Info Session: XSLT

Christine McEvilly, AJHS
Kevin Schlottmann, CJH

August 21, 2013

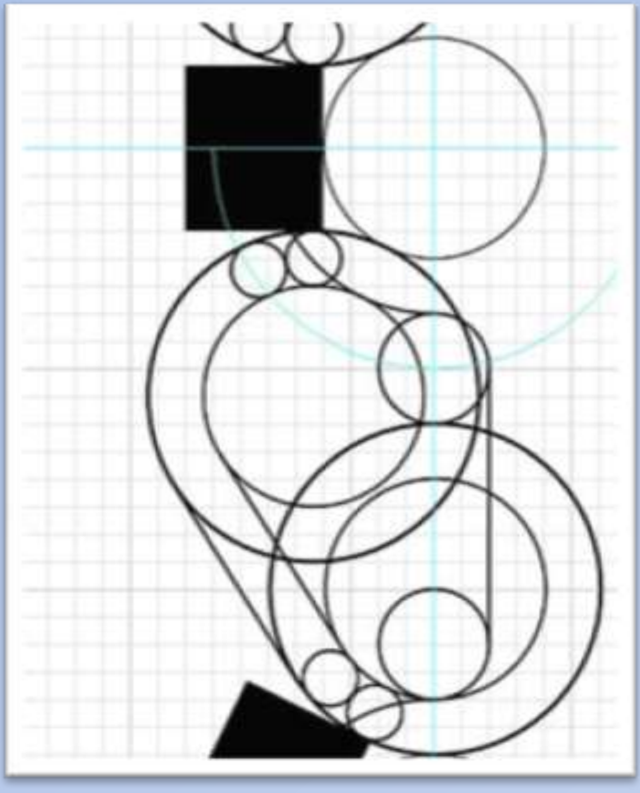
Outline

I. XML

II. XSLT

III. Examples

IV. Resources



XML (eXtensible Markup Language):

a set of rules for structuring data via markup

I. XML - What is it?

Tag:

<unitdate era="ce">2013**</unitdate>**

Attribute:

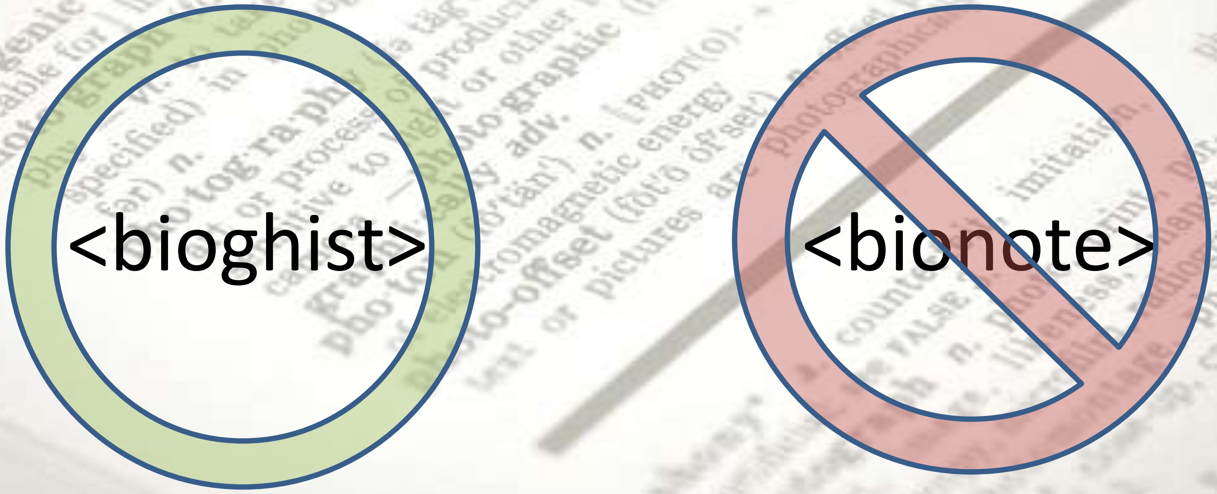
<unitdate **era="ce"**>2013</unitdate>

Element:

<unitdate era="ce">2013</unitdate>

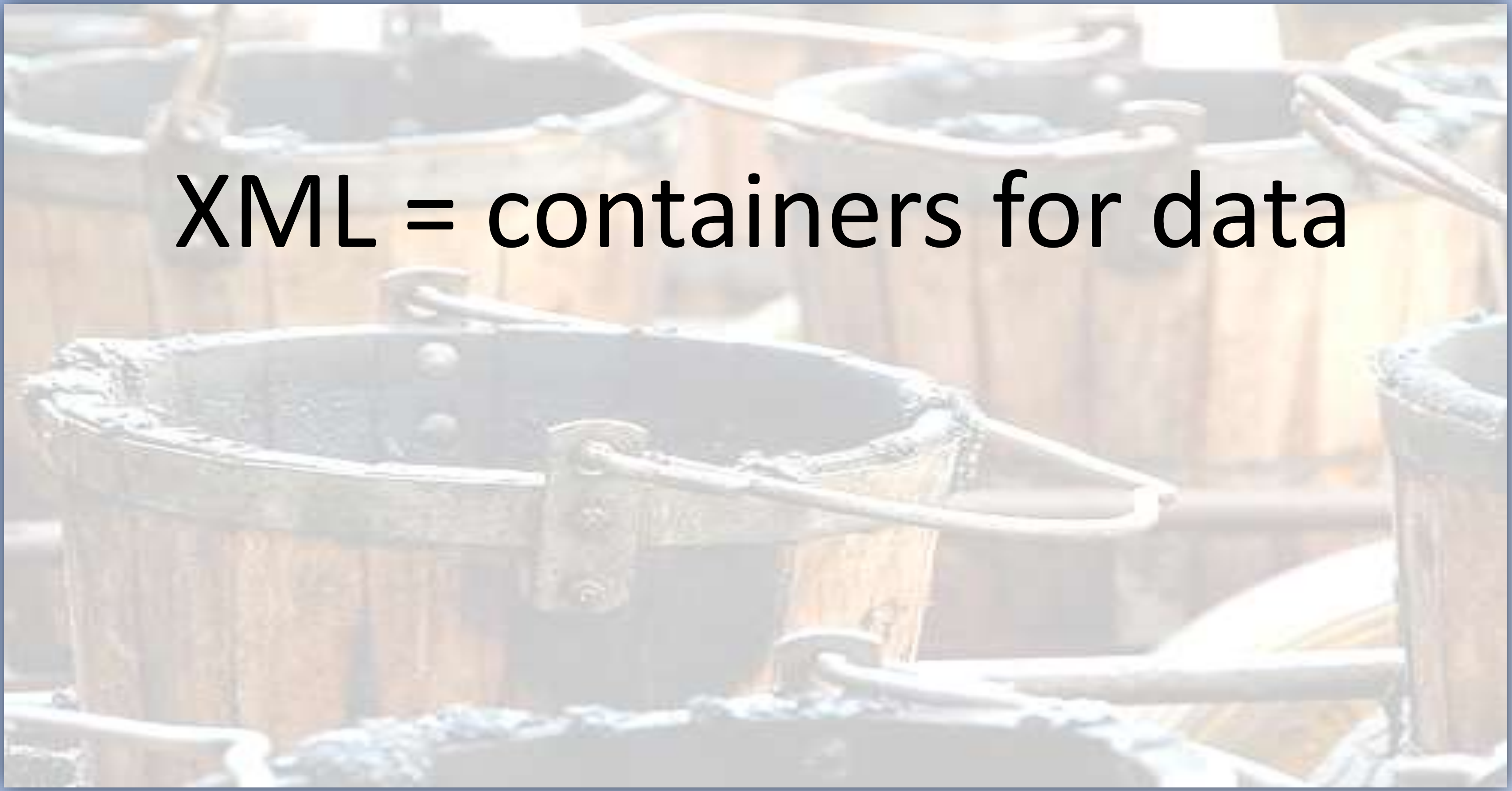
I. XML - What is it?

Elements and attributes defined by a Document Type Definition (DTD) or a Schema



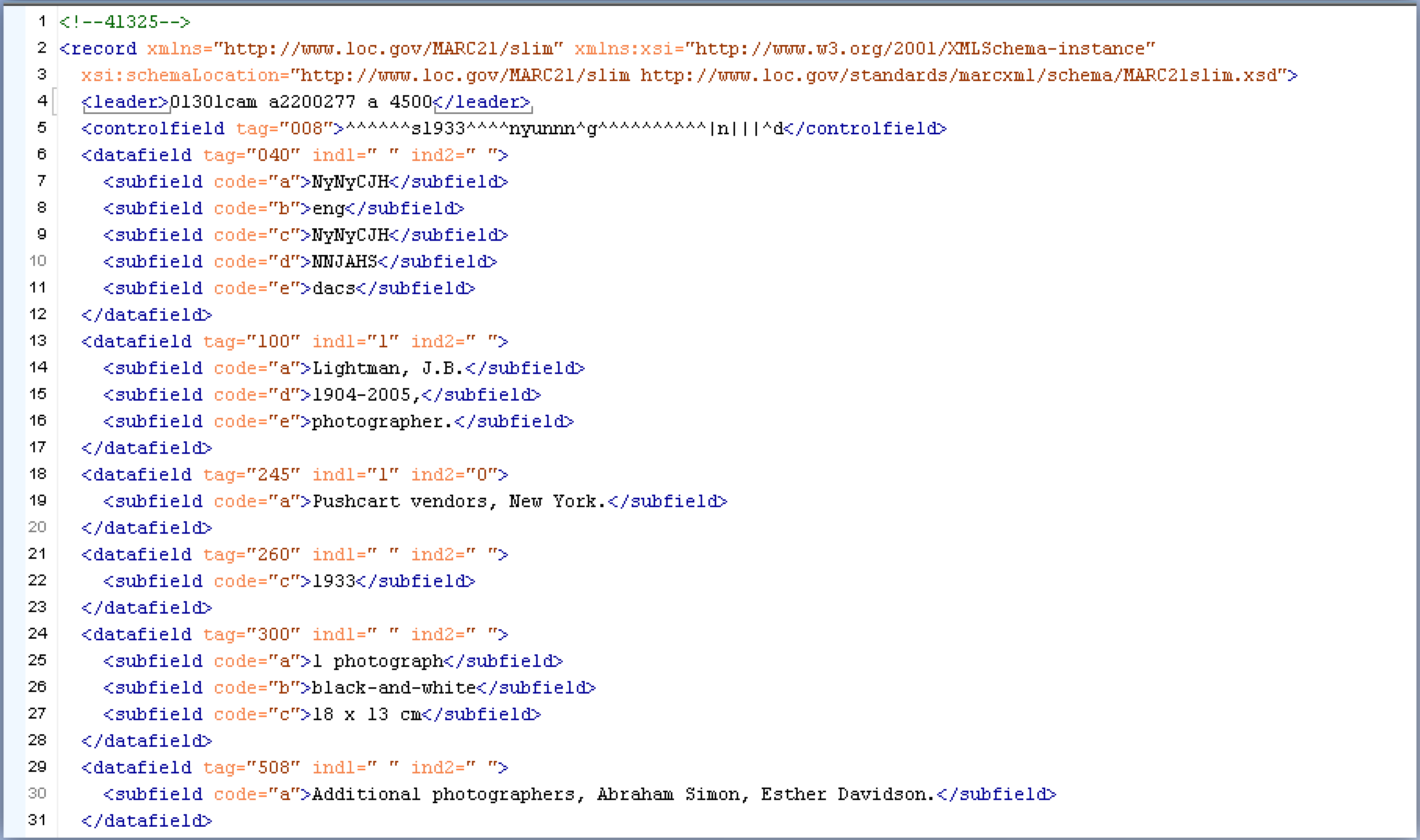
I. XML - What is it?

```
<ead>
  <eadheader>
    <titleproper>Guide to the
    Papers of Joseph Roth
    </titleproper>
  </eadheader>
</ead>
```



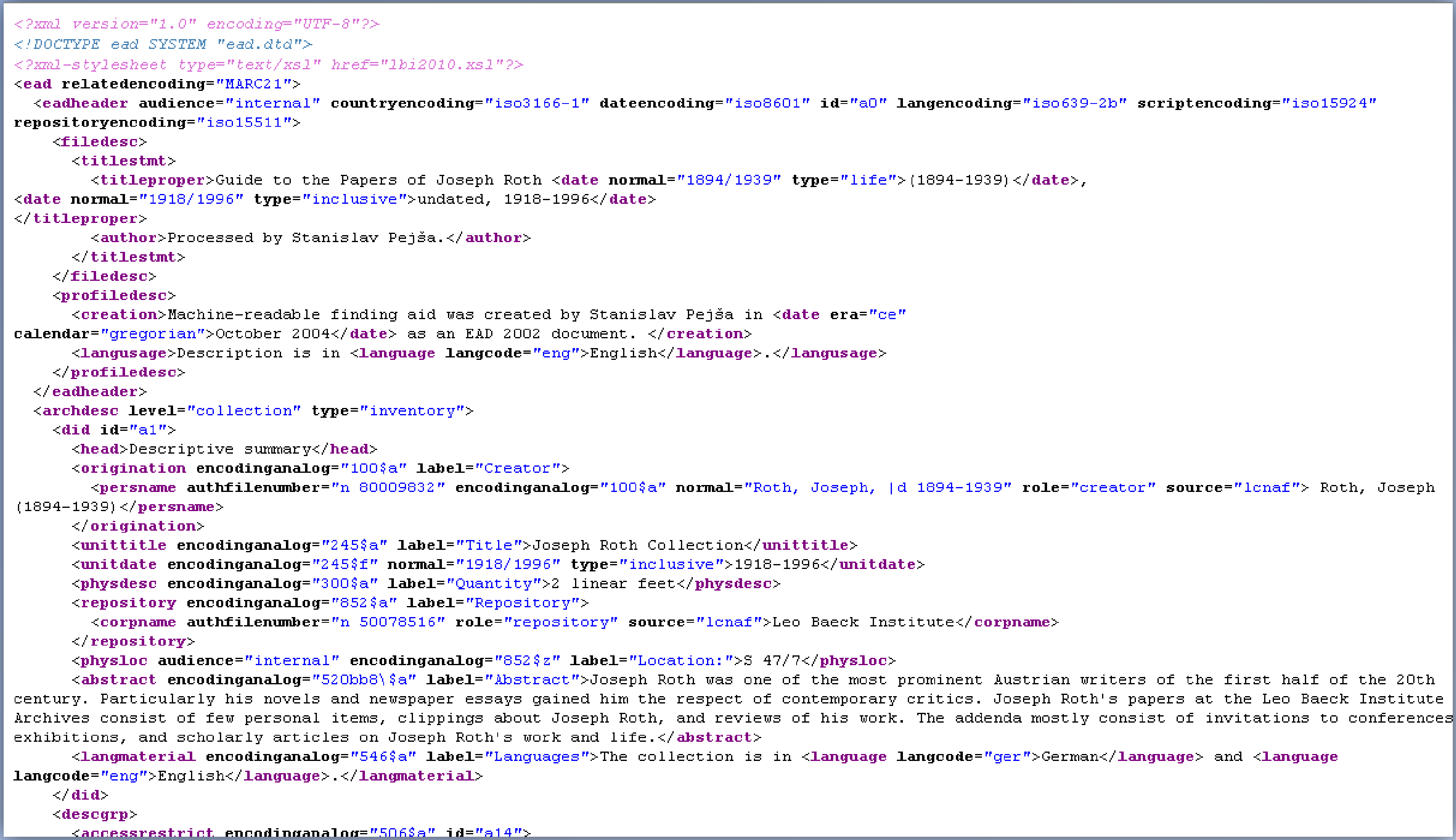
XML = containers for data

I. XML - MARCXML



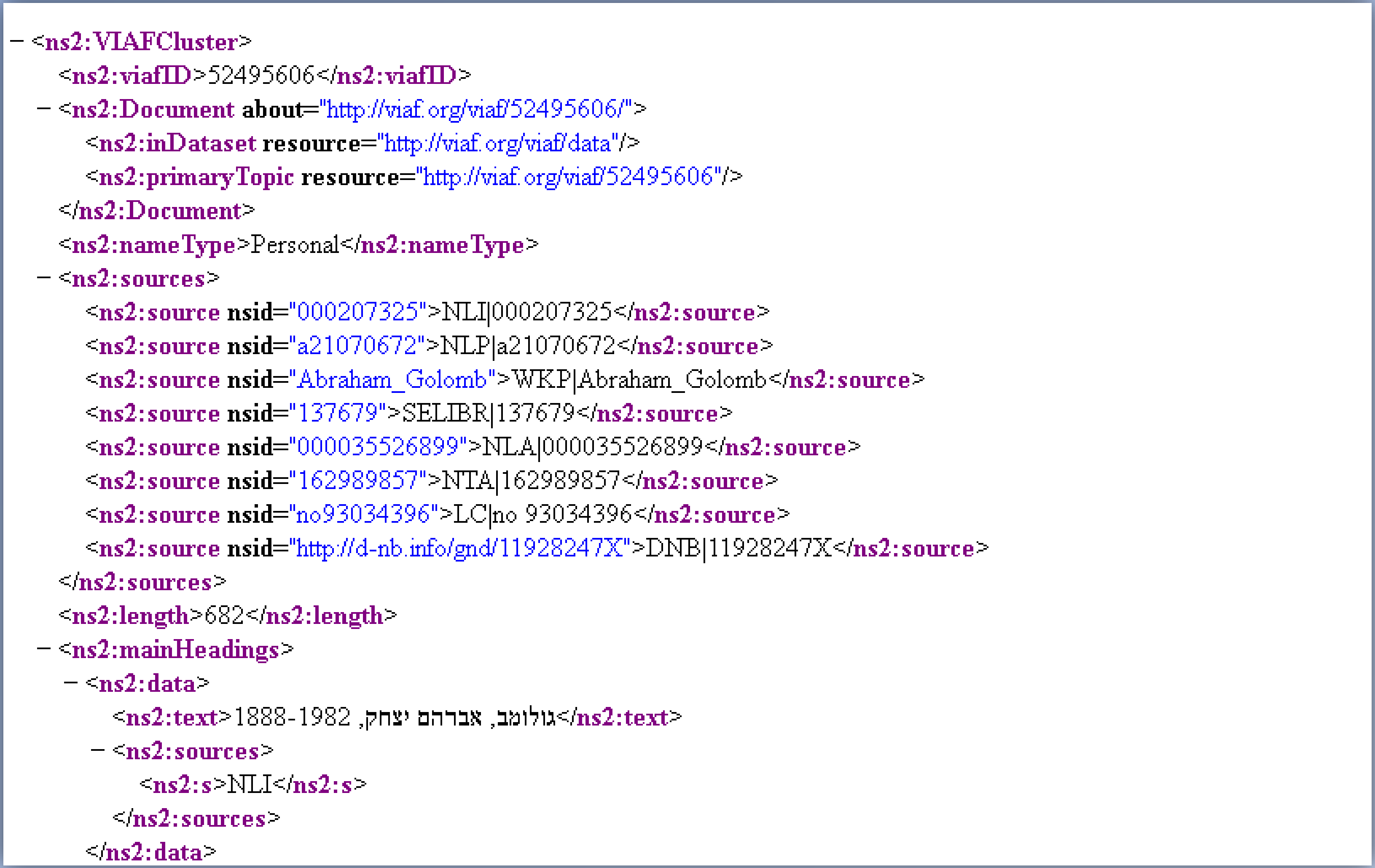
```xml
<!--41325-->
<record xmlns="http://www.loc.gov/MARC21/slim" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.loc.gov/MARC21/slim http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd">
  <leader>01301cam a2200277 a 4500</leader>
  <controlfield tag="008">^^^^^^s1933^^^^nyunnn^g^^^^^^^^^^^|n|||^d</controlfield>
  <datafield tag="040" ind1=" " ind2=" ">
    <subfield code="a">NyNyCJH</subfield>
    <subfield code="b">eng</subfield>
    <subfield code="c">NyNyCJH</subfield>
    <subfield code="d">NNJAHS</subfield>
    <subfield code="e">dacs</subfield>
  </datafield>
  <datafield tag="100" ind1="1" ind2=" ">
    <subfield code="a">Lightman, J.B.</subfield>
    <subfield code="d">1904-2005,</subfield>
    <subfield code="e">photographer.</subfield>
  </datafield>
  <datafield tag="245" ind1="1" ind2="0">
    <subfield code="a">Pushcart vendors, New York.</subfield>
  </datafield>
  <datafield tag="260" ind1=" " ind2=" ">
    <subfield code="c">1933</subfield>
  </datafield>
  <datafield tag="300" ind1=" " ind2=" ">
    <subfield code="a">1 photograph</subfield>
    <subfield code="b">black-and-white</subfield>
    <subfield code="c">18 x 13 cm</subfield>
  </datafield>
  <datafield tag="508" ind1=" " ind2=" ">
    <subfield code="a">Additional photographers, Abraham Simon, Esther Davidson.</subfield>
  </datafield>
```

I. XML - EAD



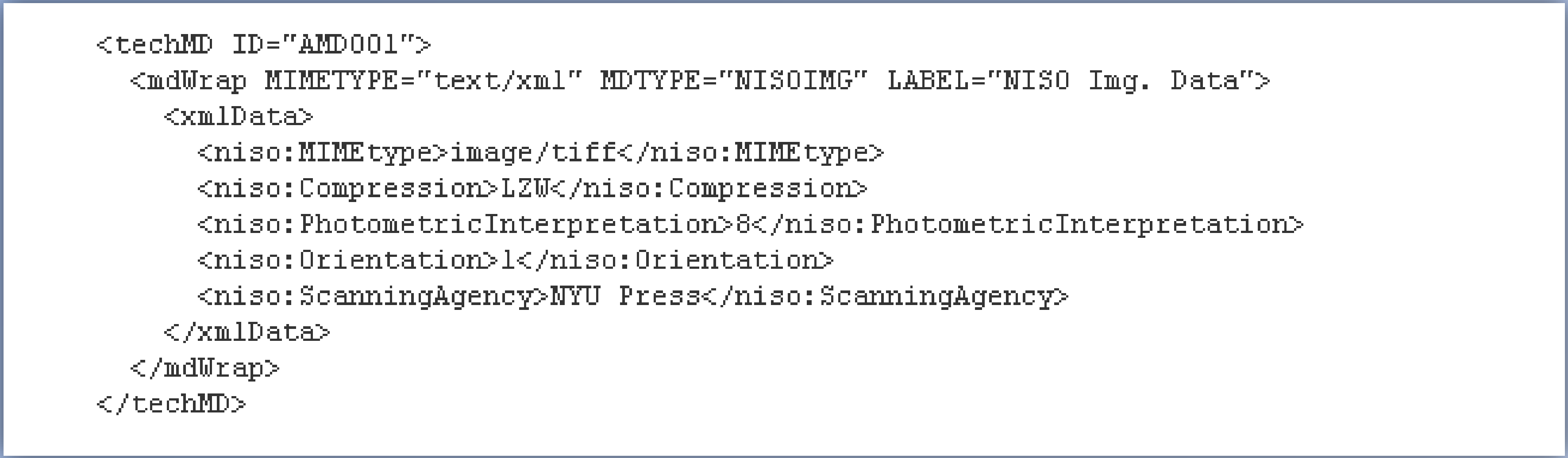
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ead SYSTEM "ead.dtd">
<?xml-stylesheet type="text/xsl" href="lbi2010.xsl"?>
<ead relatedencoding="MARC21">
  <eadheader audience="internal" countryencoding="iso3166-1" dateencoding="iso8601" id="a0" langencoding="iso639-2b" scriptencoding="iso15924"
repositoryencoding="iso15511">
    <filedesc>
      <titlestmt>
        <titleproper>Guide to the Papers of Joseph Roth <date normal="1894/1939" type="life">(1894-1939)</date>,
<date normal="1918/1996" type="inclusive">undated, 1918-1996</date>
</titleproper>
        <author>Processed by Stanislav Pejša.</author>
      </titlestmt>
    </filedesc>
    <profiledesc>
      <creation>Machine-readable finding aid was created by Stanislav Pejša in <date era="ce"
calendar="gregorian">October 2004</date> as an EAD 2002 document. </creation>
      <langusage>Description is in <language langcode="eng">English</language>.</langusage>
    </profiledesc>
  </eadheader>
  <archdesc level="collection" type="inventory">
    <did id="a1">
      <head>Descriptive summary</head>
      <origination encodinganalog="100$a" label="Creator">
        <persname authfilenumber="n 80009832" encodinganalog="100$a" normal="Roth, Joseph, |d 1894-1939" role="creator" source="lcnaf"> Roth, Joseph
(1894-1939)</persname>
      </origination>
      <unittitle encodinganalog="245$a" label="Title">Joseph Roth Collection</unittitle>
      <unitdate encodinganalog="245$f" normal="1918/1996" type="inclusive">1918-1996</unitdate>
      <physdesc encodinganalog="300$a" label="Quantity">2 linear feet</physdesc>
      <repository encodinganalog="852$a" label="Repository">
        <corpname authfilenumber="n 50078516" role="repository" source="lcnaf">Leo Baeck Institute</corpname>
      </repository>
      <physloc audience="internal" encodinganalog="852$z" label="Location:">S 47/7</physloc>
      <abstract encodinganalog="520bb8\$a" label="Abstract">Joseph Roth was one of the most prominent Austrian writers of the first half of the 20th
century. Particularly his novels and newspaper essays gained him the respect of contemporary critics. Joseph Roth's papers at the Leo Baeck Institute
Archives consist of few personal items, clippings about Joseph Roth, and reviews of his work. The addenda mostly consist of invitations to conferences
exhibitions, and scholarly articles on Joseph Roth's work and life.</abstract>
      <langmaterial encodinganalog="546$a" label="Languages">The collection is in <language langcode="ger">German</language> and <language
langcode="eng">English</language>.</langmaterial>
    </did>
    <descgrp>
      <accessrestrict encodinganalog="506$a" id="a14">
```

I. XML - VIAF cluster



```
- <ns2:VIAFCluster>
    <ns2:viafID>52495606</ns2:viafID>
  - <ns2:Document about="http://viaf.org/viaf/52495606/">
      <ns2:inDataset resource="http://viaf.org/viaf/data"/>
      <ns2:primaryTopic resource="http://viaf.org/viaf/52495606"/>
    </ns2:Document>
    <ns2:nameType>Personal</ns2:nameType>
  - <ns2:sources>
      <ns2:source nsid="000207325">NLI|000207325</ns2:source>
      <ns2:source nsid="a21070672">NLP|a21070672</ns2:source>
      <ns2:source nsid="Abraham_Golomb">WKP|Abraham_Golomb</ns2:source>
      <ns2:source nsid="137679">SELIBR|137679</ns2:source>
      <ns2:source nsid="000035526899">NLA|000035526899</ns2:source>
      <ns2:source nsid="162989857">NTA|162989857</ns2:source>
      <ns2:source nsid="no93034396">LC|no 93034396</ns2:source>
      <ns2:source nsid="http://d-nb.info/gnd/11928247X">DNB|11928247X</ns2:source>
    </ns2:sources>
    <ns2:length>682</ns2:length>
  - <ns2:mainHeadings>
    - <ns2:data>
        <ns2:text>גולומב, אברהם יצחק, 1888-1982</ns2:text>
      - <ns2:sources>
          <ns2:s>NLI</ns2:s>
        </ns2:sources>
      </ns2:data>
```

I. XML - Namespaces



```xml
<techMD ID="AMD001">
  <mdWrap MIMETYPE="text/xml" MDTYPE="NISOIMG" LABEL="NISO Img. Data">
    <xmlData>
      <niso:MIMEtype>image/tiff</niso:MIMEtype>
      <niso:Compression>LZW</niso:Compression>
      <niso:PhotometricInterpretation>8</niso:PhotometricInterpretation>
      <niso:Orientation>1</niso:Orientation>
      <niso:ScanningAgency>NYU Press</niso:ScanningAgency>
    </xmlData>
  </mdWrap>
</techMD>
```

I. XML - Many more….

<glossary of metadata standards>

[illegible]

</>

What is XPath

- XPath is just a way to locate particular parts of an XML document
- It can identify tags, attributes, whole elements, or the content of elements (any “node” in the document)
- A simple (abbreviated) Xpath expression works just like a file path name in Windows.
  - If you are looking for an element that is nested directly within other elements just list the elements with slashes
    - /element1/element2/element3
    - /ead/archdesc/did/unittitle

What is XPath

- You can easily point to an attribute instead
  - /element1/element2/element3/@attribute
  - /ead/archdesc/did/unittitle/@audience
- You can also point to an element, attribute or content node using an additional test—called a predicate
  - /ead/archdesc/did/unittitle [@audience="external"]
    - This picks the unittitle elements that have an audience attribute that equals external

What is XPath

- Sometimes you want to locate nodes (elements, attributes, or text) not using a direct path but using the "tree" structure of XML
  - Because:
    - You don't know it
    - It changes from document to document, but you want one XSLT sheet to work for all of them
    - You don't care were the element is, but want to locate all the times it appears
    - You want to locate a number of nodes based on another node—for example you want all the elements that have a child element called <unitdate>, which could include <did> and <unittitle> elements
  - Then you often need to write out the Axis—This describes the relationship between the spot you are selecting and the node you are writting
    - Axis:NodeTest [Predicate]
    - /ead/descendant::node()/unitdate/parent::node()

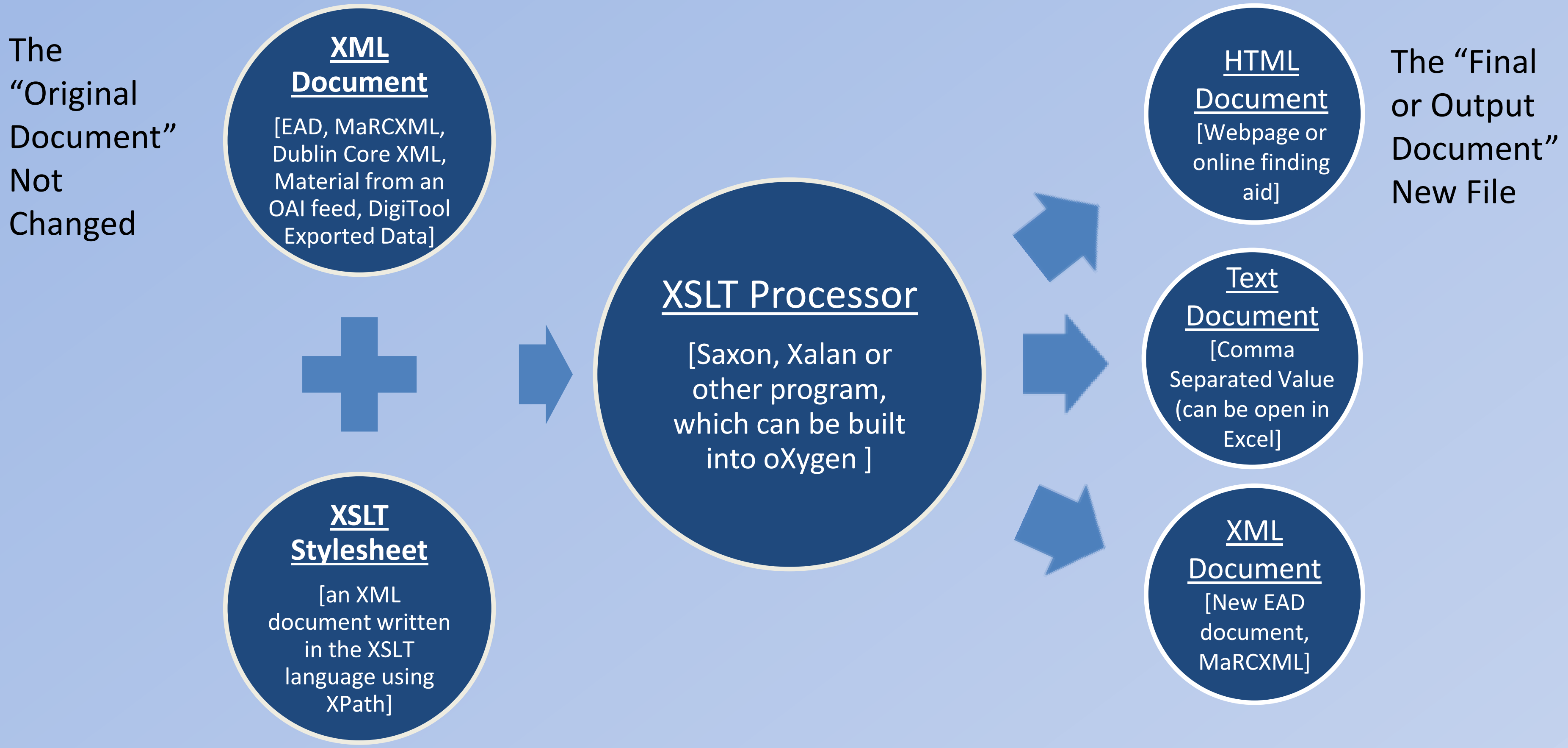
What is XSLT?

- eXtensible Stylesheet Language Transformations
- An XSLT document is just a series of rules for creating documents, written in a way computers can read
- <u>It essentially says to a computer “find location A in original XML documents, and depending on what you find there, put something in the final documents”</u>
- The original document file is NEVER affected by XSLT. The changes always appear in a new file.

What is XSLT?

- An XSLT Stylesheet essentially says to a computer "find location A in original XML documents, and depending on what you find there, put something in the final documents"
  - You use XPath to indicate the "location"
  - The "location" can be a single spot [The third time an element appears in the original] ***or*** multiple, more generic spots[every time title appears as an tag name]
  - The "location" is not identical for all documents of one type. For example since every repository has its own way of doing EAD, one XSLT sheet can't work on all EAD documents. You can't write XSLT without knowing details about the original documents.

Processing documents


The "Original Document" Not Changed
XML Document
[EAD, MaRCXML, Dublin Core XML, Material from an OAI feed, DigiTool Exported Data]
XSLT Stylesheet
[an XML document written in the XSLT language using XPath]
XSLT Processor
[Saxon, Xalan or other program, which can be built into oXygen]
HTML Document
[Webpage or online finding aid]
Text Document
[Comma Separated Value (can be open in Excel]
XML Document
[New EAD document, MaRCXML]
The "Final or Output Document" New File

The Structure of XSLT Language

You can think about how XSLT works in two basic ways:

These are also programing strategies, but remember that each coder has their own way of writing and most people combine the two

The "Push" method

- This relies on essentially telling the computer to copy the original document, while making changes
- This gives more power in the final output to the person writing EAD or other input document

The "Pull" method

- This relies on selecting specific elements and tags from the original document and placing them in specific spots in the output (with changes as needed)
- This puts more control in the hands of the XSLT writer and is best for when the original documents are different from each other but the output needs to be the same

The Structure of XSLT Language

You can think about how XSLT works in two basic ways:

These are also programing strategies, but remember that each coder has their own way of writing and most people combine the two

The "Push" method	*The "Pull" method*

You can get the same results either way. Think about a word processing document you need to edit, without changing the original.

- You can copy the whole thing and then make large changes. (Push)
- Or you can copy and paste individual paragraphs into a new document each in its final spot, then make smaller changes. (Pull)

Which makes more sense depends on whether you need to preserve the structure of the original or if you need to change the order of the paragraphs drastically.

Basic XSLT Code

[just a little bit, I promise]

- Since XSLT is a type of XML, it uses tags, namespaces, and attributes just like EAD does.
  - A “namespace” (in this case, as a simplification using the most common situation) just lets the computer know if the tag you are typing is an XSLT element that gives it instructions, or if it an tag for the final document (an EAD or HTML tag you want to “print”)
  - Namespaces go right before the element, usually as an abbreviation
    - <xsl:tag></xsl:tag> Or <dc:title></dc:title>
- It also uses XPath expressions(see above) and functions
  - Functions are just instructions that make changes in the output document, usually to the text content of an element [i.e.—Capitalize all the letters of all titles]
  - They usually are put inside attributes
    - <tag attribute=“function-name(argument, argument)”></tag>

Basic XSLT Code—The Elements

[just a little bit, I promise]

- <xsl:stylesheet> the tag that surrounds all the others in a document (like the <ead> or <html> tags)
- <xsl:template> Defines rules for the output—applies rules on the element selected and all its children
- <xsl:value-of> Puts the content of an element in the output
- <xsl:element> Creates an XML element in the output document
- <xsl:apply-templates> Tells the computer to find the most appropriate template in the stylesheet and apply it in the middle of another command
- The Built-in or default template runs on everything in the original document and puts all the element content in the final document

XSLT Starter Example

Original Document

```
<book>
   <title>Hello Word Book</title>
   <date>1997</date>
</book>
```

XSLT Stylesheet

```
<xsl:stylesheet>
   <xsl:template match=“book”>
     The title of my book is <value-of select=“title”>.
   </xsl:template>
</xsl:stylesheet>
```

Output Document

```
The title of my book is Hello Word Book.
```

How do we use XSLT? [The basics]

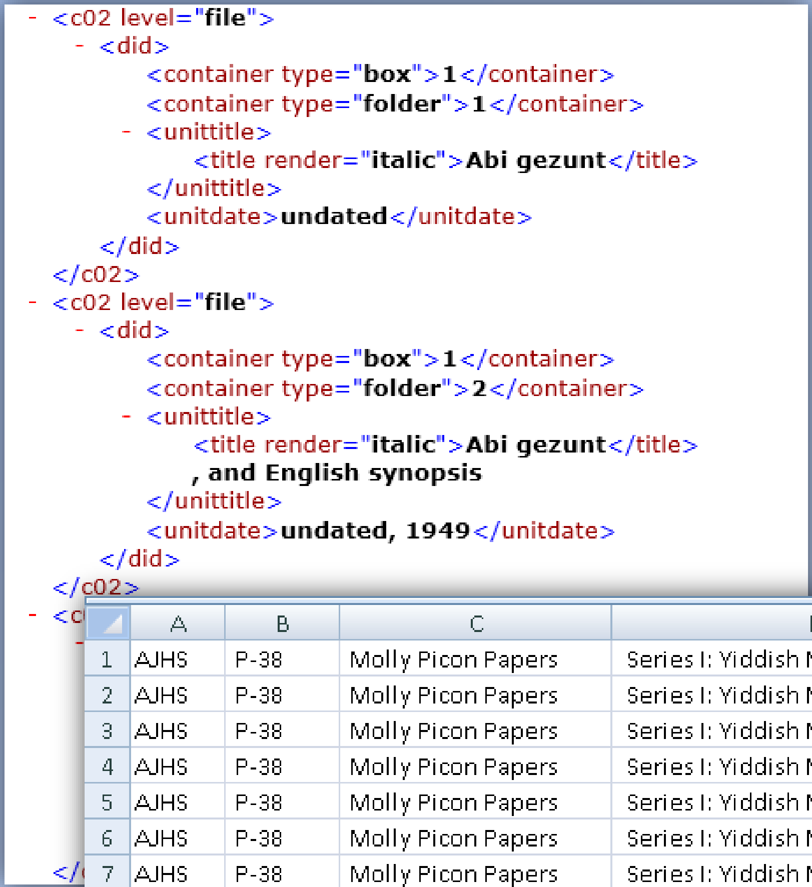
- The most common and established way we use XSLT is to produce HTML from an EAD document so that it can be shown by a web browser to on-line users
- At the Center, DigiTool stores all our EAD files and one XSLT sheet for each partner
  - Every time a *user* requests a finding aid, DigiTool automatically runs the EAD through the stylesheet and displays the final output document, an HTML page.
- All archivists who encode EAD use the correct copy of the Center's 5 partner stylesheets (hosted at URLs) to test their EAD before posting it, usually through oXygen
  - oXygen "Transformation Scenarios" control how this happens and how the archivist gets to see the HTML final document. Then the HTML is discarded or saved on some other system

BUT….

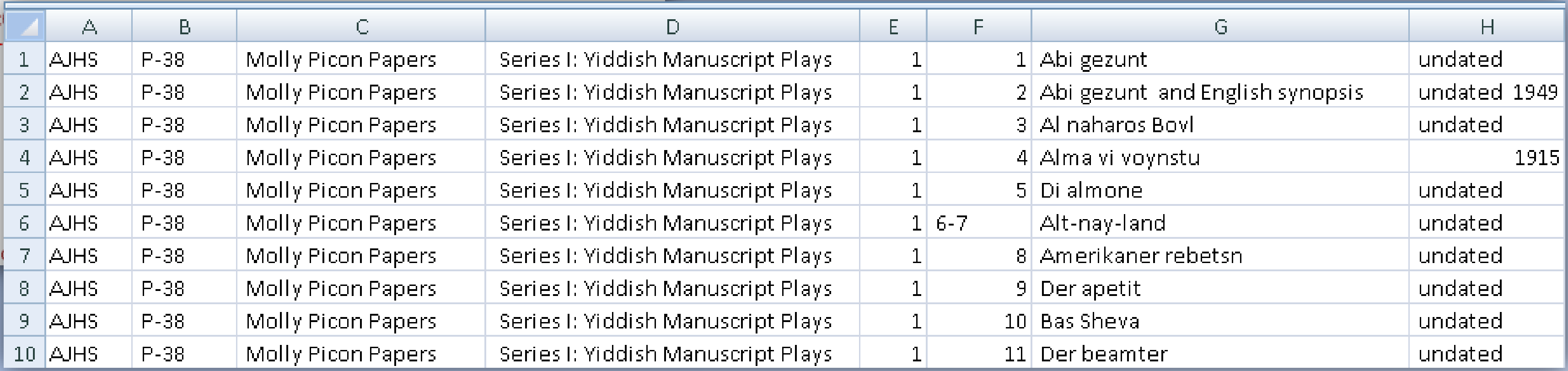
XSLT can do a whole lot more for any XML document, and today, libraries and the web run on XML

III. Examples

<dsc> to .csv

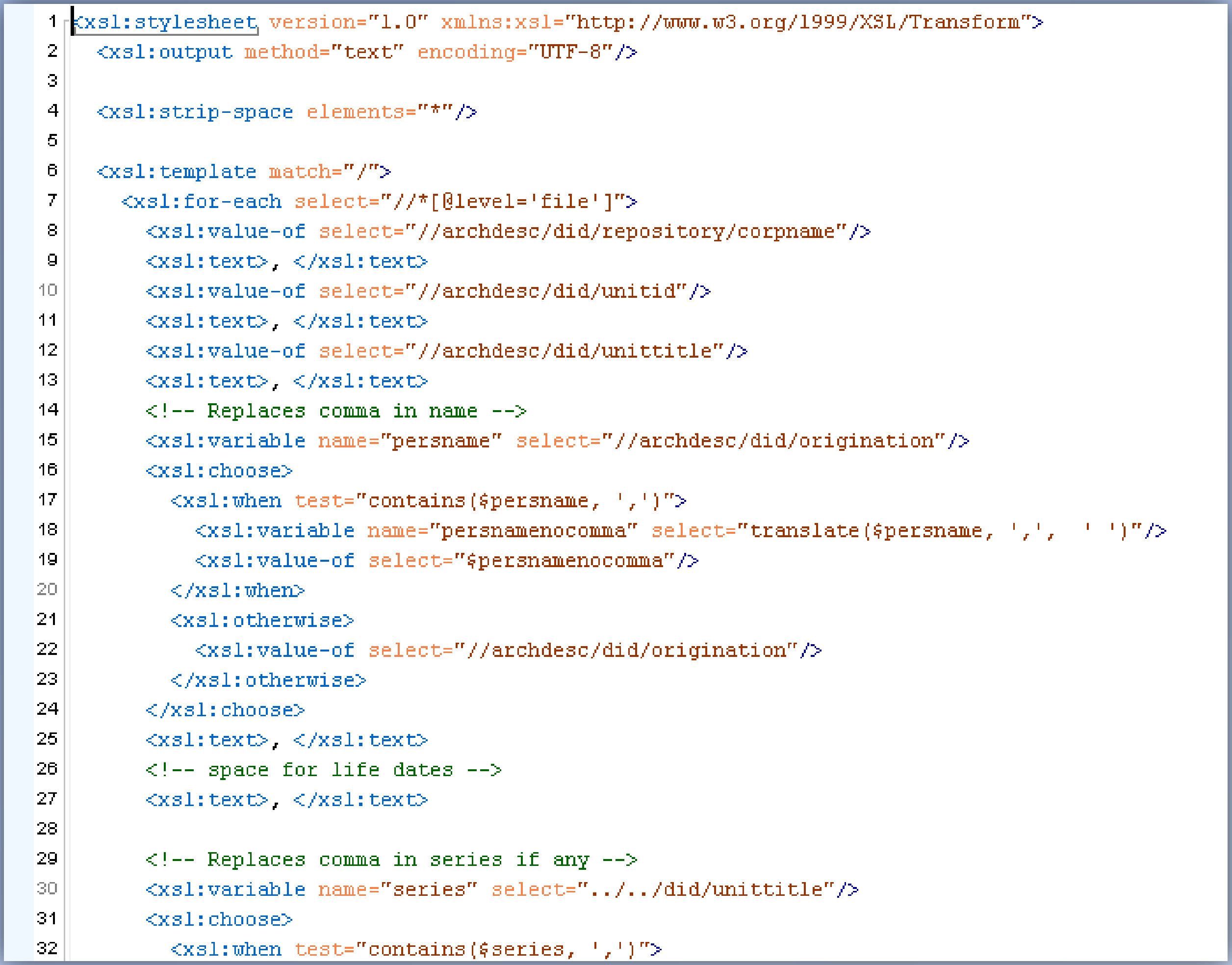


```
- <c02 level="file">
  - <did>
      <container type="box">1</container>
      <container type="folder">1</container>
    - <unittitle>
        <title render="italic">Abi gezunt</title>
      </unittitle>
      <unitdate>undated</unitdate>
    </did>
  </c02>
- <c02 level="file">
  - <did>
      <container type="box">1</container>
      <container type="folder">2</container>
    - <unittitle>
        <title render="italic">Abi gezunt</title>
        , and English synopsis
      </unittitle>
      <unitdate>undated, 1949</unitdate>
    </did>
  </c02>
- <c
  </
```



	A	B	C	D	E	F	G	H
1	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	1	Abi gezunt	undated
2	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	2	Abi gezunt and English synopsis	undated 1949
3	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	3	Al naharos Bovl	undated
4	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	4	Alma vi voynstu	1915
5	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	5	Di almone	undated
6	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	6-7	Alt-nay-land	undated
7	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	8	Amerikaner rebetsn	undated
8	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	9	Der apetit	undated
9	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	10	Bas Sheva	undated
10	AJHS	P-38	Molly Picon Papers	Series I: Yiddish Manuscript Plays	1	11	Der beamter	undated

<dsc> to .csv



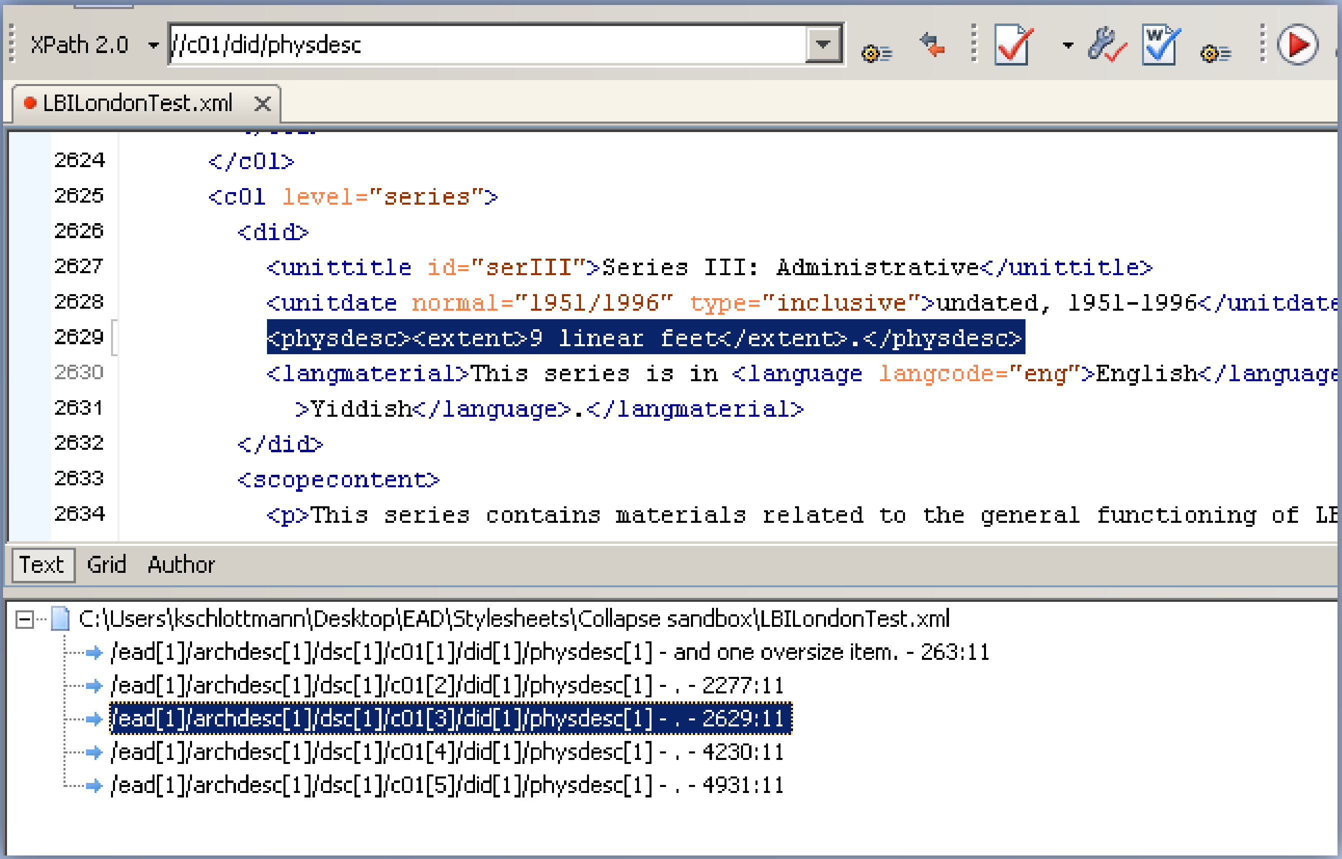
```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" encoding="UTF-8"/>

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

  <xsl:template match="/">
    <xsl:for-each select="//*[@level='file']">
      <xsl:value-of select="//archdesc/did/repository/corpname"/>
      <xsl:text>, </xsl:text>
      <xsl:value-of select="//archdesc/did/unitid"/>
      <xsl:text>, </xsl:text>
      <xsl:value-of select="//archdesc/did/unittitle"/>
      <xsl:text>, </xsl:text>
      <!-- Replaces comma in name -->
      <xsl:variable name="persname" select="//archdesc/did/origination"/>
      <xsl:choose>
        <xsl:when test="contains($persname, ',')">
          <xsl:variable name="persnamenocomma" select="translate($persname, ',',  ' ')"/>
          <xsl:value-of select="$persnamenocomma"/>
        </xsl:when>
        <xsl:otherwise>
          <xsl:value-of select="//archdesc/did/origination"/>
        </xsl:otherwise>
      </xsl:choose>
      <xsl:text>, </xsl:text>
      <!-- space for life dates -->
      <xsl:text>, </xsl:text>

      <!-- Replaces comma in series if any -->
      <xsl:variable name="series" select="../../did/unittitle"/>
      <xsl:choose>
        <xsl:when test="contains($series, ',')">
```

xPath in EAD

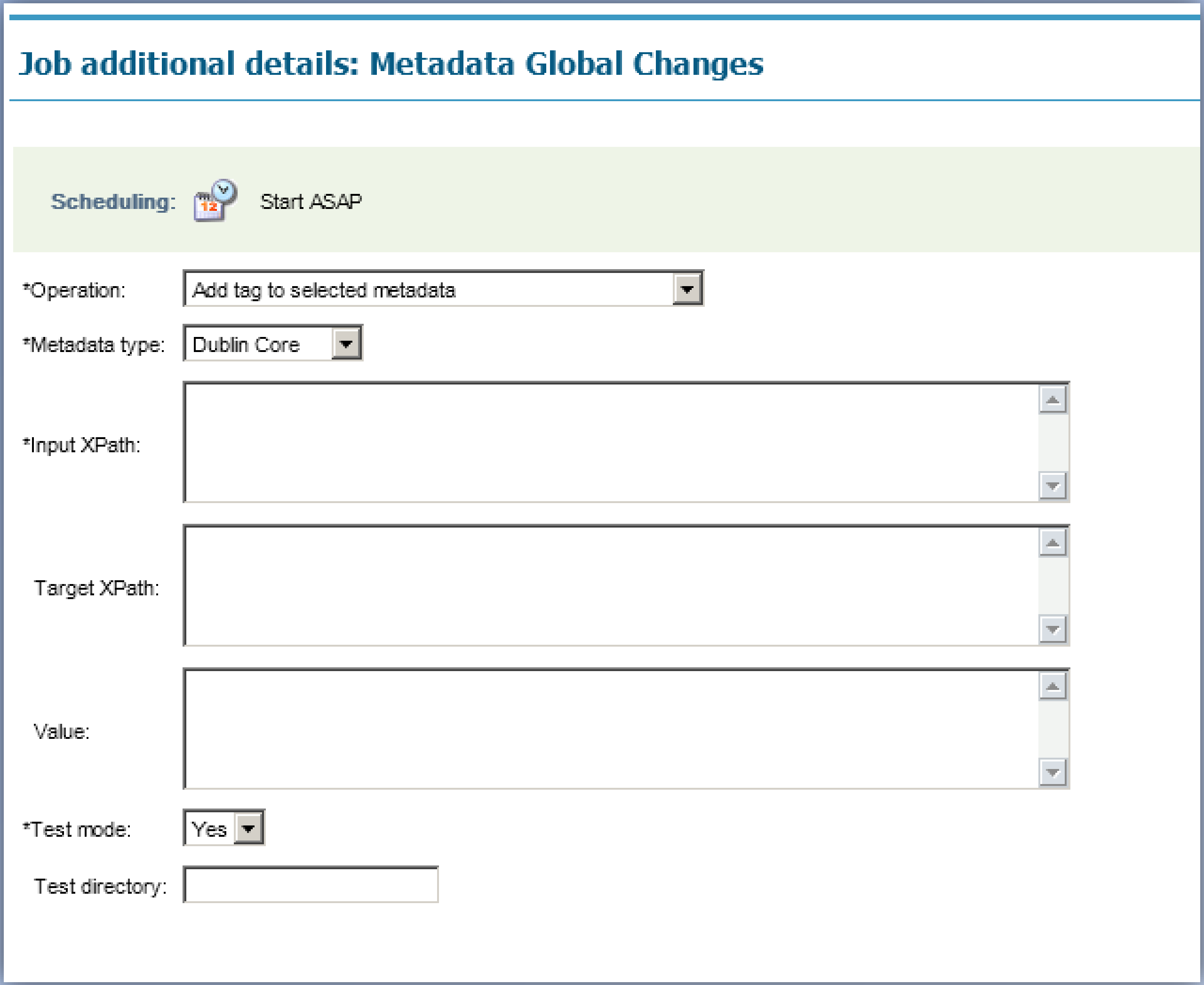


```
//c01/did/physdesc
```

```
      </c01>
      <c01 level="series">
        <did>
          <unittitle id="serIII">Series III: Administrative</unittitle>
          <unitdate normal="1951/1996" type="inclusive">undated, 1951-1996</unitdate
          <physdesc><extent>9 linear feet</extent>.</physdesc>
          <langmaterial>This series is in <language langcode="eng">English</language
            >Yiddish</language>.</langmaterial>
        </did>
        <scopecontent>
          <p>This series contains materials related to the general functioning of LE
```

```
C:\Users\kschlottmann\Desktop\EAD\Stylesheets\Collapse sandbox\LBILondonTest.xml
  /ead[1]/archdesc[1]/dsc[1]/c01[1]/did[1]/physdesc[1] - and one oversize item. - 263:11
  /ead[1]/archdesc[1]/dsc[1]/c01[2]/did[1]/physdesc[1] - . - 2277:11
  /ead[1]/archdesc[1]/dsc[1]/c01[3]/did[1]/physdesc[1] - . - 2629:11
  /ead[1]/archdesc[1]/dsc[1]/c01[4]/did[1]/physdesc[1] - . - 4230:11
  /ead[1]/archdesc[1]/dsc[1]/c01[5]/did[1]/physdesc[1] - . - 4931:11
```

Digitool Metadata

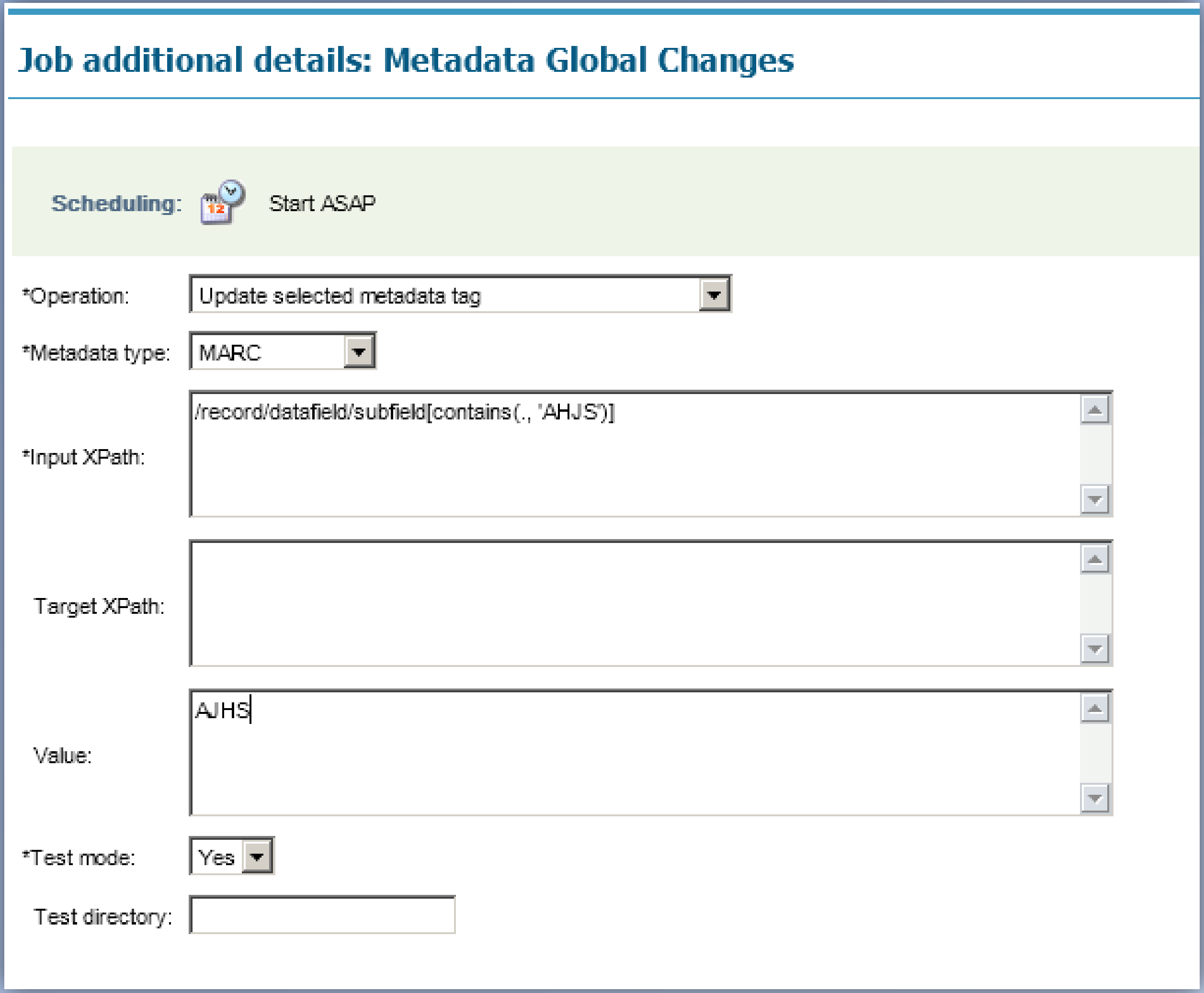


Job additional details: Metadata Global Changes

Scheduling: Start ASAP

*Operation:	Add tag to selected metadata
*Metadata type:	Dublin Core
*Input XPath:	
Target XPath:	
Value:	
*Test mode:	Yes
Test directory:	

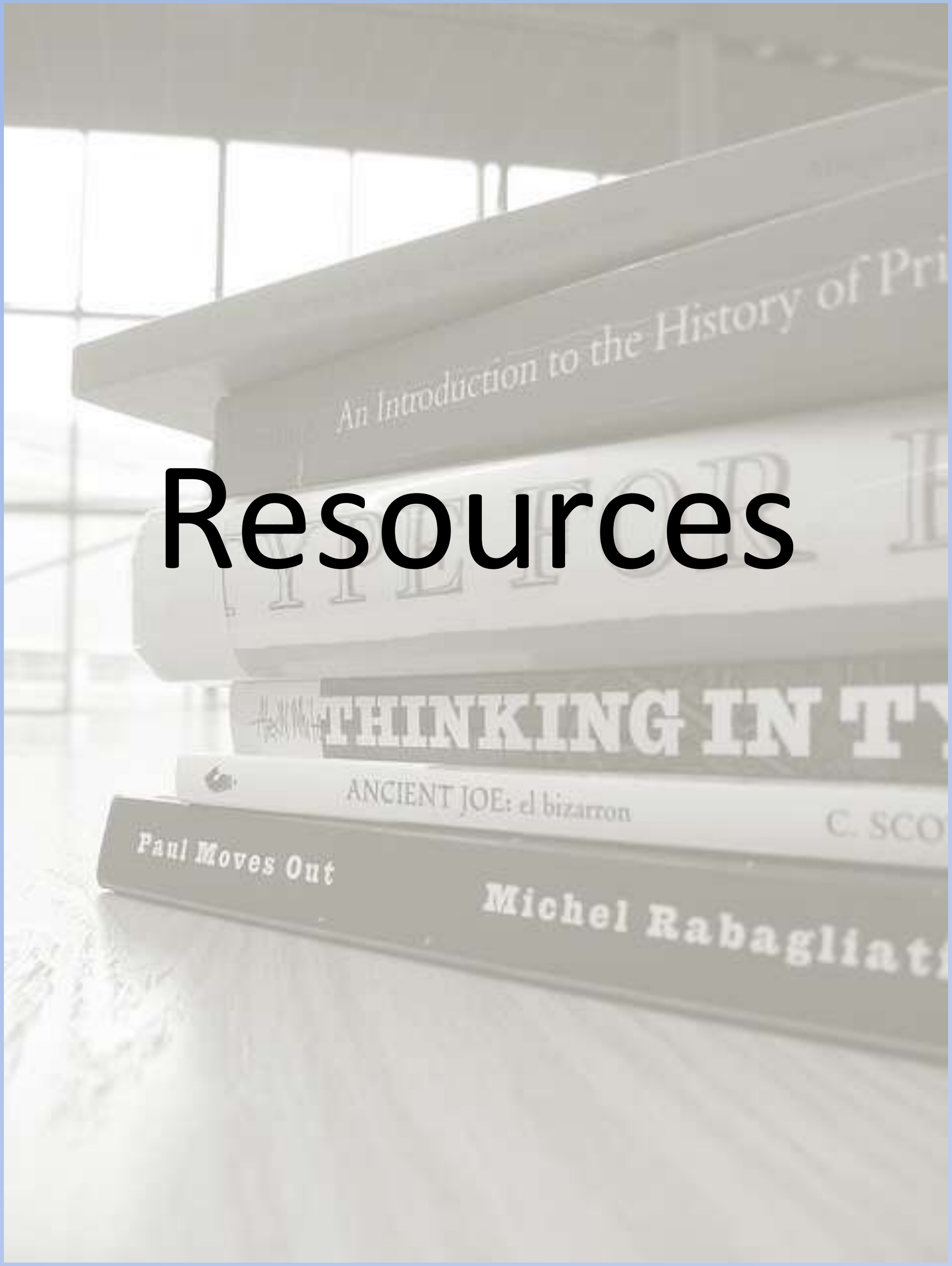
Digitool Metadata



Job additional details: Metadata Global Changes

Scheduling: Start ASAP

*Operation:	Update selected metadata tag
*Metadata type:	MARC
*Input XPath:	/record/datafield/subfield[contains(., 'AHJS')]
Target XPath:	
Value:	AJHS
*Test mode:	Yes
Test directory:	


An Introduction to the History of Pri
THINKING IN TY
ANCIENT JOE: el bizarron
C. SCO
Paul Moves Out
Michel Rabagliat

Resources

Tinker!

- Learn more about XML and XSLT
- Download the free trial of oXygen XML editor, the EAD schema, a finding aid, and a stylesheet.
- Try some basic actions: add a folder, change a controlled vocabulary term, remove a series.