XSLT: Using XML to transform other XML files

Introduction to databases
CSCC43 Winter 2011
Ryan Johnson

Thanks to Ramona Truta and Renee Miller for material in these slides

What is XSLT?

• An XML-based method for styling XML data
  – Tells web browser how to display data
  – Similar to CSS for HTML, but with absolute control over input
• Achieves (very) similar results to XQuery
  – Pro: supported by all non-ancient web browsers
  – Con: XML syntax is verbose

XSLT in a nutshell

• All XSL commands are xml (namespace xsl)
• Everything works by templates
  – Output generated by instantiating templates
  – No output unless some template matches document root
  – Templates can call other templates (recursion allowed)
• Builds on material we already know
  – XPath used to select nodes for various uses
  – Embedded expressions use {} syntax from XQuery

Control flow in XSL

• <xsl:for select='...'...</xsl:for>
  – Instantiate the template inside once for each item
  – Inside the template, context node (.) refers to “current” item
  – Optional first child: <xsl:sort>
• <xsl:if test='...'...</xsl:if>
  – Instantiate the template inside if the condition is true
  – No ‘else’ (use a second xsl:if or xsl:choose instead)
• <xsl:choose>...</xsl:choose>
  – Children: xsl:when+ xsl:otherwise?
  – <xsl:when test='...'...</xsl:when> (same usage as xsl:if)
  – <xsl:otherwise>...</xsl:otherwise> (no-match template)
  – Instantiate only one template (first match wins)
Declaring variables

- `<xsl:variable name='...' select='...'/>`
  - Store the result of an XPath query in a variable
- `<xsl:variable name='...'>...</xsl:variable>`
  - Instantiate a template and store its result in a variable

Notes

- Default value (if none specified) is an empty string
- Cannot be overwritten or updated once set

Declaring templates

- Idea: encapsulate a set of XSLT commands to be evaluated in a given context, with the result inserted into the output
- `<xsl:template match='...' mode='...'>`:
  - Template instantiated for every element satisfying the `match`'s XPath expression
  - Optional `mode` attribute allows to select from 2+ templates
- `<xsl:template name='...'>`:
  - Template instantiated explicitly
  - OK to name a `match` template

Applying vs. calling templates

- `<xsl:apply-templates select='...' mode='...'/>`
  - Format a node (default: .) using “best” matching template
  - Complicated priority rules, but more-specific and declared-later templates usually chosen
  => When in doubt disambiguate using optional `mode` attribute
- `<xsl:call-template name='...'/>`
  - Instantiate the named template
  - WARNING: context node unchanged by call!
  => Call a template which uses ‘.’ ==> possible infinite loop
- `<xsl:parameter>` and `<xsl:with-param>`
  - Templates can declare `xsl:parameter` (acts just like `xsl:variable`)
  => Same effect if template is called normally
  - Template call can have `xsl:with-param` child elements
  => Overrides value which `xsl:parameter` would have assigned

Outputting things

- `<xsl:text>...</xsl:text>`
  - Adds text (including whitespace!) to output
  - Optional yes/no attribute: `disable-output-escaping`
  => In theory, good for outputting ‘&’, ‘<’ etc.
  => WARNING: optional part of spec, often ignored
- `<xsl:value-of select='...'/>`
  - Converts the selected node(s) to text and outputs it
- `<xsl:copy-of select='...'/>`
  - Embeds the selected node(s) directly into the output
  - e.g. good for emitting snippets of pre-constructed HTML
Global output controls

- `<xsl:output method='...' indent='...'/>`
  - Methods include xml (default), html, and text
  - Defaults to html method if parser detects HTML output
  - Enabling indentation makes output prettier
  - Bad idea if whitespace matters for presentation
  - Lots of other options (look them up)

Whitespace handling

- Default: any text node with non-space characters preserves all space
- Override #1: `<xsl:strip-space elements='...'/>`
  - Specifies that the elements named (separated by spaces) should normalize whitespace before output
  - A corresponding preserve-space command exists also
  - Same conflict resolution as used for template matching
- Override #2: `<... xml:space='preserve' ...>`
  - Little-known xml attribute that tells XML processors whether whitespace matters
  - Inherited by child elements (xml:space='default' cancels)
  - Watch out for elements forbidding text (e.g. `xsl:call-template`)

Keys

- Like the ID/IDREF from DTD but better
  - No need for the DTD (often ignored even when present)
  - Multiple (named) ID domains allowed
- `<xsl:key name='...' match='...' use='...'/>`
  - Declares a key (name) for all nodes which match, as well as how to find (use) the value of the key
- `{key('...', ...)}` retrieves the value of a key
- Example:
  - Find all links which point to an image in the same file
    `<xsl:key name='imglinks' match='//a' use='@href'/>`
    `{key('imglinks', //img/@src)}`

Sample XSLT skeleton

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        ... 
    </xsl:template>
</xsl:stylesheet>
```
Sample: root node template

```xml
<xsl:template match='/'>
  <html><body>
    <xsl:call-template name='show-tree'>
      <xsl:with-param name='e' select='*'/>
      <xsl:with-param name='i' select='0'/>
    </xsl:call-template>
  </body></html>
</xsl:template>
<xsl:template match='*'>
  <xsl:value-of select='name(.)'/>
</xsl:template>
<xsl:template match='nct_id'>
  <span style='color:red'>
    <xsl:value-of select='self::text()/'></xsl:value-of>
  </span>
</xsl:template>
```

Elements template applies to HTML to embed the template in

- Instantiate show-tree template, passing e=* and i=0 as parameters
- By default, output the name of each element
- nct_id is important => special treatment

Sample: recursive template

```xml
<xsl:template name='show-tree'>
  <xsl:param name='e'/>
  <xsl:param name='i'/>
  <div style='padding-left:{$i}px'>
    <xsl:apply-templates select='name($e)'/>
    <xsl:if test='$e/@*'>
      ( <xsl:for-each select='$e/@*'>
        <xsl:value-of select='name(.)'/>,
      </xsl:for-each> )
    </xsl:if>
    <xsl:for-each select='$e/*'>
      <xsl:call-template name='show-tree'>
        <xsl:with-param name='e' select='.'/>;
        <xsl:with-param name='i' select='30'/>
      </xsl:call-template>
    </xsl:for-each>
  </div>
</xsl:template>
```

- Named templates must be "called"
- Template parameters
  - Compute div's padding
  - Format the element
  - List attributes, if any exist (surrounded by parens, separated by commas)
- Recursively apply template to children

Sample XSLT + clinical_study XML

**Rendered**

- clinical_study (rank ,)
- required_header
  - download_date
  - link_text
  - url
  - id info
    - org_study_id
    - secondary_id
    - nct_id: NCT01111110
- brief_title
- official_title

**HTML**

```html
<html><body><div style="margin-left:0px;">clinical_study
  (rank ,
  )
  required_header<div style="margin-left:30px;">download_date</div>
  link_text
  url
  id info
    org_study_id
    secondary_id
    nct_id: NCT01111110
  brief_title
  official_title
  ...</div>
</body></html>
```