More XML
XPATH, XSLT

CS 431 - February 27, 2006
Carl Lagoze - Cornell University
XPath

- Language for addressing parts of an XML document
  - XSLT
  - Xpointer
- Tree model similar to DOM
- W3C Recommendation (1999)
  - [http://www.w3.org/TR/xpath](http://www.w3.org/TR/xpath)
Remember to think in terms of DOM trees

```xml
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <title lang="en">XML Basics</title>
</book>
```
Xpath Concepts

• **Context Node**
  - current node in XML document that is basis of path evaluation
  - Default to root (remember that root is “Document”)

• **Location Steps - selection from context node**
  - **Axis** - sub-tree(s) selection from context node
  - **Node Test** - select specific elements or node type(s)
  - **Predicates** - predicate for filtering after axis and node tests
Context, Axis, Node Test, Predicate
Location Path Specification

- /step/step/…. - absolute from document root
- step/step .... - relative from context
- //step/step - anywhere in document tree

- where step is: axis::node-test[predicate]
• child:: all children of context
• descendant:: all children, grandchildren, ...
• parent:: parent of context
• ancestor:: all nodes on path to root from context
• **Element name:** e.g. “Book”
  - make sure to pay attention to namespaces!!!!

• **Wildcard:** *

• **Type():** where type is “node”, “text”, etc.
  - *Remember in DOM that everything is a node*
• Boolean and comparative operators
• Types
  - Numbers
  - Strings
  - node-sets (the set of nodes selected)
• Functions
  - Examples
    • boolean starts-with(string, string)
    • number count(node-set)
    • number position()
xpath examples

- /child::source/child::AAA
  - or /source/AAA since child is default axis
- /child::source/child::*[position()=2]
  - or /source/*[2]
- /child::source/child::AAA[position()=2]/attribute::id
  - or /source/AAA[2]/@id
- /child::source/child::AAA/@*
  - or /source/AAA/@*
- /child::source/child::AAA[contains(.,'a1')]
XML Transformations (XSLT)

• **Origins: separate rendering from data**
  - Roots in CSS

• **W3C Recommendation**
  - [http://www.w3.org/TR/xslt](http://www.w3.org/TR/xslt)

• **Generalized notion of transformation for:**
  - Multiple renderings
  - Structural transformation between different languages
  - Dynamic documents

• **XSLT - rule-based (declarative) language for transformations**
Stylesheet Document or Program

- XML document rooted in `<stylesheet>` element
- XSL tags are in namespace
  http://www.w3.org/1999/XSL/Transform
- Body is set of templates or rules
  - match attribute specifies xpath of elements in source tree
  - Body of template specifies contribution of source elements to result tree
Stylesheet Document or Program

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="para">
    <p>
      <xsl:apply-templates/>
    </p>
  </xsl:template>
  <xsl:template match="emphasis">
    <b>
      <xsl:apply-templates/>
    </b>
  </xsl:template>
</xsl:stylesheet>
```
XSL Execution Model

- Templates represent a set of rules
- Rule matching is done within current tree context
- Rules are not executed in order
- Default behavior is depth-first walk of tree, outputting element values

Template Form

```xml
<xsl:template match="para">
  <p>
    This is the sentence
    <xsl:apply-imports/>
  </p>
</xsl:template>
```

- Elements from `xsl` namespace are transform instructions
- `match` attribute value is xpath expression setting rule for execution of body
- Sequential execution within template
- Non-xsl namespace elements are literals.
- `<xsl:apply-templates>`
  - set context to next tree step (default depth-first)
  - re-evaluate rules
Result Tree Creation

- Literals - any element not in xsl namespace is inserted into result tree

```xml
<xsl:template match="para">
  <p>
    This is the sentence
    <xsl:apply-templates/>
  </p>
</xsl:template>
```
Result Tree Creation

- `<xsl:text>` - send content directly to output (retain whitespaces)
Result Tree Creation

- `<xsl:value-of>` - extract element values (anywhere in the tree)
Result Tree Creation

• `<xsl:copyof>` - Copy selected nodes into result tree

  `<xsl:copy-of select="table"/>`
Result Tree Creation

- `<xsl:element>` - instantiate an element
- `<xsl:attribute>` - instantiate an attribute

```xml
<xsl:element name="newEl">
  <xsl:attribute name="newAttr">
    <xsl:value-of select="/top/AAA[1]"/>
  </xsl:attribute>
</xsl:element>
```
Default Rules (Must replace to change them)

```
<xsl:template match="*/">
  <xsl:apply-templates/>
</xsl:template>
```

• Applies to root node and element nodes

• Recurses depth first

```
<xsl:template match="text()|@*">
  <xsl:value-of select="."/>
</xsl:template>
```

• Applies to text and attribute nodes

• Copies value to output tree
A simple example

• XML base file

• XSLT file
Modifying rule set and context

- **Context setting**
  - `<xsl:apply-templates select="//bar">`
  - Modifies default depth-first behavior
- There are conflict resolution rules

Modifying rule set and context

- **Mode setting**
  - `<xsl:apply-templates mode="this"/>`
  - `<xsl:template match="foo" mode="this"/>`
  - `<xsl:template match="foo" mode="that"/>`

Namespaces in XSLT

• The XSL document MUST know about the namespaces of elements that it references (via XPath expressions) in the instance document

• Watch out for the default namespace!!
XSLT Procedural Programming

- Sequential programming style
- Basics
  - `for-each` - loop through a set of elements
  - `call-template` - like a standard procedure call
For-each programming example

• **XML base file**

• **XSLT file**
Call-template programming example

• XML base file

• XSLT file
Various other programming constructs

• **Conditionals**

• **Variables (declaration and use)**
  - Once set, can’t be reset
  - Functional programming style
  - Use recursion

• **Some type conversion**

• **Parameters**

• **Sorting**
Associating an XML document with a transform

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <para>
    this is
    <emphasis>
      big
    </emphasis>
    text
  </para>
</xsl:stylesheet>
```