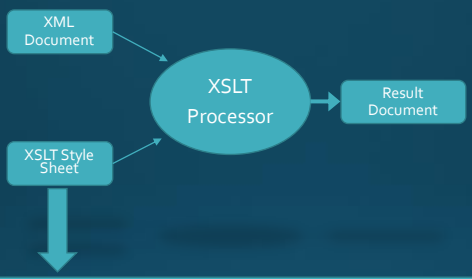
Transforming an XML Document

- Server-side transformation
  - A client can request an XML document from the server. The server applies the style sheet and returns the resulting document and not the original XML document
- Client-side transformation
  - A client can request and XML document and a stylesheet from a server and perform the transformation itself.



5

XSLT Stylesheets

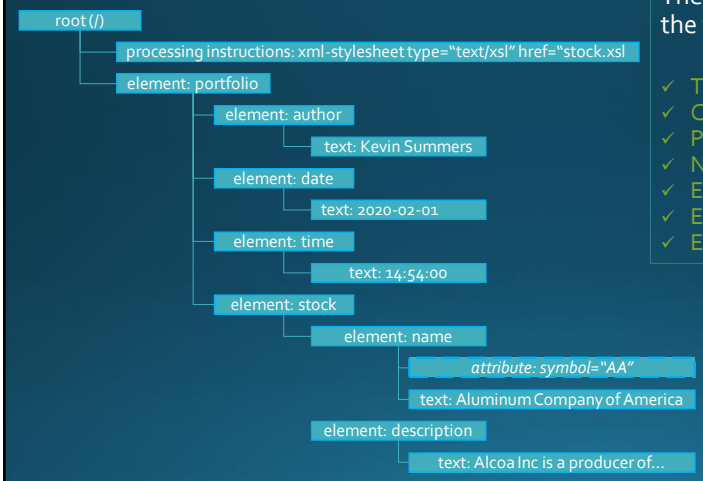



```
<?xml version ="1.0"?>
<xsl:stylesheet version=1.0  xmlns:xsl=  "http://www.w3.org/1999/XSL/Transform">
    Content of the style sheet
</xsl:stylesheet>
```

6

Remember the Node Tree…

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

7

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

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**

8

XPath

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**

9

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*

10

Using XPath to Reference Nodes

An absolute path:	Starting from the root node: */child1/child2/child3/…*
A relative path regardless of location	A relative path : *//descendant*
Referencing groups of nodes	Use the wildcard character (*) To select all of the nodes from any starting point: ***Example**: /portfolio/stock/**
Referencing all nodes	To select every node in the node tree you could use: ***//****
Referencing Attribute Nodes	The syntax for referencing an attribute node is: ***@attribute*** where *attribute* is the name of the attribute ***Example**: /portfolio/stock/name/@symbol*

11

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()*

12

Templates

13

Templates

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

14

The Root Template

Root Template

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

Other Templates

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

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

15

Example Stylesheet with Root Template

Outputting HTML literals

No XSLT Elements

```
<?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>
```

16

The *xsl:output* element

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

17

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

Attributes of the *<xsl:output/>* Element

18

Transforming a Document

Most XSLT processors provide the capability to create the result document as a separate file

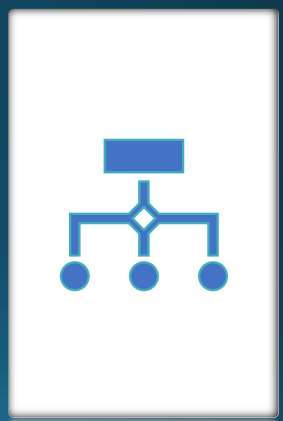
An XSLT processor could transform an XML file into a HTML file.

A **browser** with a **built-in XSLT** processor allows you to view the result document

19

Example: Transforming an XML Document into a HTML Document

- XSLT can be used to transform XML into many different types of documents
- From an ***Enterprise*** perspective, this is most useful when ***transforming XML from one schema to another schema*** (or any other data format).
- To explore some of its capabilities however, we'll look at some examples of transforming XML into HTML so that a browser can display it.
- 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



20

Back to our Stylesheet - Insert 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

```
<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>
```

21

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

```
<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>
```

22