**­**

***Stylesheets for EAD:***

***Delivering Your Finding Aids on the Web***

**Agenda**

9:00-9:30 Course objectives, ground rules, caveats

 Lessons 1-3

9:30- 10:15 Lesson 4-7

10:15-10:30 Break

10:30-12:00 Lessons 8-12

12:00-1:00 Lunch

1:00-2:30 Review

 Lessons 13-15

2:30-2:45 Break

2:45-4:45 Lessons 16-17

4:45-5:00 Review***Stylesheets for EAD***

**Table of Contents**

Lesson 1. A gentle introduction to stylesheets 2

Lesson 2. Writing an XSLT stylesheet 4

Lesson 3. High-level XSLT elements 5

Lesson 4. Building templates 7

Lesson 5. Getting down to business 13

Lesson 6. Simple templates 16

Lesson 7. Operators 22

Lesson 8. Templates for multiple elements 24

Lesson 9. More elements that are alike 28

Lesson 10. Testing, sorting, supplying 31

Lesson 11. Lists 36

Lesson 12. Back to the <did> 38

Lesson 13. Description of subordinate components and <c01> 42

Lesson 14. Named and called templates 51

Lesson 15. Adding more components 53

Lesson 16. Table of contents 57

Advanced Topics

Lesson 17. Linking 65

Lesson 18. Functions 73

Lesson 19. Expressions 75

Lesson 20. Manipulating columns and container information 85

Lesson 21. Conditional statements 87

Lesson 22. Beautifying the title page 92

Lesson 23. Modularization 93

Alternate formatting syntax 94

Quick list of XSLT elements and attributes used in the workshop 97

XSLT and XPath symbols 99

Stylesheets with schemas 100

XSLT resources 102

**Course objectives**

Provide an introduction to XSLT, as it applies to the presentation of archival finding aids over the web.

Students will understand XSLT concepts sufficiently to produce a very basic stylesheet or to make simple modifications to an existing one.

Not a comprehensive introduction to the subject.

The workbook contains a section on advanced topics that you will be able to explore later on your own.

**Ground rules**

Keep the discussion informal—ask questions!

Resist the urge to work ahead

Follow instructions!

No email except during breaks

Silence cell phones

Please return on time from breaks and lunch

We didn’t invent this stupid programming language

Be calm. Don't worry

**Workshop caveats**

**Nota Bene for the Class:**

This workshop is more about the logical analysis of our data and finding aids than about archival theory. The computer programming language we use is simply a grammar for expressing our solutions to practical problems.

Do not expect to understand everything we cover here right away. Subsequent review and reflection will be necessary. The instructors will be available to answer questions after the workshop.

We will not discuss every element and all possible attributes in the Extensible Stylesheet Language. We will focus on those of most utility for producing EAD finding aids.

***Lesson 1: A Gentle Introduction to Stylesheets***

**1.1 What are styles and stylesheets?**

Styles and stylesheets prescribe the appearance, the presentation of information on the printed page and on computer displays/

Stylesheets written in the Extensible Stylesheet Language (XSL) language do this differently from other presentation methods and thereby offer greater functionality.

**1.2 Technical environments**

**Styles in text processing**: wordprocessors, graphics and printing software

**The HTML family:** HTML, CSS (Cascading Style Sheets)

**The XML family** XSL (Extensible Stylesheet Language)

* XSL Transformation (XSLT) Transforming XML
* XSL Format Objects (XSL-FO) Formatting XML for print
* XML Path Language (XPath) Navigating XML

**1.3 XSL Transformations**

1. XML documents consist of content inside a semantic wrapper.

2. An XSL transformation creates a new wrapper and flows the contents of the original file into it.

**Original wrapper New Wrapper**

<head> <h2>

 Biography of Bill Smith Biography of Bill Smith

</head> </h2>

3. XSL Stylesheets are computer files that specify the structure of the new wrapper and what content flows into it. Stylesheet files are separate from the XML (EAD) file.

4. The structure and content of the original XML file are unchanged by the transformation process.

**Transformations**

**XML**

**Document**

**Text**

**Stylesheet**

**XSLT**

**Processor**

**HTML**

**Document**

**Saxon**

**Xalan**

**MSXSL**

**Browsers**

**OS**

**PDF**

**RTF**

**XML**

**EAD**

**MARC**

**DC**

**METSMODS**

**EAD**

**XSLT**

###### *Lesson 2: Writing an XSLT Stylesheet*

###### 2.1 Stylesheet writing process

1. Determine the content and appearance of the desired output

2. Identify the information in the source document that corresponds to the output you want.

3. Identify the HTML styling that will produce the display characteristics of that output.

1. Translate the styling and element selection into the XSLT syntax.

**2.2 Tools for writing stylesheets**

Simple text editors

 Notepad, Wordpad, NoteTab

Multipurpose XML editors

 oXygen, XMLSpy

**2.3 The workshop environment**

* Source document: *fonds.xml*
* Stylesheet: *style.xsl*
* Use oXygen software to edit the stylesheet
* XSLT Processor: *MSXML* (in Oxygen)
* Output: *fonds.html*

***Lesson 3: High-level XSLT Elements***

#

**3.1 XSLT is XML**

 The XML declarationisrequired at the beginning of every XML document:

<?xml version="1.0"?>

**3.2 xsl:stylesheet—the root element**

**<xsl:stylesheet>**
Definition: The document element of the stylesheet. Like <ead>.
Key attributes:

version—specifies the version of XSLT utilized (mandatory)

xmlns—declares that the elements having a particular prefix belong to a particular namespace

**3.3 Namespace declaration**

XSL requires the following standard declaration to indicate that the elements that contain the xsl prefix belong to the XSL namespace.

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

**<xsl:output>**
Definition: specifies the type of output
Key attributes:

method—specifies one of three options: XML, HTML, text

indent—if the value is "yes," white space is inserted into the output file to provide an indented presentation

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

**3.4 Strip-space element**

**<xsl:strip-space>**
Definition: removes certain white space nodes from the source document
Key attribute:

elements—specifies the elements from which white space is removed

Removes white space from the text nodes between elements and from elements that consist solely of white space. XML defines white space to include characters for blank spaces, tabs, line feeds, and carriage returns.

**Example: Our starting template**

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

<xsl:stylesheet

version="1.0"

xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

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

<xsl:strip-space elements="\*"/>

</xsl:stylesheet>

***Lesson 4: Building Templates***

**4.1 The template**

The template is the basic building block of the stylesheet.

**<xsl:template>**
Definition: prescribes an action to be performed by an XSLT processor
Key attributes:

match—specifies the elements or attribute nodes in the source document to which the template applies. Establishes the context.

Templates have three parts:

1. The match pattern specifies the element or attributes in the document to which the template applies (this is done in the match attribute). (see 4.2)

2. The template body defines the action to be taken on the node identified in the match pattern. Template bodies have two parts:

1. Formatting instructions (HTML syntax in this workshop) (see 4.3)
2. Transformation instructions (see 4.4)

**This sample template**

<xsl:template match="title">

 <i>

 <xsl:apply-templates/>

 </i>

</xsl:template>

**Transforms this EAD encoding**

 <title>Moby Dick</title>

**Into this HTML output**

 <i>Moby Dick</i>

**4.2** **Template Body Part 1:**  **Match Patterns**

Match patterns identify a specific node on the tree to be transformed by a template.

a. Patterns consist of a series of **steps**, each of which specifies an element or attribute in the family tree. Steps are separated by a slash.

<xsl:template match=”**archdesc/scopecontent/p”>** = three steps

b. The match is to the last node in the path, before the qualifier (if there is one). Include as much of the tree in the path as you need to uniquely identify the node.

c. Match patterns may be **absolute--**

bioghist matches all <bioghist> elements

p matches all paragraph elements

bioghist/p matches only paragraph elements that are children of <bioghist> elements, not ones that are the children of a <scopecontent> element

or **relative**

 \* the asterisk is a wild card standing for any element

archdesc/\*/note

 matches

 archdesc/did/note

 and

archdesc/controlaccess/note

 but not

archdesc/note

d. Match patterns may also refer to attributes--

@type refers to any type attribute

container/@type refers to type attributes that are children of container

e. Match patterns may also be qualified by **predicates** or **step qualifiers** which appear in square brackets--

container[@type] matches container elements that have type attributes

f. Predicates may be further qualified--

container[@type='box'] matches container elements that have type attributes whose value is "box"

1. More elaborate patterns, called expressions, are defined by XPath.
2. Include as much of the path as needed to identify the node unambiguously.

**4.3 Template Body Part 2: Formatting Instructions**

We use HTML syntax and Cascading Style Sheets (CSS) extensions to specify the appearance of our XML files so that they might display properly in a browser.

**html** The document or root element of an HTML document

**head** Wrapper element for information about the web document. Contains the title and style elements

**title** Child of the head element that generates the display title at the top of a web page. This is a mandatory element in HTML 4.0

**body** Wrapper element for the part of the HTML document that displays in a browser

**a** Anchor that serves as either the origination or target of a link

**b** Bold

**br** Line break

**img** Image element containing a link to an external image file

**h1 to h6** Headings of various sizes. Most browsers display these as bold text with a blank line following

**i** Italic

**p** Paragraph. A block element, followed by a blank line.

**table** Wrapper for an HTML tabular display

**td** Table data--one cell of data in a table row

**tr** Table row--one row across the screen in a tabular format

# Attributes

**colspan** Defines the numbers of columns in a table that a table data cell spans

**style** Contains the values for display features such as those listed below

**valign** Vertical alignment within a table data cell

**width** Specifies the width of a table, table row, or table data cell

# CSS Properties

**text-align** center | left | right

**text-indent** Number of points (pts) of indention for the first line of text

**font-family** Specifies the font to be used: Arial, Times, etc.

**Entities**

**&#160;** Non-breaking space

**4.4 Template Body Part 3: Transformation Instructions**

**Three possible ways to insert text from the source tree into the result tree:**

1. <xsl:apply-templates>

2. <xsl:value-of> Limited application with EAD

3. <xsl:call-template> Covered in Lesson 14.

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<p>Herman Melville wrote many novels but certainly the most famous is <title render="italic">Moby Dick</title>, the story of a sea captain and the white whale with which he was obsessed.</p>

**EAD Tree\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

element: p

|

|\_\_\_\_\_\_ text: Herman Melville wrote many novels but certainly the most

| famous is , the story of a sea captain and the white whale

| with which he was obsessed.

|

|

|\_\_\_\_\_\_element: title

 |

|

|\_\_\_\_\_\_text: Moby Dick

**Desired Result\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Herman Melville wrote many novels but certainly the most famous is *Moby Dick*, the story of a sea captain and the white whale with which he was obsessed.

**HTML Encoding\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<div>Herman Melville wrote many novels but certainly the most famous is <i>Moby Dick</i>, the story of a sea captain and the white whale

with which he was obsessed.</div>

**4.4.1 <xsl:apply-templates>**

**<xsl:apply-templates>**

Definition: processes all children of the context element. If the select attribute is present, it processes templates for the node specified. Otherwise, it processes all child nodes.

Key attribute:
select—specifies particular child nodes to which templates are to be applied

mode—processes only templates with a matching mode attribute

**XSLT Template A:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="p">

<div>

**<xsl:apply-templates/>**

</div>

</xsl:template>

**HTML Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<div>Herman Melville wrote many novels but certainly the most famous is Moby Dick, the story of a sea captain and the white whale

with which he was obsessed.</div>



**4.4.2 The default template**

If an <xsl:apply-templates> statement fails to find an explicit template that matches any child or descendant node of the context node, as defined in the match pattern, XSLT applies a default template that inserts the text of those nodes into the result tree without formatting it further.

**XSLT Template A (again):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="p">

<div>

**<xsl:apply-templates/>**

</div>

</xsl:template>

**XSLT Template B:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Add another template for the display of the <title> element.

<xsl:template match="title[@render='italic']">

<i>

<xsl:apply-templates/>

</i>

</xsl:template>

**HTML Output from Templates A and B**

<div>Herman Melville wrote many novels but certainly the most famous is <i>Moby Dick</i>, the story of a sea captain and the white whale

with which he was obsessed.</div>



***Lesson 5: Getting Down to Business***

**5.1 The base HTML template**

**XSLT Template\_1:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Creates the basic HTML shell for the output file. No content from the EAD instance has yet been inserted as no transformation instruction is given so the display produces a blank screen.

<xsl:template match="ead">

<html>

<head>

 <title></title>

</head>

<body>

</body>

</html>

</xsl:template>

Transformation produces an output tree that looks like this:



**HTML Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<html>

<head>

<title></title>

</head>

<body></body>

</html>



 [A blank screen appears]

**5.2 Inserting content into the output tree**

**XSLT Template 2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The entire contents of the EAD file will be inserted, unformatted, into the HTML output. This template modifies Template 1.

**<xsl:template match="ead">**

**<html>**

**<head>**

**<title></title>**

**</head>**

**<body>**

<xsl:apply-templates/>

**</body>**

**</html>**

**</xsl:template>**

**HTML Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<html>

<head>

<title></title>

</head>

<body>2468William Fonds ProvenanceAn Inventory of His Papers at the Cupcake Corners Historical SocietyInventory prepared by B.W. MoosCUPCAKE … ***and all the rest of the text of the finding aid***

</body>

</html>

***Lesson 6: Simple Templates***

**6.1 Templates for a single element and its children**

A template may govern the transformation and display of multiple elements. One common example is a template that covers a parent element and its children.

**6.2 Finding Aid: Title Page**

# Desired Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# William Fonds Provenance

## An Inventory of His Papers at the Cupcake Corners Historical Society

**HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_**

<h1 style="text-align:center">

William Fonds Provenance

</h1>

<h2 style="text-align:center">

An Inventory of His Papers at the Cupcake Corners Historical Society

</h2>

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<ead>

<eadheader>

<eadid>2468</eadid>

<filedesc>

<titlestmt>

<titleproper>William Fonds Provenance</titleproper>

<subtitle>An Inventory of His Papers at the Cupcake Corners Historical Society</subtitle>

<author>Inventory prepared by B.W. Moos</author>

</titlestmt>

</filedesc>

</eadheader>

**XSLT Template\_3:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The titleproper and subtitle are inserted into the output. Use of the select attribute causes the <eadheader> children not so specified to be suppressed.

<xsl:template match="eadheader">

<h1 style="text-align:center">

<xsl:apply-templates select="filedesc/titlestmt/titleproper"/>

</h1>

<h2 style="text-align:center">

<xsl:apply-templates select="filedesc/titlestmt/subtitle"/>

</h2>

</xsl:template>

**HTML Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<html>

<head>. . .</head>

<body>

<h1 style="text-align:center">William Fonds Provenance</h1>

<h2 style="text-align:center">An Inventory of His Papers at the Cupcake Corners Historical Society</h2>

Collection SummaryProvenance, William Fonds, 1897-1956Papers of William . . . ***and all the rest of the text of the finding aid***

</body>

</html>



**6.3 Suppressing the display of content**

A template that lacks formatting instructions and transformation instructions for some or all of the nodes specified in the match statement will cause those parts of the context element not to appear in the output tree and therefore not to be displayed.

The use of the select attribute in the preceding template caused only the two specified children of <eadheader> to be transformed. The effect is to suppress the display of the other child elements that we do not wish to appear.

**6.4 Finding aid: Collection summary**

The <did> element in <archdesc> has a complex display. For the moment, we will apply a template that will suppress the display of all the archdesc/did elements. We will return to complete this template in Lesson 15.

**XSLT Template\_4:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Lacking any specific formatting or transformation instructions, this template suppresses the display of the <did> element.

<xsl:template match="archdesc/did">

</xsl:template>



**6.5 Templates for single elements**

It is also possible and often desirable to limit a template to a single element or a series of related elements. Most of the remaining templates in this workbook follow this pattern.

## Desired Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Biography of William Provenance

Archivist and author William Fonds Provenance was born at Last Chance, Nevada, to Fred and Mary Jones Provenance on January 4, 1897. Little is known of his early life prior to serving in World War I as an ambulance driver. After graduating from Freen College in 1924 with a degree in cryptogamic biology, he first followed a career in commercial horticulture and later worked as an itinerant archivist. Provenance also had a lifelong interest in creative writing, producing both novels and poetry. He died at Frostbite Falls, Minnesota on March 15, 1956.

**HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_\_**

<h2>Biography of William Provenance</h2>

<p style="text-indent:25pt">Archivist and author William Fonds Provenance was born at Last Chance, Nevada, to Fred and Mary Jones Provenance on January 4, 1897. Little is known of his early life prior to serving in World War I as an ambulance driver. After graduating from Freen College in 1924 with a degree in cryptogamic biology, he first followed a career in commercial horticulture and later worked as an itinerant archivist. Provenance also had a lifelong interest in creative writing, producing both novels and poetry. He died at Frostbite Falls, Minnesota, on March 15, 1956. </p>

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<bioghist encodinganalog="545">

<head id="t2">Biography of William Provenance</head>

<p>Archivist and author William Fonds Provenance was born at Last Chance, Nevada to Fred and Mary Jones Provenance on January 4, 1897. Little is know of his early life prior to serving in World War I as an ambulance driver. After graduating from Freen College in 1924 with a degree in cryptogamic biology, he first followed a career in commercial horticulture and later worked as an itinerant archivist. Provenance also had a lifelong interest in creative writing, producing both novels and poetry. He died at Frostbite Falls, Minnesota, on March 15, 1956.</p>

</bioghist>

**XSLT Template 5:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

These two templates create an HTML head and paragraph for <bioghist>.

<xsl:template match="archdesc/bioghist/head">

<h2>

<xsl:apply-templates/>

</h2>

</xsl:template>

###### XSLT Template 6:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<xsl:template match="archdesc/bioghist/p">

<p style="text-indent:25pt">

<xsl:apply-templates/>

 </p>

</xsl:template>



**XSLT Template\_C:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Alternatively, one could create a single template for both the head and paragraph children of <bioghist>.

**<xsl:template match=**"**archdesc/bioghist**"**>**

**<h2>**

**<xsl:apply-templates select=**"**head**"**/>**

**</h2>**

**<p>**

**<xsl:apply-templates select=**"**p**"**/>**

**</p>**

**<xsl:template>**

**XSLT Template\_D:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Template C does not work if <bioghist> contains more than one paragraph. Instead an <xsl:for-each> loop is required to create a separate output paragraph for each input paragraph. This illustrates one of the problems of the single template approach when one might not anticipate every encoding scenario possible. See Lesson 10.2 for more on the <xsl:for-each> element.

**<xsl:template match=**"**archdesc/bioghist**"**>**

**<h2>**

**<xsl:apply-templates select=**"**head**"**/>**

**</h2>**

**<xsl:for-each select=**"p">

**<p>**

**<xsl:apply-templates/>**

**</p>**

**</xsl:for-each>**

**<xsl:template>**

***Lesson 7: Operators***

**7.1 Operators**

Specify a variety of processes that XSLT software can perform on data in the source document, particularly on numerical information.

**7.1.1 Comparison operators**

ValueWritten as

equals =

does not equal !=

less than <

greater than >

less than or equal to <=

greater than or equal to >=

**7.1.2 Numerical operators**

ValueWritten as

addition +

subtraction -

multiplication \*

**7.1.3 Logical operators**

ValueWritten as

and and

or or

Use the "or" operator to test for multiple possibilities

<xsl:if test="relatedmaterial or separatedmaterial">

tests to see if there is a relatedmaterial or separatedmaterial element node

Use the "and" operator to limit the nodes matched

<xsl:if test="unittitle and unitdate">

tests to see if there are both unittitle and unitdate element nodes

**7.1.4 Union operator**

ValueWritten as

union of |

Use the "union" operator to match multiple nodes

<xsl:template match="scopecontent | bioghist | arrangement">

matches all scopecontent, bioghist, and arrangement nodes

Technically speaking, the union operator returns a node set that is the sum of all the nodes named in the match pattern.

###### XSLT Template E:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This template would output all <persname> and <corpname> elements in italics.

<xsl:template match="persname | corpname">

<i>

<xsl:apply-templates/>

</i>

</xsl:template>

***Lesson 8: Templates for Multiple Elements;
or Similar Markup, Similar Output***

Several EAD elements or parts of EAD elements may have identical encoding and identical output display. Many elements in the finding aid for the William Fonds Provenance papers have similar encoding and display. Note, for example, that <scopecontent> and <bioghist> are the same with regard to the head and paragraphs.

There are three ways to handle such similarities in stylesheets:

1. Create multiple templates

2. Employ the union operator

3. Invoke named templates (Lesson 14)

**8.1 Multiple templates**

One may create multiple templates that are identical except for the element they match.

Since <scopecontent> has the same structure as <bioghist>, we could create new templates for it that are almost exactly like Template 5 and Template 6.

**Desired Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Scope and Contents of the Papers

The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance.

The bulk of the collection consists of correspondence, principally with his mother, other archivists and writers. His diaries describe his experiences as an ambulance driver in France during World War I in vivid detail. Major correspondents represented in the collection include Ernest Hemingway, Ernst Posner, and Provenance's long-time companion Ima Gusdorf.

**HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_\_**

<h2>Scope and Contents of the Papers</h2>

<p style="text-indent:25pt">The collection consists of diaries, correspondence, manuscripts, and miscellaneous

materials documenting the literary and archival career of William Fonds Provenance.</p>

<p style="text-indent:25pt">The bulk of the collection consists of correspondence, principally with his mother, other archivists and writers. His diaries describe his experiences as an ambulance driver in France during World War I in vivid detail. Major correspondents represented in the collection include Ernest Hemingway, Ernst Posner, and Provenance's long-time companion Ima Gusdorf.

</p>

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<scopecontent>

<head>Scope and Contents of the Papers</head>

<p>The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance.</p>

<p>The bulk of the collection consists of correspondence, principally with his mother, other archivists and writers. His diaries describe his experiences as an ambulance driver in France during World War I in vivid detail. Major correspondents represented in the collection include Ernest Hemingway, Ernst Posner, and Provenance's long-time companion Ima Gusdorf.</p>

</scopecontent>

**XSLT Template F:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

These two templates create an HTML head and multiple HTML paragraphs for <scopecontent>.

<xsl:template match="archdesc/scopecontent/head">

<h2>

<xsl:apply-templates/>

</h2>

</xsl:template>

**XSLT Template G:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="archdesc/scopecontent/p">

 <p style="text-indent:25pt">

 <xsl:apply-templates/>

 </p>

</xsl:template>

**Compare these with Templates 5 and 6.8.2 Multiple parallel elements**

The union operator permits formulation of a match statement that applies to multiple nodes.

**XSLT Template 7: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

These templates use the union operator to consolidate the previous four templates into two, matching both <bioghist> and <scopecontent>. The first modifies Template 5, the second modifies Template 6.

<xsl:template match="archdesc/bioghist/head

 **| archdesc/scopecontent/head** ">

 <h2>

 <xsl:apply-templates/>

 </h2>

</xsl:template>

**XSLT Template 8: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="archdesc/bioghist/p

 **| archdesc/scopecontent/p** ">

 <p style="text-indent:25pt">

 <xsl:apply-templates/>

 </p>

</xsl:template>

**Output from Templates 7 and 8:**



***Lesson 9: More Elements that Are Alike***

The content and encoding of multiple elements may follow the same pattern.

**Desired Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Arrangement of the Papers

The papers are arranged into two series:

Correspondence, 1919-1955.

Diaries, 1917-1918.

## Related Records

The papers of Ima Gusdorf are located in the Freen University archives.

## Index Terms

This collection is indexed under the following headings in the catalog of the Cupcake Corners Historical Society. Researchers desiring materials about related topics, persons, or places should search the catalog using these headings.

**Persons:**

Gusdorf, Ima May

Hemingway, Ernest, 1899-1961

Posner, Ernst

**Subjects:**

Cataloging of archival materials

United States – History – World War, 1914-1918 – Personal narratives, American

**Document Types:**

Diaries

Novels

## Restrictions on Access

Access to the correspondence between Provenance and Ernest Hemingway is restricted until 2025.

## Acquisition Information

Acquired as a gift from Ima Gusdorf, December 17, 1952.

## Processing Information

Collection processed and cataloged by B.W. Moos, January, 1962.

**HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_\_**

<h2>Arrangement of the Papers</h2>

<p style="text-indent:25pt">The papers are arranged into two series:</p>

<h2>Related Records</h2>

<p style="text-indent:25pt">The papers of Ima Gusdorf are located in the Freen University archives.</p>

<h2>Index Terms</h2>

<p style="text-indent:25pt">This collection is indexed . . . . </p>

<h2>Restrictions on Access</h2>

<p style="text-indent:25pt">Access to the correspondence between Provenance and Ernest Hemingway is restricted until 2025.</p>

<h2>Acquisition Information</h2>

<p style="text-indent:25pt">Acquired as a gift from Ima Gusdorf, December 17, 1952</p>

<h2>Processing Information</h2>

<p style="text-indent:25pt">Collection processed and cataloged by B.W. Moos, January, 1962.</p>

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## <arrangement>

## <head>Arrangement of the Papers</head>

<p>The papers are arranged into two series:</p>

<list>. . .</list>

</arrangement>

<relatedmaterial>

<head>Related Records</head>

<p>The papers of Ima Gusdorf are located in the Freen University archives.</p>

</relatedmaterial>

<controlaccess>

 <head>Index Terms</head>

 <p>This collection is indexed under the following headings….</p>

 <controlaccess> … </controlaccess>

</controlaccess>

<accessrestrict>

<head>Restrictions on Access</head>

<p>Access to the correspondence between Provenance and Ernest Hemingway is restricted until 2025.</p>

</accessrestrict>

<acqinfo>

<head>Acquisition Information</head>

<p>Acquired as a gift from Ima Gusdorf, December 17, 1952</p>

</acqinfo>

<processinfo>

<head>Processing Information</head>

<p>Collection processed and cataloged by B.W. Moos, January, 1962.</p>

</processinfo>

**XSLT Template 9: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The two existing templates are extended to cover <head> and <p> elements in all the children of <archdesc>. This illustrates the power of the union operator to consolidate and thereby simplify templates. Template 9 modifies Template 5. Template 10 modifies Template 6.

<xsl:template match="archdesc/bioghist/head |

 archdesc/scopecontent/head **|**

 **archdesc/arrangement/head |**

 **archdesc/relatedmaterial/head |**

 **archdesc/controlaccess/head |**

 **archdesc/accessrestrict/head |**

 **archdesc/acqinfo/head |**

 **archdesc/processinfo/head"**>

 <h2>

 <xsl:apply-templates/>

 </h2>

</xsl:template>

**XSLT Template 10: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="archdesc/bioghist/p |

 archdesc/scopecontent/p **|**

 **archdesc/arrangement/p |**

 **archdesc/relatedmaterial/p |**

 **archdesc/controlaccess/p |**

 **archdesc/accessrestrict/p |**

 **archdesc/acqinfo/p |**

 **archdesc/processinfo**">

 <p style="text-indent:25pt">

 <xsl:apply-templates/>

 </p>

</xsl:template>

Lesson 10: Testing, Sorting, and Supplying

The transformation of <controlaccess> is more complex because the encoding created a simple list of elements, perhaps even in random order rather by type or alphabetical, and the presentation requires groupings by child and the insertion of headings.

**Desired Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Index Terms

This collection is indexed under the following headings in the catalog of the Cupcake Corners Historical Society. Researchers desiring materials about related topics, persons, or places should search the catalog using these headings.

**Persons:**

Gusdorf, Ima May

Hemingway, Ernest, 1899-1961

Posner, Ernst

**Subjects:**

Cataloging of archival materials

United States – History – World War, 1914-1918 – Personal narratives, American

**Document Types:**

Diaries

Novels

**HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_\_**

<h2>Index Terms</h2>

<p style="text-indent:25pt">This collection is indexed … should search the catalog using these headings.</p>

<h3 style="margin-left:25pt">Persons:</h3>

<div style="margin-left:50pt">Gusdorf, Ima May </div>

<div style="margin-left:50pt">Posner, Ernst</div>

<div style="margin-left:50pt">Hemingway, Ernest,1899-1961. </div>

<h3 style="margin-left:25pt">Subjects:</h3>

<div style="margin-left:50pt">Cataloging of archival materials</div>

<div style="margin-left:50pt">United States-History-World War, 1914-1918- Personal narratives, American</div>

***and so forth***

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<controlaccess>

<head>Index Terms</head>

<p>This collection is indexed under the following headings in the catalog of the Cupcake Corners Historical Society. Researchers desiring materials about

related topics, persons, or places should search the catalog using these headings.</p>

<persname>Hemingway, Ernest, 1899-1961</persname>

<persname>Posner, Ernst</persname>

<persname>Gusdorf, Ima May </persname>

<subject>Cataloging of archival materials</subject>

<subject>United States -- History -- World War, 1914-1918 -- Personal narratives, American</subject>

<genreform>Novels</genreform>

<genreform>Diaries</genreform>

</controlaccess>

**10.1 Single conditions: making simple choices**

Use <xsl:if> when testing for a condition, such as the presence of an element or attribute, the position of an element, or the value of an element or attribute.

**<xsl:if>**
Definition: test for a condition specified in a test attribute
Key attribute:

test—an expression that specifies the condition being tested for

**10.2 Transforming multiple instances of a child element**

When a parent element has more than one instance of a particular child element, it is sometimes necessary to process each such child element separately.

**<xsl:for-each>**
Definition: creates a loop through successive instances of an element
Key attribute:

select—specifies the node to be returned

**10.3 Sorting the output**

Sometimes it is helpful to sort the output alphabetically or numerically.

**<xsl:sort>**

Definition: defines the sequence for processing nodes

Key attributes:
order—specifies whether nodes are processed in ascending or descending order

data-type—specifies whether the values are to be collated alphabetically or numerically

select—specifies the sort key

**10.4 Inserting text**

It is sometimes useful or necessary to insert text into the output file.

**<xsl:text>**

Definition: Delimits text that is added to the output tree that is not part of the source document. The content of this element is not parsed.

Key attributes: None

**XSLT Template 11:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template transforms <controlaccess>. It does the following

1. Applies the previous templates for <head> and <p>
2. Tests, in turn, for the presence of other child elements
3. For each type of child (persname, subject, genreform, etc.) present, it inserts a display heading indented 25 points.
4. Sorts the child elements of that type alphabetically
5. Inserts each instance of that child into an output <div> element indented 50 points.

<xsl:template match="archdesc/controlaccess">

 <xsl:apply-templates select="head"/>

 <xsl:apply-templates select="p"/>

 <xsl:if test="persname">

 <h3 style="margin-left:25pt">

 <b>

 <xsl:text>**Persons**:</xsl:text>

 </b>

 </h3>

 <xsl:for-each select="persname">

 <xsl:sort order="ascending" data-type="text"/>

 <div style="margin-left:50pt">

 <xsl:apply-templates/>

 </div>

 </xsl:for-each>

 </xsl:if>

<xsl:if test="subject">

 <h3 style="margin-left:25pt">

 <b>

 **<xsl:text>Topics:</xsl:text>**

 </b>

 </h3>

 <xsl:for-each select="subject">

 <xsl:sort order="ascending" data-type="text"/>

 <div style="margin-left:50pt">

 **<xsl:apply-templates/>**

 </div>

 </xsl:for-each>

 </xsl:if>

 <xsl:if test="genreform">

 <h3 style="margin-left:25pt">

 <b>

 **<xsl:text>Document Types:</xsl:text>**

 </b>

 </h3>

 <xsl:for-each select="genreform">

 <xsl:sort order="ascending" data-type="text"/>

 <div style="margin-left:50pt">

 **<xsl:apply-templates/>**

 </div>

 </xsl:for-each>

 </xsl:if>

</xsl:template>

Repeat this pattern for other child elements as needed, e.g., for geogname, function, corpname etc.

Lesson 11: Lists

The encoding of <arrangement> is more complex because of the inclusion of a list to facilitate in this case the creation of hyperlinks within the finding aid between the arrangement element and the series descriptions. The hyperlink will be added in a latter lesson.

**Desired Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Arrangement of the Papers

The papers are arranged into two series:

Correspondence, 1919-1955.

Diaries, 1917-1918.

**HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_\_**

<h2>Arrangement</h2>

<p style="text-indent:25pt"> The papers are arranged into two series:</p>

<div style="text-indent:50pt">Correspondence, 1919-1955.</div>

<div style="text-indent:50pt">Diaries, 1917-1918.</div>

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## <arrangement>

## <head>Arrangement of the Papers</head>

<p>The papers are arranged into two series:</p>

<list>

<item>

<ref target="series1">Correspondence, 1919-1955.</ref>

</item>

<item>

<ref target="series2">Diaries, 1917-1918.</ref>

</item>

 </list>

</arrangement>

**XSLT Template\_12:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template creates a separate entry for each item in the list of series. Later in the workbook, these will be converted into hyperlinks to the associated series descriptions in the <dsc>. This is a new template.

<xsl:template match="archdesc/arrangement/list/item">

 <div style="margin-left:50pt">

 <xsl:apply-templates/>

 </div>

</xsl:template>

**HTML result\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<div style="margin-left:50pt">Correspondence, 1919-1955.</div>

<div style="margin-left:50pt">Diaries, 1917-1918</div>



***Lesson 12: Back to the <did>***

**Desired Output\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Collection Summary**

**Creator:** Provenance, William Fonds, 1897-1956

**Title:** Papers of William Fonds Provenance

**Dates:**  1917-1955

**Quantity:** 1.2 cubic feet (3 boxes)

**Identification:** Mss 2 A

**Repository:** Cupcake Corners Historical Society

**Abstract:** Correspondence, diaries, and writings of an archival theorist and author documenting his experiences in World War I, his literary endeavors, and his ideas on modern archival theory, especially regarding the centrality of the fonds.

**HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_\_**

<h2>Collection Summary</h2>

<table width="100%">

<tr>

 <td width="25%"></td>

 <td width="75%"></td>

</tr>

<tr>

 <td><b>Creator:</b></td>

 <td>Provenance, William Fonds, 1897-1956</td>

</tr>

<tr>

 <td><b>Title:</b></td>

 <td>Papers of William Fonds Provenance</td>

</tr>

<tr>

 <td><b>Dates:</b></td>

 <td>1917-1955</td>

</tr>

<tr>

 <td><b>Quantity:</b></td>

 <td>1.2 cubic feet (3 boxes)</td>

</tr>

<tr>

 <td><b>Identification:</b></td>

 <td>Mss 2 A</td>

</tr>

etc.

</table>

**EAD Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<archdesc level="collection">

<did>

<head>Collection Summary</head>

<origination label="Creator:">

<persname>Provenance, William Fonds, 1897-1956</persname>

</origination>

<unittitle label="Title:">Papers of William Fonds Provenance</unittitle>

<unitdate label="Dates:" type="inclusive">1917-1955</unitdate>

<physdesc label="Quantity:">1.2 cubic feet (3 boxes)</physdesc>

<unitid label=”Identification:”>Mss 2 A</unitid>

<repository label="Repository:">

<corpname >Cupcake Corners Historical Society</corpname>

</repository>

<abstract label="Abstract:">Correspondence, diaries, and writings of an archival theorist and author documenting his experiences in World War I, his literary endeavors, and his ideas on modern archival theory, especially regarding the centrality of the fonds.</abstract>

</did>

**12.1 Transform <did> <head>**

**XSLT Template 13:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

We can add the <did> to the list of elements whose <head> is displayed in a standard way as an HTML <h2> element. This template further modifies Template 9.

<xsl:template match="**archdesc/did/head |**

 archdesc/bioghist/head |

 archdesc/scopecontent/head |

 archdesc/arrangement/head |

 archdesc/relatedmaterial/head |

 archdesc/controlaccess/head |

 archdesc/accessrestrict/head |

 archdesc/acqinfo/head |

 archdesc/processinfo/head">

 <h2>

 <xsl:apply-templates/>

 </h2>

</xsl:template>

**12.2 Transform other <did> elements and specify the order of output**

Template 14 specifies the display of the children of archdesc/did (origination, unitdate, unittitle, etc.). Unless an order for display is indicated, these elements will appear in the order in which they occur in the EAD instance. This templates modifies Template 4.

**XSLT Template 14: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template creates a shell into which the <did> child elements are inserted. First the <head> element is inserted and then an HTML table is created. The other child elements are then inserted in the order specified in a series of <xsl:apply-templates> statements.

**<xsl:template match="archdesc/did">**

<xsl:apply-templates select="head"/>

<table width="100%">

 <tr>

 <td width="25%"> </td>

 <td width="75%"> </td>

</tr>

<xsl:apply-templates select="origination"/>

<xsl:apply-templates select="unittitle"/>

<xsl:apply-templates select="unitdate"/>

<xsl:apply-templates select="physdesc"/>

<xsl:apply-templates select="unitid"/>

<xsl:apply-templates select="repository"/>

<xsl:apply-templates select="abstract"/>

</table>

</xsl:template>

**XSLT Template 15: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template formats each child element of archdesc/did by creating an HTML row with two cells for each. The text of the LABEL attribute is inserted into the first cell and the text of the element into the second.

**<xsl:template match="archdesc/did/repository**

**| archdesc/did/origination | archdesc/did/unittitle**

**| archdesc/did/unitdate | archdesc/did/physdesc**

**| archdesc/did/unitid | archdesc/did/abstract">**

 <tr valign="top">

 <td>

 <b>

<xsl:apply-templates select="@label"/>

</b>

 </td>

 <td>

<xsl:apply-templates/>

</td>

 </tr>

 </xsl:template>

***Lesson 13: Description of Subordinate Components***

***and the <c01>***

**Desired Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Detailed Description of the Collection**

The following section contains a detailed listing of the materials in the collection*.*

**Correspondence, 1919-1955. 8 folders**

Incoming letters and copies of outgoing correspondence with family, business associates, and prominent archivists and writers. Letters are arranged alphabetically by the writer’s or recipient’s name.

**Box** **Folder**

1 1 A-F

2 Gusdorf, Ida, 1942-1955

**Box** **Folder**

2 1 H-P

2 Schellenberg, Theodore

**Box** **Folder**

3 1 T-Z

**Diaries, 1917-1918. 32 v. in 5 folders**

Daily accounts of Provenance’s experiences during his military service in France during World War I, primarily documenting the daily activities of camp life, weather, military battles, and operations of the army medical service. Also contains detailed and graphic accounts of his work as an ambulance driver.

**Box** **Folder**

3 2-3 1917

 3 January-March

 3 April-December

 4-5 1918

 4 January

 5 February-June

**13.1 Formatting the <dsc> head and paragraph**

**XSLT Template 16: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The following three templates add <dsc> to the list of elements whose <head> and introductory <p> are transformed in a standard way. This template modifies Templates 9 and 13.

 **<xsl:template match="archdesc/did/head |**

 **archdesc/bioghist/head |**

 **archdesc/scopecontent/head |**

 **archdesc/arrangement/head |**

 **archdesc/relatedmaterial/head |**

 **archdesc/controlaccess/head |**

 **archdesc/accessrestrict/head |**

 **archdesc/acqinfo/head |**

 **archdesc/processinfo/head |**

 **archdesc/dsc/head">**

 **<h2>**

 **<xsl:apply-templates/>**

 **</h2>**

**</xsl:template>**

**XSLT Template H: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The wildcard, \*, replaces the names of all the child elements of <archdesc> in the previous template. This template is an alternate syntax for Template 16.

<xsl:template match="archdesc/\*/head">

 <h2>

 <xsl:apply-templates/>

 </h2>

</xsl:template>

**XSLT Template 17: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template modifies Templates 6 and 10.

<xsl:template match="archdesc/bioghist/p |

 archdesc/scopecontent/p |

 archdesc/arrangement/p |

 archdesc/relatedmaterial/p |

 archdesc/controlaccess/p |

 archdesc/accessrestrict/p |

 archdesc/acqinfo/p |

 archdesc/processinfo/p |

 **archdesc/dsc/p**">

 <p style="text-indent:25pt">

 <xsl:apply-templates/>

 </p>

</xsl:template>



**13.2 Creating the table**

###### 13.2.1 Determine the <c01> table structure

One HTML table for each <c01>

10 columns of 10% each

###### 13.2.2 Lay out the columns on the table

Determine what EAD element goes where.

In what column does each element start?

How many columns does each element span?

######  Specify the columns for the <c01> child elements

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| c0l/didc0l/scopecontent c02box c03box | colspan =10Indent first line c02 folder  c03 folder | colspan =10 c02rest of did<td></td**>** | colspan=8 c03 rest of did | colspan=7 |  |  |  |  |  |

 **HTML Encoding Required\_\_\_\_\_\_\_\_\_\_\_\_**

<table width="100%">

<tr>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

<td width="10%"> </td>

</tr>

</table>

**XSLT Template 18:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template creates a table of ten columns for each <c01> element with each column representing 10% of the screen width. It concludes by applying all templates for elements that are children of <c01>. This is a new template.

<xsl:template match="c01">

 <table width="100%">

 <tr>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 </tr>

 <xsl:apply-templates/>

 </table>

</xsl:template>

**XSLT Template 19:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template creates a row in the <c01> table for <did> and inserts its <unittitle>, <unitdate>, and <physdesc> elements. The use of the select attribute on the apply-templates statement results in these three elements, and only these three, being dislayed even if others are present in the finding aid. The data will appear in the sequence of the apply-template statements. This is one method for creating an output order that differs from the sequence of the data in the finding aid. This a new template.

<xsl:template match="c01/did">

 <tr>

 <td colspan="10">

 <b>

 <xsl:apply-templates select="unittitle"/>

 <xsl:apply-templates select="unitdate"/>

 <xsl:apply-templates select="physdesc"/>

 </b>

 </td>

 </tr>

</xsl:template>



**13.3 Adding whitespace**

Previously, we used <xsl:text> to insert text into our output. Here we use it to insert blank spaces (white space) into the output file.

**<xsl:text>**

Definition: Delimits text that is added to the output tree that is not part of the source document. The content of this element is not parsed.

Key attributes: None

**XSLT Template\_20: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

If the finding aid does not include spaces within the <unittitle> and <unitdate> elements, the text of these elements and <physdesc> will run together when transformed, as shown in the previous screen print. This template tests to see if there are both <unittitle> and <unitdate> elements. If there is, it inserts one blank space between them by using the character reference for a space, &#160;. If there is a <physdesc> element, it similarly inserts a blank space after <unitdate>. This template modifies Template 21.

<xsl:template match="c01/did">

 <tr>

 <td colspan="10">

 <b>

 <xsl:apply-templates select="unittitle"/>

 <xsl:if test="unittitle and unitdate">

 <xsl:text>&#160;</xsl:text>

 </xsl:if>

 <xsl:apply-templates select="unitdate"/>

<xsl:if test="physdesc">

 <xsl:text>&#160;</xsl:text>

 </xsl:if>

 <xsl:apply-templates select="physdesc"/>

 </b>

 </td>

 </tr>

</xsl:template>

****

**13.4 Adding the component scope notes**

**XSLT Template 21:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template creates a row for the <scopecontent><p> in <c01>. The text spans all ten columns but is indented 25 points. This is a new template.

<xsl:template match="c01/scopecontent/p">

 <tr>

 <td colspan="10" style="text-indent:25pt">

 <xsl:apply-templates/>

 </td>

 </tr>

</xsl:template>

**Browser Display for Templates 18-21**



***Lesson 14: Named and Called Templates***

**14.1 Named templates**

A given set of XSLT code may appear in more than one place in a stylesheet. These multiple occurrences may be consolidated into one template and invoked at multiple places in the stylesheet at the time a transformation occurs.

<xsl:template>

Definition: defines a template for producing output

Key attributes:
match—specifies the elements or attribute nodes in the source document to which the template applies. Establishes the context.

name—assigns a name to the template

A named template creates a result node that is not immediately inserted into the output tree but can be invoked at some other point in the stylesheet.

**14.2 Call-template**

**<xsl:call-template>**

Definition: invokes a named template

Key attribute:
name—name of the previously defined template that is called

**XSLT Template 22: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template defines a named template that has been arbitrarily named "component-did". It contains the standard instructions for formatting a <did> element at the component level. This same stylesheet code will be inserted into the templates for <did> at the <c01>, <c02>, and <c03> levels. This template uses the body from Template 20.

<xsl:template name="component-did">

 <xsl:apply-templates select="unittitle"/>

 <xsl:if test="unittitle and unitdate">

 <xsl:text>&#160;</xsl:text>

 </xsl:if>

 <xsl:apply-templates select="unitdate"/>

 <xsl:if test="physdesc">

 <xsl:text>&#160;</xsl:text>

 </xsl:if>

 <xsl:apply-templates select="physdesc"/>

</xsl:template>

**XSLT Template\_23: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The <xsl:call-template> statement in this template inserts the contents of the named template, "component-did", into the row that displays the <c01> <did>. This template replaces parts of Template 19.

<xsl:template match="c01/did">

 <tr>

 <td colspan="10">

 <b>

 <xsl:call-template name="component-did"/>

 </b>

 </td>

 </tr>

</xsl:template>

***Lesson 15: Adding more components***

**Desired Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Detailed Description of the Collection**

The following section contains a detailed listing of the materials in the collection*.*

**Correspondence, 1919-1955. 8 folders**

Incoming letters and copies of outgoing correspondence with family, business associates, and prominent archivists and writers. Letters are arranged alphabetically by the writer’s or recipient’s name.

**Box** **Folder**

1 1 A-F

2 Gusdorf, Ida, 1942-1955

**Box** **Folder**

2 1 H-P

2 Schellenberg, Theodore

**Box** **Folder**

3 1 T-Z

**Diaries, 1917-1918. 32 v. in 5 folders**

Daily accounts of Provenance’s experiences during his military service in France during World War I, primarily documenting the daily activities of camp life, weather, military battles, and operations of the army medical service. Also contains detailed and graphic accounts of his work as an ambulance driver.

**Box** **Folder**

3 2-3 1917

 2 January-March

 3 April-December

 4-5 1918

 4 January

 5 February-June

**15.1 Formatting the <c02>**

**XSLT Template 24:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template creates a row for the <c02><did> that is three columns wide. In the first column, it inserts the contents of the first <container> element, the box number. In the second column, it inserts the contents of the second <container> element, the folder number. The third text block, which spans the remaining 8 columns, consists of the <unittitle> and <unitdate> elements. This is a new template.

<xsl:template match="c02/did">

 <tr>

 <td>

 <xsl:apply-templates select="container[@type='Box']"/>

 </td>

 <td>

 <xsl:apply-templates select="container[@type='Folder']"/>

 </td>

 <td colspan="8">

 <xsl:call-template name="component-did"/>

 </td>

 </tr>

</xsl:template>

 **XSLT Template I:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This an alternate syntax for Template 24 using the position() function. Functions are explained in the Advanced Topics section.

<xsl:template match="c02/did">

 <tr>

 <td>

 <xsl:apply-templates select="container[1]"/>

 </td>

 <td>

 <xsl:apply-templates select="container[2]"/>

 </td>

 <td colspan="8">

 <xsl:call-template name="component-did"/>

 </td>

 </tr>

</xsl:template>

**Output from Template 24:**



**15.2 Formatting the <c03>**

**XSLT Template\_25:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template creates a row for the <c03><did> that is four columns wide. In the first column, it inserts the contents of the first <container> element, the box number. In the second column, it inserts the contents of the second <container> element, the folder number. The third column is empty. The fourth text block, which spans the remaining 7 columns, consists of the <unittitle> and <unitdate> elements as formatted in the named template, component-did. This is a new template.

<xsl:template match="c03/did">

 <tr>

 <td>

 <xsl:apply-templates select="container[@type='Box']"/>

 </td>

 <td>

 <xsl:apply-templates select="container[@type='Folder']"/>

 </td>

 <td> </td>

 <td colspan="7">

 <xsl:call-template name="component-did"/>

 </td>

 </tr>

</xsl:template>



***Lesson 16: Table of Contents***

**16.1 Purpose of the TOC**

Overview of the structure and contents of the finding aid

Quick hyper-navigation within the document

**16.2 Format options**

In-line

Frame

Table

**16.3 The Process**

**Desired Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Table of Contents

Collection Summary

Biography of William Provenance

Scope and Contents of the Papers

Arrangement of the Papers

Related Records

Index Terms

Restrictions on Access

Acquisition Information

Processing Information

Detailed Description of the Collection

**EAD source for those entries**:

Collection Summary archdesc/did/head

Scope and Contents archdesc/scopecontent/head

Biography archdesc/bioghist/head

Arrangement archdesc/arrangement/head

Related Records archdesc/relatedmaterial/head

Index Terms archdesc/controlaccess/head

Restrictions on Access archdesc/accessrestrict/head

Acquisition Information archdesc/acqinfo/head

Processing Information archdesc/processinfo/head

Detailed Description of the Collection archdesc/dsc/head

**<xsl:apply-templates>**

Definition: processes all children of the context element. If the select attribute is present, it processes templates for the node specified. Otherwise, it processes all child nodes.

Key attribute:
select—specifies particular child nodes to which templates are to be applied

mode—processes only templates with a matching mode attribute

**<xsl:template>**
Definition: prescribes an action to be performed by an XSLT processor
Key attributes:

match—specifies the elements or attribute nodes in the source document to which the template applies. Establishes the context.

mode—the template is invoked only by an <xsl:apply-templates> statement with a matching mode attribute

**XSLT Template\_26: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**

This template formats the <head> elements that form the table of contents as HTML paragraphs. Compare this template to Templates 5, 9, and 16. This is a new template.

 <xsl:template match="archdesc/\*/head" mode="toc">

<p style="text-indent:25pt">

 <b>

 <xsl:apply-templates/>

 </b>

</p>

 </xsl:template>

**XSLT Template 27: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Template 26 creates the heading for the table of contents. It then applies the templates for those elements that have the value "toc" in their the mode attribute. The presence of this attribute distinguishes the templates that govern the display of the <head> elements in this context from those that govern the display of <head> elements in the body of the finding aid. This template modifies Templates 1 and 2.

**<xsl:template match="ead">**

**<html>**

**<head>**

**<title></title>**

**</head>**

**<body>**

<h2>

<xsl:text>Table of Contents</xsl:text>

</h2>

<xsl:apply-templates select=

"archdesc/\*/head" mode="toc"/>

**<xsl:apply-templates/>**

**</body>**

**</html>**

**</xsl:template>**

****

**XSLT Template J: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template specifies and therefore limits the head elements that appear in the Table of Contents. They will display in the sequence given. This template is an alternate syntax for Template 27.

<xsl:template match="ead">

<html>

<head>

<title></title>

</head>

<body>

<h2>

 <xsl:text>Table of Contents</xsl:text>

</h2>

<xsl:apply-templates select="archdesc/did/head" mode="toc"/>

<xsl:apply-templates select="archdesc/scopecontent/head" mode="toc"/>

<xsl:apply-templates select="archdesc/bioghist/head" mode="toc"/>

<xsl:apply-templates select="archdesc/controlaccess/head" mode="toc"/>

<xsl:apply-templates/>

</body>

</html>

</xsl:template>

**16.4 Insert the finding aid title before the Table of Contents**

**XSLT Template\_28: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template causes the "title page" elements from <eadheader> to appear before the table of contents. This template modifies Templates 1, 2, and 27.

**<xsl:template match="ead">**

**<html>**

**<head>**

**<title></title>**

**</head>**

**<body>**

<xsl:apply-templates select="eadheader"/>

**<h2>**

 **<xsl:text>Table of Contents</xsl:text>**

**</h2>**

**<xsl:apply-templates select="archdesc/\*/head" mode="toc"/>**

**<xsl:apply-templates/>**

**</body>**

**</html>**

**</xsl:template>**

**XSLT Template\_29: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The previous template generated an unintended result—the "title page" data appears twice: once because of the <xsl:apply-templates select="eadheader"/> statement and the second time because of the general <xsl:apply-templates/> statement.The following template corrects the problem by excluding <eadheader> elements from the general <xsl:apply-templates/> statement. It now reads in effect—apply all templates for children of the context element, <ead>, that are not <eadheader>. This template modifies Templates 1, 2, 27, and 28.

**<xsl:template match="ead">**

**<html>**

**<head>**

**<title></title>**

**</head>**

**<body>**

**<xsl:apply-templates select="eadheader"/>**

**<h2>**

 **<xsl:text>Table of Contents</xsl:text>**

**</h2>**

**<xsl:apply-templates select="archdesc/\*/head" mode="toc"/>**

**<xsl:apply-templates** select="\*[not(self::eadheader)]"/>

**</body>**

**</html>**

**</xsl:template>**



***Advanced Topics***

***Lesson 17: Linking***

**17.1 Principles of linking**

1. Links are used in three ways in EAD.

* To embed an external file such as an image so that it appears at a particular place in the finding aid. Such an image is said to be "inline."
* To move the reader away from the finding aid to a different web page, file, or a computer application such as an image viewer.
* To move the reader from one place in the finding aid to another.

2. Links have two parts: a source and a target.

3. In finding aids, the source of the link is found in an EAD element. Two elements are employed as the source of links: Digital Archival Object <dao> and Reference <ref>.

* <dao is used when the link is to a digital representation of some portion of the archival materials being described.
* <ref> is used for all other type of links.

4. The source element has the address (location) of the target as the value of one of its attributes. The attribute involved varies with the type of link.

 <ref href="www.mnhs.org/findingaids/247.html"/>

**17.2 Linking scenarios**

There are four scenarios (and variations) for links in EAD. You must determine which of the following situations describes the result you wish to achieve with a link in order to determine which EAD encoding must be added to your markup and which XSLT template must be used to correctly transform your data into the appropriate HTML or other output.

1. Cause a digital representation of collection material to appear in-line in the finding aid, such as a scanned image of a document.

2. Cause a digital representation of non-collection material to appear in-line in the finding aid, such as an institutional logo or picture of the creator of the collection.

3. Create a link that, when clicked, transports the user to an another file, web site, or computer application such as an image viewer.

4. Create a link from one place in a finding aid to another, such as from a list of series in a statement about the arrangement of the collection to a detailed description of a given series.

**17.3 Applying the linking scenarios**

**Scenario 1:** Cause a digital representation of collection material to appear in-line in the finding aid, such as a scanned image of a document.

**EAD Encoding**

* <dao> is the link source
* The HREF attribute contains the address (uri) of the target
* The value "embed" in the SHOW attribute specifies that the image is to appeat inline.
* <dao> includes the <daodesc> child to provide a caption.

<c4 level="item">

 <did>

 <dao href="c:\findaids\6378\12-1-40a.jpg" show="embed">

 <daodesc>Letter to Ima Gusdorf</daodesc>

 </dao>

**XSLT Template\_K: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="dao[@show='embed']">

<img src="{@href}" style="text-align:center">

 <xsl:apply-templates/>

</img>

</xsl:template>

**HTML Output**

* The HTML output is an Image <img> element.
* The value of the HREF attribute in the source <dao> element becomes the value of the SRC attribute in the output.
* The image is centered on the page using a STYLE attribute.

<img src="c:\findaids\6378\12-1-40a.jpg" style="text-align:center">

Letter to Ima Gusdorf

</img>

**Scenario 2:**  Cause a digital representation of non-collection material to appear in-line in the finding aid, such as an institutional logo or a picture of the creator of the collection.

**EAD Encoding**

* <ref> is the link source
* The HREF attribute contains the address (uri) of the target
* The value "embed" in the SHOW attribute specifies that the image is to appeat inline.
* <ref> includes PCDATA to provide a caption.

<biography>

 <ref href="c:\findaids\6389\wandrei.jpg" show="embed">

 Portrait of Donald Wandrei by Syd Fossum

 </ref>….

**XSLT Template\_L: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="ref[@show='embed']">

<img src="{@href}" style="text-align:center">

 <xsl:apply-templates/>

</img>

</xsl:template>

**HTML Output**

* The HTML output is an Image <img> element.
* The value of the HREF attribute in the source <ref> element becomes the value of the SRC attribute in the output.
* The image and its text are centered on the page using a STYLE attribute.

<img src="c:\findaids\6389\wandrei.jpg" style="text-align:center">

 Portrait of Donald Wandrei by Syd Fossum

</img>

**Scenario 3:** Create a link, that when clicked, transports the user to an another file, web site, or computer application such as an image viewer.

**EAD Encoding**

* <ref> is the link source
* The HREF attribute contains the address (uri) of the target
* The value "new" in the SHOW attribute specifies that the browser is to open a new page.
* <ref> includes PCDATA to provide a caption for the link.

<p>Provenance was a close friend of archival pioneer

 <ref href = "http://en.wikipedia.org/wiki/Margaret\_C.\_Norton" show="new">

 Margaret Cross Norton

 </ref>, Illinois State Archivist and co-founder of the Society of American Archivists.

</p>

**XSLT Template\_M: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="ref[@show='new']">

<a href="{@href}">

<xsl:apply-templates/>

 </a>

</xsl:template>

HTML Output

* The HTML output is an Anchor <a> element.
* The value of the HREF attribute in the source <ref> element becomes the value of the SRC attribute in the output.
* The link appears in-line in the text of the paragraph.

<a href = "http://en.wikipedia.org/wiki/Margaret\_C.\_Norton">

 Margaret Cross Norton

</a>

**Scenario 4:**  Create a link from one place in a finding aid to another, such as from a list of series in a statement about the arrangement of the collection to a detailed description of a given series.

This link is more complex because one must encode both the source and target of the link within the finding aid and create two templates: one for the source and one for the target.

EAD Encoding- Source:

* <ref> is the link source
* The TARGET attribute contains the address (the ID attribute) of the target
* <ref> includes PCDATA to provide text for the link.

<list>

 <item>
 <ref target="**series1**">Correspondence, 1919-1955.</ref>

**XSLT Template\_30: Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<xsl:template match="ref'[@target]">

<a href="#{@target}">

<xsl:apply-templates/>

 </a>

</xsl:template>

HTML Output—Source:

* The template is qualified to apply only to <ref> elements with a TARGET attribute, a situation that is unique to Option 4.
* The HTML output is an Anchor <a> element.
* The value of the TARGET attribute in the source <ref> element becomes the value of the HREF attribute in the output.
* The link appears in-line in the text of the paragraph.
* The inclusion of the hash sign (#) indicates to the browser that the target is in the same document as the source of the link.

<a href="#**series1**">

Correspondence, 1919-1955

</a>

EAD Encoding—Target:

The ID attribute of the target is used as the unique identifier for the destination of the link.

The ID attribute of the target is the value of the TARGET attribute in the source element.

<c01 level="series" id="**series1**">

 <did>….

**XSLT Template\_31: Target\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template modifies Template 18.

<xsl:template match="c01">

 <table width="100%">

 <tr>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 <td width="10%"> </td> <td width="10%"> </td>

 <a name="{@id}"/>

 </tr>

<xsl:apply-templates/>

</table>

</xsl:template>

HTML Output—Target

* The HTML output is an Anchor <a> element..
* The value of the NAME attribute is that of the ID attribute of the target element.

<a name="series1">… </a>

**17.4 Linking from the Table of Contents**

Linking from the Table of Contents to the body of the finding aid is a variation on Scenario 4 in the preceeding section. Initially, it would seem to be just another instance of linking from one place in the finding aid to another. However, it is more complex than that.

In Option 4, the source element was encoded into the text of the finding aid. However, we cannot do that with the Table of Contents because it does not actually exist in the finding aid. Rather it is generated by the stylesheet at the time of transformation. As a result, the source of the link must be built into the stylesheet rather than the EAD instance. The destination of the link continues to be designated by the ID attribute of the target.

EAD Encoding—Source:

There is none. It is generated by the stylesheet.

**XSLT Template\_32: Source\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template modification turns each element in the Table of Contents into a link source by making it an HTML anchor element whose href attribute is the value of a unique id for that <head> element in the body of the finding aid. This template modifies Template 26.

<xsl:template match="archdesc/\*/head" mode="toc">

<p style="text-indent:25pt">

 <b>

<a href="#{id}">

 <xsl:apply-templates/>

</a>

 </b>

</p>

</xsl:template>

HTML Output—Source

This example is taken from the <bioghist> element.

<a href="#t2">Biography of William Provenance</a>

EAD Encoding—Target

The ID attribute of the target is used as the unique identifier for the destination of the link.

The ID attribute of the target is the value of the TARGET attribute for the source as specified in the stylesheet.

<bioghist>

 <head id="t3>

 <p>….

**XSLT Template 33: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The following template modification converts each <head> element in the body of the finding aid into the target of a link from the Table of Contents by making it an HTML anchor element whose name attribute is the value of the id attribute for that <head>. The unique value is assigned during EAD markup. This template modifies Templates 16 and H.

<xsl:template match="archdesc/\*/head">

<h2>

<a name="{id}">

<xsl:apply-templates/>

</a>

</h2>

</xsl:template>

HTML Output—Target

<h2>

 <a name="table3">

 Biography of William Provenance

 </a>

</h2>

***Lesson 18: Functions***

**18.1 What are functions?**

XSLT and XPath specify 32 tasks called functions (computer routines) that transformation applications should be able to perform.

Some functions test for information about the context of a node.

 position()

 last()

Some functions perform mathematical operations

 count()

 number()

Some functions manipulate text strings

 contains()

 string()

 starts-with()

 translate()

Some functions perform Boolean tests

 false()

 true()

 not()

Some functions return a value within the document

generate-id()

text()

current()

**18.2 How functions work**

Functions search the text of a document and return a value to the stylesheet. That value may be a number, true or false, the value of an element, or text.

**Examples:**

1. not()

<xsl:apply-templates select="title[not(parent::chapter)]"/>

This template fragment matches titles that do not have chapter as their parent.

Functions may include an argument in the parentheses. The argument may be a text string or an XPath expression.

Which of these nodes does the expression above match?

**book/title**

**section/title**

**chapter/title**

2. position()=last()

position() returns the number of the position of the current element with respect to its siblings

last() returns the number of the position of the last sibling

<xsl:template match="chapter">

*[Applies some formatting instructions and transforms the chapter element]*

<xsl:if test="position()=last()">

<xsl:text>The End</xsl:text>

</xsl:if>

</xsl:template>

3. generate-id(c01)

assigns a unique id value to each <c01>

instead of encoding the value of an ID attribute

<c01 id="series1">......

the stylesheet assigns one

<c01 id="IDAG2W1">......

4. <xsl:if test="string(archdesc/bioghist/head)">

 tests to see if there is a bioghist/head present and if it has content

***Lesson 19: Expressions***

**19.1 XPath expressions**

Expressions provide a richer syntax for identifying a particular node than the match pattern syntax from XSL that we have seen in templates thus far.

XPath is shared by XSLT and XLink.

Expressions appear in

* the select attribute of

<xsl:apply-templates>

<xsl:value-of>

* the test attribute of

<xsl:if>

<xsl:when>

* in predicates (qualifiers)
* in functions

**19.2 Steps and step separators**

 <xsl:template match=”c01/did/container [@type=’Box’]”>

* Each part of the path is called a step.
* Each step delimited by a step separator—a forward slash or double slashes.
* Each step specifies a particular node.
* Each step may have three parts:

Axis

Node test

Step Qualifier (Predicate)

**19.3 Axes**

Axes indicate the direction up, down, and sideways on the tree as we move from a specified element to some other element. Up to now, we have been traveling only down the tree, from parent to child.

<xsl:template match=”c01/did/container [@type=’Box’]”>

**19.4 Syntax for axes**

In the match patterns we have used up to now, the child axis has been implied for each step. The full syntax for the match pattern above is actually this:

<xsl:template match=”c01/child::did/child::container [attribute::type=’Box’]”>

The double colons are read as: "has the name."

"child::did" means the "the child element that has the name did"

There is an abbreviated syntax for five commonly used axes:

**Abbreviated syntax** **Full syntax example**

**.** self::node ( ) (the context node)

**..** parent::*node name*

title child::title (default value)

@numberattribute::number

// descendant-or-self::*node name*

**19.5 13 basic axes**

child

 child node of the context node. This is the default axis.

parent

 parent of the context node, if there is one

ancestor

 parent node and the parent’s parent node, etc. back to the root node

descendant

 child or a child of a child and so on

following-sibling

 all sibling nodes following the context node

following

 all nodes in document order after the context node

preceding-sibling

 all preceding siblings of the context node

preceding

all nodes in the same document that are before the conext node in document order

self

 the context node

attribute

 attributes of the context node

ancestor-or-self

 context node or an ancestor

descendant-or-self

 context node and its descendants

namespace

 namespace node of the context element

**The following pages contain graphic illustrations of some of the different axes available in XSLT in relation to the context element in a SELECT attribute.**









**19.6 Examples of axes and their syntax**

1. **child::p** selects the p element children of the context node
2. **child::\*** selects all element children of the context node
3. **child::text()** selects all text node children of the context node
4. **child::node()** selects all the children of the context node, whatever their node type
5. **attribute::name** selects the name attribute of the context node
6. **attribute::\*** selects all the attributes of the context node
7. **descendant::p** selects the p element descendants of the context node
8. **ancestor::c01** selects all c01 ancestors of the context node
9. **ancestor-or-self::c02** selects the c02 ancestors of the context node and, if the context node is a c02 element, the context node as well
10. **descendant-or-self::p** selects the p element descendants of the context node and, if the context node is a p element, the context node as well
11. **self::p** selects the context node if it is a p element, and otherwise selects nothing
12. **child::bioghist/descendant::p** selects the p element descendants of the bioghist element children of the context node
13. **child::\*/child::p** selects all p grandchildren of the context node
14. **/** selects the document root (which is always the parent of the document element)
15. **/descendant::p** selects all the p elements in the same document as the context node
16. **/descendant::list/child::item** selects all the item elements that have an list parent and that are in the same document as the context node
17. **child::p[position()=1]** selects the first p child of the context node
18. **child::p[position()=last()]** selects the last p child of the context node
19. **child::p[position()=last()-1]** selects the last but one p child of the context node
20. **child::p[position()>1]** selects all the p children of the context node other than the first p child of the context node

**19.7 Putting Expressions and Functions to Work**

Assume we want to retrieve the value of the type attribute of the first container of a <c02> element that is a file in order to display that text as the display head at the top of a column of box numbers. In the following example, that value is the word "Box".

<c02 level="file">

 <did>

 <container type="Box">1</container>

 <container type="Folder">2</container>

 </did>

</c02>

The expression in the SELECT attribute of the following XSLT code would select that value.

<xsl:template match="c02">

 …

 **<xsl:apply-templates select=**

 **"did[parent::c02/@level='file']**

 **/container[position()=1]**

 **/@type"/>**

 …

</xsl:template>

There are four possible components of an expression, as illustrated by the previous example:

 **Steps**: In the example above, there are four steps, each separated by a forward slash:

 **Axes:**

Threesteps move down the child axis: @level, container, and @type

 (if no axis is specified, child axis is the default)

 One step moves up the parent axis: parent::c02

 **Node name:** each of the three steps is a separate node on the tree.

 did, container, @type

 **Qualifiers**: two nodes are qualified

 [parent::c02/@type='file'] if the parent of the did is a c02 that has the level file.

 This qualifier actually contains two steps within itself with two node names- up to c02 and down to its type attribute.

 [position()=1] if the container is the first container to appear in the <did>

***Lesson 20: Manipulating Columns and Container Information***

**Desired Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Correspondence, 1919-1955. 8 folders**

Incoming letters and copies of outgoing correspondence with family, business associates, and prominent archivists and writers. Letters are arranged alphabetically by the writer’s or recipient’s name.

**Box** **Folder**

1 1 A-F

 2 Gusdorf, Ida, 1942-1955

**Box** **Folder**

2 1 H-P

 2 Schellenberg, Theodore

**Box** **Folder**

3 1 T-Z

**Diaries, 1917-1918. 32 v. in 5 folders**

Daily accounts of Provenance’s experiences during his military service in France during World War I, primarily documenting the daily activities of camp life, weather, military battles, and operations of the army medical service. Also contains detailed and graphic accounts of his work as an ambulance driver.

**Box** **Folder**

3 2-3 1917

 2 January-March

 3 April-October-December

 4-5 1918

 4 January

 5 February-June

**20.1 Display options**

First issue—how to insert columns heads like "Box" and "Folder"

1. EAD encoding can specify where headings appear and what they say through the <thead> element. OR

2. Alternatively, stylesheets can supply both the placement and content of the headings. It may be useful to have headings at regular intervals (as text rolls off the screen).

Second issue—displaying all container values…or not

1. Display all containers as encoded. There might be containers for every component or only some. OR

2. Display containers selectively.

a. Suppress duplicates

b. Suppress duplicates but supply with each new <c01>

**20.2 Markup for container elements**

Q: When do the box numbers display in the example on the previous page?

A: Whenever the box number appears for the first time in a given <c01>.

Q: When is there a new column heading in the example on the previous page?

A: Whenever the box number appears for the first time in a given <c01>.

**There are two scenarios:**

Scenario A: The box number is new or there is a new series (c01). Display the box number and insert the headings on the previous line.

Scenario B: A given box number has already appeared. No box number is displayed or column headings inserted.

**20.3 Implementing displays through the stylesheet**

Q: How do we get the stylesheet to implement Scenario A or Scenario B?

A: For each component, we look back up the tree to see if a previous box had the same number.

**Learn how to do this in the next lesson.**

***Lesson 21: Conditional Statements***

**21.1 Multiple conditions: making complex choices**

**<xsl:choose>**
Definition: initiates a condition
Key attributes: None

**<xsl:when>**
Definition: specifies one option in a series of possibilities
Key attribute:

test—an expression that specifies the condition being tested for

**<xsl:otherwise>**
Definition: specifies the last option in a series of possibilities
Key attributes: None

Use <xsl:choose> when there is more than one possibility. It is equivalent to if…then…else statements in other programming languages.

**21.2 Applying options to a finding aid**

**21.2.1 Use a multiple condition test**

<xsl:choose>

 <xsl:when test="Scenario B">

 Do encoding option B--don’t display and don't insert a heading

 </xsl:when>

 <xsl:otherwise>

 Do encoding option A--insert head and display box number

 </xsl:otherwise>

</xsl:choose>

**21.2.2 Writing an expression for the test statement**

1. Determine the situations when each condition might occur.

2. Express those situations as logical statements in XPath syntax.

The following expression uses the preceding axis to text as to whether or not a given box number appears for the first time in a particular component. It does so by comparing the value of its box number with that of the box number of the previous component.

**xsl:test="**

**container[@type='Box']=preceding::did[1]/container[@type='Box']"**

Syntax of the axis:

container the <container> in the component being processed

[@type='Box'] <container> is qualified to one whose type attribute is Box

preceding::did checks all preceding <did> elements in the document

[1] limits the search for preceding <did> elements to the first one

container the <container> is that of the first preceding did

[@type='Box'] <container> is qualified to one whose type attribute is Box

**EAD Markup\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

<c01>

<did>

<unittitle>Correspondence,</unittitle>

<unitdate type="inclusive">1919-1955.</unitdate>

<physdesc>8 folders</physdesc>

</did>

<c02>

<did>

<container type="Box">1</container>

<container type="Folder">1</container>

<unittitle>A-F</unittitle>

</did>

</c02>

<c02>

<did>

<container type="Box">1</container>

<container type="Folder">2</container>

<unittitle>Gusdorf, Ida</unittitle>

</did>

</c02>

<c02>

<did>

<container type="Box">1</container>

<container type="Folder">3</container>

<unittitle> Gusdorf, Ida, 1942-1955</unittitle>

</did>

</c02>

<c02>

<did>

<container type="Box">2</container>

<container type="Folder">1</container>

<unittitle>H-P</unittitle>

</did>

</c02>

**Desired Results\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Correspondence, 1919-1955. 8 folders**

Incoming letters and copies of outgoing correspondence with family, business associates, and prominent archivists and writers. Letters are arranged alphabetically by the writer’s or recipient’s name.

**Box** **Folder**

1 1 A-F

 2 Gusdorf, Ida

**Box** **Folder**

2 1 H-P

 2 Schellenberg, Theodore

**XSLT Template\_34: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template modifies Template 24.

<xsl:template match="c02/did">

<xsl:choose>

 <xsl:when test="container[@type='Box']=

 preceding::did[1]/container[@type='Box']">

<tr>

 <td> </td>

 <td>

 <xsl:apply-templates select="container[@type='Folder']"/>

 </td>

 <td colspan="8">

 <xsl:call-template name="component-did"/>

 </td>

 </tr>

</xsl:when>

 <xsl:otherwise>

<tr>

 <td>

 <b>

 <xsl:text>Box</xsl:text>

 </b>

 </td>

 <td>

 <b>

 <xsl:text>Folder</xsl:text>

 </b>

 </td>

 </tr>

 <tr>

 <td>

 <xsl:apply-templates select="container[@type='Box']"/>

 </td>

 <td>

 <xsl:apply-templates select="container[@type='Folder']"/>

 </td>

 <td colspan="8">

 <xsl:call-template name="component-did"/>

 </td>

 </tr>

 </xsl:otherwise>

 </xsl:choose>

</xsl:template>

**XSLT Template 35: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

This template modifies Template 25.

<xsl:template match="c03/did">

<xsl:choose>

 <xsl:when test="container[@type='Box']=

 preceding::did[1]/container[@type='Box']">

<tr>

 <td> </td>

 <td>

 <xsl:apply-templates select="container[@type='Folder']"/>

 </td>

 <td></td>

 <td colspan="7">

 <xsl:call-template name="component-did"/>

 </td>

 </tr>

</xsl:when>

<xsl:otherwise>

<tr>

 <td>

 <b>

 <xsl:text>Box</xsl:text>

 </b>

 </td>

 <td>

 <b>

 <xsl:text>Folder</xsl:text>

 </b>

 </td>

 </tr>

 <tr>

 <td>

 <xsl:apply-templates select="container[@type='Box']"/>

 </td>

 <td>

 <xsl:apply-templates select="container[@type='Folder']"/>

 </td>

 <td></td>

 <td colspan="7">

 <xsl:call-template name="component-did"/>

 </td>

 </tr>

 </xsl:otherwise>

 </xsl:choose>

</xsl:template>

 ***Lesson 22: Beautifying the Title Page***

**XSLT Template 36: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The following template inserts the institutional logo and several blank lines to make the "title page" more attractive. This template modifies Template 29.

<xsl:template match="ead">

 <html>

 <head><title></title></head>

 <body>

 <center>

 <img src="cchslogo.gif"/>

 </center>

 <br></br>

 <br></br>

 <xsl:apply-templates select="eadheader"/>

 <br></br>

 <br></br>

 <h2>

<xsl:text>Table of Contents</xsl:text>

 </h2>

 <xsl:apply-templates select="archdesc/\*/head" mode="toc"/>

 <br></br>

 <br></br>

 <xsl:apply-templates select="\*[not(self::eadheader)]"/>

 </body>

 </html>

</xsl:template>



***Lesson 23: Modularization***

###### Modularization by importing another external stylesheet:

Create a full, separate stylesheet containing part of your XSLT code.

Call it in another stylesheet at the point where you want it to execute.

Works as if the inserted stylesheet were actually written there.

Use the <xsl:import> or <xsl:include> element.

**<xsl:import>**

Definition: inserts the contents of one stylesheet into another

Key attribute:

 href—contains the URI of the imported stylesheet

**<xsl:include>**

Definition: inserts the contents of one stylesheet into another

Key attribute:

 href—contains the URI of the imported stylesheet

Notice that the definitions of the two are identical.

Sometimes it is possible that more than one template might apply to a given node on the source tree. XSLT has elaborate rules to determine which one applies in the case of such conflicts.

The only difference between <xsl:include> and <xsl:import> relates to such tie-breaking rules. If you use <xsl:include>, the order of precedence is the same for rules in original and in the included stylesheet. With <xsl:import>, templates in the stylesheet being imported take precedence.

***Alternate Formatting Syntax***

**EAD Encoding\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**<scopecontent>**

**<p>**The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance**. </p>**

**</scopecontent>**

**1. Transform from XML to HTML**

<xsl:template match="archdesc/scopecontent/p">

 **<p style="text-indent:25 pt.; font-family:times>**

<xsl:apply-templates/>

 **<p>**

</xsl:template>

**Produces:**

**<p style="text-indent:25 pt.; font-family:times>**The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance**. <p>**

**2. Transform into plain text**

<xsl:template match="archdesc/scopecontent/p">

 **<xsl:apply-templates/>**

</xsl:template>

**Produces:**

The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance**.**

**3. Transform from one XML format to another: EAD to MARCXML**

<xsl:template match="archdesc/scopecontent/p"

**<marc:data-field tag="520" ind1="2" ind2="&#160;">**

**<marc:subfield code="a">**

<xsl:apply-templates/>

**</marc:subfield>**

**</marc:data-field>**

</xsl:template>

**Produces this XML:**

**<marc:data-field tag="520" ind1="2" ind2="&#160;">**

**<marc:subfield code="a">**The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance**.**

**</marc:subfield>**

**</marc:data-field>**

**4. Convert from XML to XSL:FO**

<xsl:template match="archdesc/scopecontent/p">

**<fo:block text-indent="25 pt." font-family="times" font-size="10 pt."
line-height="14 pt." font-variant="small-caps" space-after="0.4 cm.">**

<xsl:apply-templates/>

**</fo:block>**

</xsl:template>

**Produces this XML-FO:**

**<fo:block text-indent="25 pt." font-family="times" font-size="10 pt."**

**line-height="14 pt." font-variant="small-caps" space-after="0.4 cm.">**

The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance**.**

**</fo:block>**

**Which results in this PDF display when manipulated by a format objects processor:**

The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance**.**

**5. Convert EAD XML to Word Markup Language**

<w:styles>

 <w:style w:type="paragraph" w:default="on" w:styleId="normal">

 <w:name w:val="normal"/>

 <w:rPr>

 <wx:font wx:val="Times New Roman"/>

 <w:sz w:val="24"/>

 </w:rPr>

 </w:style>

</w:styles>

<xsl:template match="archdesc/scopecontent/p">

<w:p>

 <w:pPr>

<w:pStyle w:val="normal"/>

<w:ind w:left="720"/>

</w:pPr>

<w:r>

 <w:t>

 <xsl:apply-templates/>

 </w:t>

 </w:r>

</w:p>

<xsl:template>

**Produces this Microsoft Word text:**

The collection consists of diaries, correspondence, manuscripts, and miscellaneous materials documenting the literary and archival career of William Fonds Provenance**.**

**Quick List of XSLT Elements and Attributes Used in the Workshop**

**<xsl:apply-templates>** Processes all children of the context element. If the select attribute is present, it processes templates for the node specified. Otherwise, it processes all child nodes.

select—specifies particular child nodes to which templates are to be applied

mode—processes only templates with a matching mode attribute

**<xsl:call-template>** Invokes a named template.

name—name of the previously defined template that is called

**<xsl:choose>** Initiates a condition.

**<xsl:for-each>**  Creates a loop through successive instances of an element.

select—specifies the node to be returned

**<xsl:if>** Test for a conditon specifed in a test attribute.

test—an expression that specifies the condition being tested for

**<xsl:otherwise>** Specifies the last option in a series of possibilities.

**<xsl:output>** Specifies the type of output.

method—specifies one of three options: XML, HTML, text

indent—if the value is "yes," white space is inserted into the output file to provide an indented presentation

**<xsl:sort>** Defines the sequence for processing nodes.

order—specifies whether nodes are prcessed in ascending or descending order

data-type—specifies whether the values are to be collated alphabetically or numerically

select—specifies the sort key

**<xsl:strip-space>**  Removes white space nodes from the source document.

elements—specifies the elements from which white space is removed

**<xsl:stylesheet>**  The document element of the stylesheet. Like <ead>.

version—specifies the version of XSLT utilized (mandatory)

xmlns—declares that the elements having a particular prefix belong to a particular namespace

**<xsl:template>** Prescribes an action to be performed by an XSLT processor.

match—specifies the elements or attribute nodes in the source document to which the template applies. Establishes the context.

mode—the template is invoked only by an <xsl:apply-templates> statement with a matching mode attribute

name—assigns a name to the template

**<xsl:text>**Delimits text that is added to the ouput tree that is not part of the source document. The content of this element is not parsed.

**<xsl:when>** Specifies one option in a series of possibilities.

test—an expression that specifies the condition being tested for

**XSLT and XPath Symbols**

/ (forward slash) separates steps in a path statement, e.g., chapter/p

\* wild card that can stand for any element

@ shorthand for attribute, e.g., @type is a type attribute

[ ] an expression in square brackets is a qualifier for the preceding element or attribute

**::** in an XPath expression, means "that is defined as"

**..** in a XPath expression, stands for the parent node

**.** in an XPath expression, the context node

// in an XPath expression, descendant-or-self

( ) part of the syntax of a function, can contain an argument

{ } curly brackets are used when an attribute value is generated at the time of output

$ indicates the name of a variable

Operators

= equals

!= does not equal

+ and

| or

**Stylesheets With Schemas**

When one uses an EAD schema rather than a DTD, one must declare and insert the EAD namespace specification before each EAD element name in the stylesheet..

<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
 xmlns:ead="urn:isbn:1-931666-22-9">
 <xsl:output method="html" indent="yes"></xsl:output>
 <xsl:strip-space elements="\*"></xsl:strip-space>

 <xsl:template match="ead:ead">
 <html>
 <head></head>
 <body>
 <center>
 <img src="cchslogo.gif"></img>
 </center>
 <br></br>
 <br></br>
 <xsl:apply-templates select="ead:eadheader"></xsl:apply-templates>
 <br></br>
 <br></br>
 <h2>
 <xsl:text>Table of Contents</xsl:text>
 </h2>
 <xsl:for-each select="ead:archdesc/ead:\*">
 <xsl:apply-templates select="ead:head" mode="toc"
 ></xsl:apply-templates>
 </xsl:for-each>
 <br></br>
 <br></br>
 <xsl:apply-templates select="\*[not(self::ead:eadheader)]"
 ></xsl:apply-templates>
 </body>
 </html>
 </xsl:template>

 <xsl:apply-templates select="ead:filedesc/ead:titlestmt/ead:titleproper"
 ></xsl:apply-templates>
 </h1>
 <h2 style="text-align:center">
 <xsl:apply-templates select="ead:filedesc/ead:titlestmt/ead:subtitle"
 ></xsl:apply-templates>
 </h2>
 </xsl:template>

 <xsl:template match="ead:archdesc/ead:did">
 <xsl:apply-templates select="ead:head"/>
 <table width="100%">
 <tr>
 <td width="25%"> </td>
 <td width="75%"> </td>
 </tr>
 <xsl:apply-templates select="ead:origination"/>
 <xsl:apply-templates select="ead:unittitle"/>
 <xsl:apply-templates select="ead:unitdate"/>
 <xsl:apply-templates select="ead:physdesc"/>
 <xsl:apply-templates select="ead:unitid"/>
 <xsl:apply-templates select="ead:repository"/>
 <xsl:apply-templates select="ead:abstract"/>
 </table>
 </xsl:template>

 <xsl:template match="ead:archdesc/ead:did/ead:repository
 | ead:archdesc/ead:did/ead:origination | ead:archdesc/ead:did/ead:unittitle
 | ead:archdesc/ead:did/ead:unitdate | ead:archdesc/ead:did/ead:physdesc
 | ead:archdesc/ead:did/ead:unitid | ead:archdesc/ead:did/ead:abstract">

 <tr valign="top">
 <td>
 <b>
 <xsl:apply-templates select="@label"/>
 </b>
 </td>
 <td>
 <xsl:apply-templates/>
 </td>
 </tr>
 </xsl:template>
 **XSLT RESOURCES**

**The Standards**

The following documents specify how XSL transformations work. They are technical publications and not intended as tutorials.

Extensible Stylesheet Language (XSL) - base document for this family of standards and the specifications for XSL-FO, used to format print outputs from XML.

 http://www,W3.org/TR/xsl

XSL Transformations--defines how an XML document maybe converted into another XML syntax, into HTML, or into text.

 http://www.w3.org/TR/xslt

XPath--defines a language by which one may specify a certain portion of XML document for purposes of transforming it.

 http://www.w3.org/TR/xpath

**Published Tutorials**

The XSL Transformation standard was finalized in November 1999, after several years of development during which its syntax underwent substantial changes. Many publications on XML currently available were published before that date and contain chapters on XSL that reflect earlier syntax and functionality and so must be read with some caution. Four exceptions are:

 *The XML 1.1 Bible.* Elliotte Harold. (John Wiley) Authoritative work on all aspects of XML that contains a useful overview of XSLT. The XSLT chapter is also available online at:

 http://www.ibiblio.org/xml/books/bible3/chapters/ch15.html

*XSLT 2.0 Programmer's Reference.* Michael Kay. (Chicago: WROX, 2004). General overview and reference manual from the creator of the popular XSLT transformation software, SAXON. (see pp. 365-367 for diagram of axes)

*Definitive XSLT and XPath*. G. Ken Holman. (Available by download from the author)

*XSLT for Dummies*. Richard Wagner. (New York: Hungry Minds, 2002)

**FAQ**

A list of frequently asked questions about XSL is available at

 http://www.dpawson.co.uk/xsl/xslfaq.html