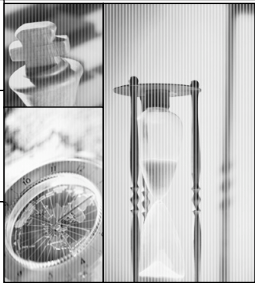


Practical XML & XSLT

Roy Tennant, The California Digital Library



Practical XML & XSLT

Roy Tennant
California Digital Library

Setting Expectations

- We can only do so much in three hours (and I will be dumping a lot on you)
- XSLT cannot be done by beginners without reference to examples, books, etc.
- My goals:
 - Introduce you to key concepts
 - Demonstrate some basic operations
 - “Break the ice” for your own continued learning

Outline

- XML Basics
- Displaying XML with CSS
- Transforming XML with XSLT
- Serving XML to Web Users
- Resources
- Tips & Advice

Documents

- XML is expressed as “documents”, whether an entire book or a database record
- Must have:
 - At least one element
 - Only one “root” element
- Should have:
 - A document type declaration; e.g., `<?xml version="1.0"?>`
 - Namespace declarations
- Can have:
 - One or more properly nested elements
 - Comments
 - Processing instructions

Elements

- Must have a name; e.g., `<title>`
- Names must follow rules: no spaces or special characters, must start with a letter, are case sensitive
- Must have a beginning and end; `<title></title>` or `<title/>`
- May wrap text data; e.g., `<title>Hamlet</title>`
- May have an attribute that must be quoted; e.g., `<title level="main">Hamlet</title>`
- May contain other “child” elements; e.g., `<title level="main">Hamlet <subtitle>Prince of Denmark</subtitle></title>`

Element Relationships

- Every XML document must have only one “root” element
- All other elements must be contained within the root
- An element contained within another tag is called a “child” of the container element
- An element that contains another tag is called the “parent” of the contained element
- Two elements that share the same parent are called “siblings”

Practical XML & XSLT

Roy Tennant, The California Digital Library

The Tree

```

<?xml version="1.0">
<book>
  <author>
    <lastname>Tennant</lastname>
    <firstname>Roy</firstname>
  </author>
  <title>The Great American Novel</title>
  <chapter number="1">
    <chaptitle>It Was Dark and Stormy</chaptitle>
    <p>It was a dark and stormy night.</p>
    <p>An owl hooted.</p>
  </chapter>
</book>
  
```

Diagram illustrating the XML tree structure:

- Root element: `<book>`
- Parent of `<lastname>`: `<author>`
- Child of `<author>`: `<lastname>` and `<firstname>`
- Siblings: `<chaptitle>` and `<p>` (under `<chapter>`)

Comments & Processing Instructions

- You can embed comments in your XML just like in HTML:


```
<!-- Whatever is here (whether text or markup) will be ignored on processing -->
```
- A processing instruction tells the XML parser information it needs to know to properly process an XML document:


```
<?xml-stYLESHEET type="text/css" href="style2.css"?
```

Well-Formed XML

- Follows *general* tagging rules:
 - All tags begin and end
 - But can be minimized if empty: `
` instead of `
</br>`
 - All tags are case sensitive
 - All tags must be properly nested:
 - `<author> <firstname>Mark</firstname> <lastname>Twain</lastname> </author>`
 - All attribute values are quoted:
 - `<subject scheme="LCSH">Music</subject>`
- Has identification & declaration tags
- Software can make sure a document follows these rules

Valid XML

- Uses only specific tags and rules as codified by one of:
 - A document type definition (DTD)
 - A schema definition
- Only the tags listed by the schema or DTD can be used
- Software can take a DTD or schema and verify that a document adheres to the rules
- Editing software can prevent an author from using anything except allowed tags

Namespaces

- A method to keep metadata elements from different schemas from colliding
- Example: the tag `<name>` may have a very different meaning in different standards
- A namespace declaration specifies from which specification a set of tags is drawn


```
<mets xmlns="http://www.loc.gov/METS/" xsi:schemaLocation="http://www.loc.gov/standards/mets/mets.xsd">
```

Character Encoding

- XML is Unicode, either UTF-8 or UTF-16
- However, you can *output* XML into other character encodings (e.g., ISO-Latin1)
- But, in XML you must use Unicode character encodings — see “Where is My Character?” at <http://www.unicode.org/unicode/standard/where/>
- Or, use `<![CDATA[]]>` to wrap any special characters you don't want to be treated as markup (e.g., ` `)

Practical XML & XSLT

Roy Tennant, The California Digital Library

Special Character Entities

- There are 5 characters that are reserved for special purposes; therefore, to use these characters when not part of XML tags, you must use an *entity reference*:
 - & (ampersand) becomes: &
 - < (less than) becomes: <
 - > (greater than) becomes: >
 - ' (apostrophe) becomes: '
 - " (quote) becomes: "

Displaying XML: CSS

- A modern web browser (e.g., MSIE, Mozilla) and a cascading style sheet (CSS) may be used to view XML as if it were HTML
- A style must be defined for every XML tag, or else the browser displays it in its default mode
- All display characteristics of each element must be *explicitly* defined
- Elements are displayed in the order they are encountered in the XML
- *No reordering of elements or other processing is possible*

Displaying XML with CSS

- Must put a processing instruction at the top of your XML file (but below the XML declaration):

```
<?xml-stylesheet type="text/css" href="style.css"?>
```
- Must specify *all* display characteristics of *all* tags, or it will be displayed in default mode (whatever the browser wants)
- Demonstration

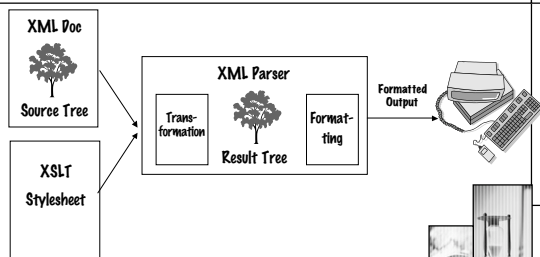
Transforming XML: XSLT

- XML Stylesheet Language — Transformations (XSLT)
- A markup language and programming syntax for processing XML
- Is most often used to:
 - Transform XML to HTML for delivery to standard web clients
 - Transform XML from one set of XML tags to another
 - Transform XML into another syntax/system

XSLT Primer

- XSLT is based on the process of matching **templates** to **nodes** of the XML tree
- Working down from the top, XSLT tries to match segments of code to:
 - The root element
 - Any child node
 - And on down through the document
- You can specify different processing for each element if you wish

XSLT Processing Model



From *Professional XSL*, Wrox Publishers

Practical XML & XSLT

Roy Tennant, The California Digital Library

Nodes and XPath

- An XML document is a collection of **nodes** that can be identified, selected, and acted upon using an **Xpath statement**
- Examples of nodes: root, element, attribute, text

XPath Essentials

- `//article` = Select all `<article>` elements of the root node
- `//article[@name='test']` = Select all `<article>` elements of the root node that have a name attribute with the value 'test'
- `//article/title` = Select all `<title>` elements that have an `<article>` element as a parent
- A period (`.`) denotes the current context node (e.g., `./title` selects any title tag that is a child of the current node)
- Two periods (`..`) denote the parent node of the current context

Templates

- An XSLT stylesheet is a collection of templates that act against specified nodes in the XML source tree
- For example, this template will be executed when a `<para>` element is encountered:

```
<xsl:template match="para">  
  <p><xsl:value-of select="."/></p>  
</xsl:template>
```

Calling Templates

- A template can call other templates
- By default (tree processing):
`<xsl:apply-templates/>` [processes all children of the current node]
- Explicitly:
`<xsl:apply-templates select="title"/>`
[processes all `<title>` elements of the current node]
`<xsl:call-template name="title"/>`
[processes the named template, regardless of the source tree]

Push vs. Pull Processing

- In *push processing*, the source document controls the order of processing (e.g., CSS is strictly push processing); e.g.,
`<xsl:apply-templates/>`
- Pull processing can address particular elements in the source tree regardless of position in the source document; e.g.,
`<xsl:apply-templates select="//title"/>`

Selecting Elements and Attributes

- To select the contents of a particular element, use this `<xsl:select>` statement:
`<xsl:select value-of="XPATH STATEMENT"/>`
`<xsl:select value-of="title"/>`
- To select the contents of an attribute of a particular element, use an XPath statement like:
`<xsl:select value-of="title[@type]"/>`

Practical XML & XSLT

Roy Tennant, The California Digital Library

Decision Structure: Choose

- A way to process data differently based on specified criteria; if you don't need "otherwise", you can use <xsl:if>

```
<xsl:choose>
  <xsl:when test="SOME STATEMENT">
    CODE HERE TO BE EXECUTED IF THE STATEMENT IS TRUE
  </xsl:when>
  <xsl:when test="SOME OTHER STATEMENT">
    CODE HERE TO BE EXECUTED IF THE STATEMENT IS TRUE
  </xsl:when>
  <xsl:otherwise>
    DEFAULT CODE HERE, IF THE ABOVE TWO
    TESTS FAIL
  </xsl:otherwise>
</xsl:choose>
```

Decision Structure: If

- A decision structure when you don't need a default decision (otherwise use xsl:choose instead)

```
<xsl:if test="SOME STATEMENT">
  CODE HERE TO BE EXECUTED IF THE STATEMENT IS TRUE
</xsl:if>
<xsl:if test="SOME OTHER STATEMENT">
  CODE HERE TO BE EXECUTED IF THE STATEMENT
  IS TRUE
</xsl:if>
```

Decision Structure: Tests

Focusing in on: <xsl:when test="SOME STATEMENT">

Some examples of what "SOME STATEMENT" can be:

```
<xsl:when test="state='AZ'">Arizona</xsl:when>
[true when the contents of the <state> tag is equal to 'AZ']

<xsl:when test="@width">Width=<xsl:select value-
of="@width"/></xsl:when> [true when the attribute
"width" exists at the current node]
```

Looping

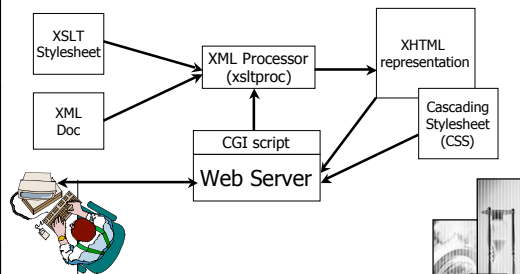
- XSLT looping selects a set of nodes using an Xpath expression, and performs the same operation on each; e.g.,
- ```
<xsl:for-each select=EXPRESSION>
 CODE HERE
</xsl:for-each>
```

## HTML in XSLT

- HTML codes can be inserted anywhere among the XSLT commands so long as:
  - You follow all XML tagging rules (e.g., all tags are properly nested, no disallowed character entities unless explicitly specified as CDATA)
  - You spell out the syntax when using XSLT within an HTML tag; e.g.,

```
<xsl:element name="a"><xsl:attribute name="href">
<xsl:value-of select="//url"/></xsl:attribute>
<xsl:value-of select="//book/title"/></xsl:element>
Becomes when rendered:
Book Title
```

## XSLT Demonstration




# Practical XML & XSLT

Roy Tennant, The California Digital Library



## Serving XML to Web Users

- Basic requirements: an XML doc and a web server
- Additional requirements for simple method:
  - A CSS Stylesheet
- Additional requirements for complex, powerful method:
  - An XSLT stylesheet
  - An XML parser
  - XML web publishing software or an in-house CGI or Java program to join the pieces
  - A CSS stylesheet (optional) to control how it looks in a browser



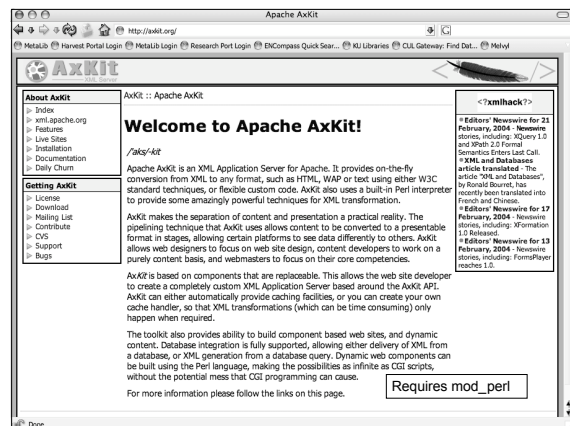
## XML Web Publishing Software

- Software used to add XML serving capability to a web server
- Makes it easy to join XML documents with XSLT to output HTML for standard web browsers
- A couple examples, both free...

The Apache Cocoon Project

Requires a Java servlet container such as Tomcat (free) or Resin (commercial)




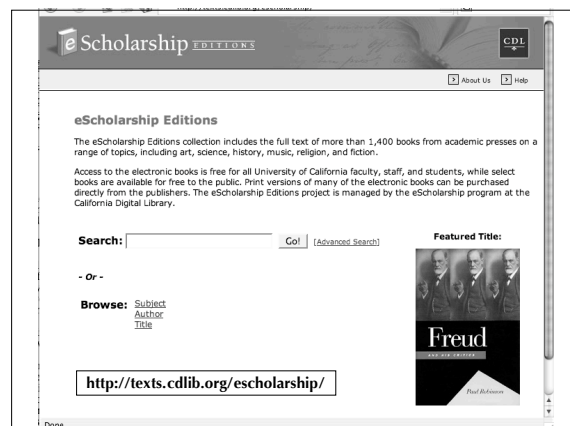
Apache AxKit

Welcome to Apache AxKit!

Requires mod\_perl

## Case Study: Publishing Books @ the California Digital Library

- Goals:
  - To create *highly usable* online versions of books
  - To create versions that will *migrate easily* as technology changes
  - To create an infrastructure that will support *dynamic presentations* of the same content

eScholarship Editions

The eScholarship Editions collection includes the full text of more than 1,400 books from academic presses on a range of topics, including art, science, history, music, religion, and fiction.

Access to the electronic books is free for all University of California faculty, staff, and students, while select books are available for free to the public. Print versions of many of the electronic books can be purchased directly from the publishers. The eScholarship Editions project is managed by the Scholarship program at the California Digital Library.

Search:   [Advanced Search](#)

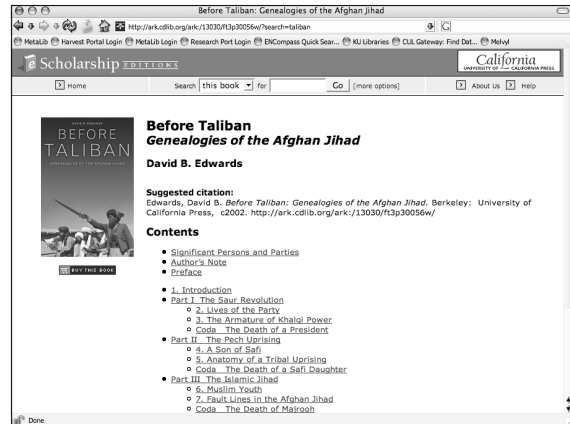
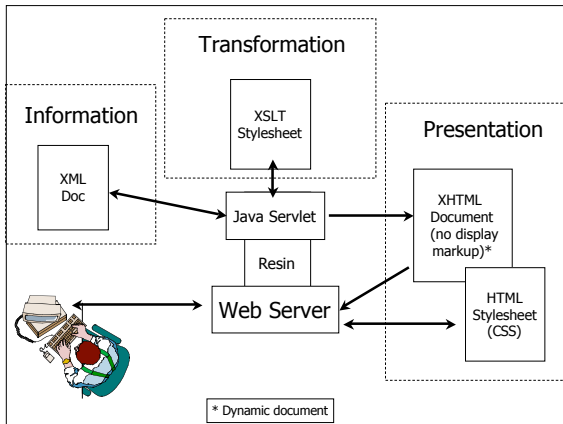
Featured Title:

**Freud**

<http://texts.cdlib.org/escholarship/>

# Practical XML & XSLT

Roy Tennant, The California Digital Library



## XML & XSLT Resources

- Eric Morgan's "Getting Started with XML" a good place to begin
- Many good web sites, and Google searches can often answer specific questions you may have
- Be sure to join the XML4Lib discussion

## Tips and Advice

- Begin transitioning to XML *now*:
  - XHTML and CSS for web files, XML for static documents with long-term worth
  - Get your hands dirty on a simple XML project
- *Do not rely on browser support of XML*
- DTDs? We don't need no stinkin' DTDs!
- Buy my book! (just kidding...)

## Contact Information

Roy Tennant  
 California Digital Library  
 roy.tennant@ucop.edu  
<http://escholarship.cdlib.org/rtennant/>  
 510-987-0476