Purpose and scope of XSLT

Welcome

- Structure and method of course
- Reference book
- Warning about XSLT 2 and XSLT 1
- Examples being used:
  - RSS news feed
  - Issues of Punch magazine
  - Cemetery records
  - Dummy student records
- using oXygen

What we will cover on this course

- most of the XSLT 1.0 language
- XPath notation
- using XSLT and XPath from the oXygen editor and on the command line
- starting XSLT 2.0

Questions we will (probably) not try to answer on this course

- can I set up a web publishing framework using XSL?
- how do I embed XSL in my Java/PHP/Perl program?
- how do I process XML if I don't like XSL?

Typical scenarios

- transforming a TEI XML file to HTML
- extracting information from an XML data file
- cleaning XHTML documents
- transforming RSS XML into HTML
- processing Word and OpenOffice XML document formats

Reminder: an example XML document


```
<?xml version="1.0" encoding="utf-8" ?>
<cookBook>
 <recipe n="1">
  <head>Nail Soup</head>
  <ingredientList>
   <ingredient>an onion</ingredient>
   <ingredient>two carrots</ingredient>
   <ingredient>water</ingredient>
      ...
  <ingredient>a nail</ingredient>
   <ingredient>some gullible peasants</ingredient>
  </ingredientList>
  <procedure>
   <step>put the water on to boil</step>
      ....
  <step>take out the nail and serve</step>
  </procedure>
 </recipe>
 <recipe n="2">
<!-- contents of second recipe here -->
 </recipe>
<!-- hic desunt multa -->
</cookBook>
```

Reminder: the rules of the XML Game

- An XML document represents a (kind of) tree
- It has a single root and many nodes
- Each node can be
  - a subtree
  - a single element (possibly bearing some attributes)
  - a string of character data
- Each element has a type or generic identifier
- Attribute names are predefined for a given element; values can also be constrained

Reminder: defining the rules

A valid XML document conforms to rules which are stated in an external schema of some sort.
A schema specifies:

- the name of the root element
- names for all elements used
- names and datatypes and (occasionally) default values for their attributes
- rules about how elements can nest
- and a few other things, depending on the schema language

Reminder: namespace declarations

An XML document can use elements declared in different name spaces.

- a namespace declaration associates a namespace prefix with an external identifier (which looks like an URL)
- the default current namespace may be declared using a special `xmlns` attribute
- the current namespace is inherited until some element changes it
- other name spaces can be declared and associated with a special prefix
- the namespace is not inherited from elements which use the prefixes

Reminder: namespace examples


```
<TEI xmlns="http://www.tei-c.org/ns/1.0"
>
<p xmlns="http://www.tei-c.org/ns/1.0"
>Hello world</p>
</TEI>
```

```
<atom:feed>
 <atom:entry>
  <atom:author>
   <atom:name>Peter Robinson</atom:name>
  </atom:author>
  <atom:category term="h07"/>
  <atom:content type="xhtml">
   <xhtml:div>
    <xhtml:p>hello world </xhtml:p>
   </xhtml:div>
  </atom:content>
 </atom:entry>
</atom:feed>
```

There is a special xml namespace, used by for global attributes xml:id and xml:lang

What is the XSL family?

- XPath: a language for expressing paths through XML trees
- XSLT: a programming language for transforming XML
- XSL FO: an XML vocabulary for describing formatted pages

XSLT

The XSLT language is

- expressed in XML; uses namespaces to distinguish output from instructions
- purely functional
- reads and writes XML trees
- designed to generate XSL FO, but now widely used to generate HTML

How is XSLT used? (1)

- With a command-line program to transform XML (eg to HTML)
  - Downside: no dynamic content, user sees HTML
  - Upside: no server overhead, understood by all clients
- In a web server *servlet*, eg serving up HTML from XML (eg Cocoon, XTF, Axkit)
  - Downside: user sees HTML, server overhead
  - Upside: understood by all clients, allows for dynamic changes

How is XSLT used? (2)

- In a web browser, displaying XML on the fly
  - Downside: many clients do not understand it
  - Upside: user sees XML
- Embedded in specialized program
- As part of a chain of production processes, performing arbitrary transformations

XSLT implementations

MSXML	Built into Microsoft Internet Explorer
Saxon	Java-based, standards leader, implements XSLT 2.0 (basic version free)
Xalan	Java-based, widely used in servlets (open source)
libxslt	C-based, fast and efficient (open source)
transformiix	C-based, used in Mozilla (open source)

What is a transformation?

Take this:

```
<people>
 <name>Corey Burger</name>
 <name>Naaman Campbell</name>
 <name>Milo Casagrande</name>
 <name>Korky Kathman</name>
 <name>Jonathan Riddell</name>
 <name>Jerome Gotangco</name>
</people>
```

and make this:

```
<n>1</n>
<sname>Burger</sname>
<n>2</n>
<sname>Campbell</sname>
<n>3</n>
<sname>Casagrande</sname>
<n>4</n>
<sname>Gotangco</sname>
<n>5</n>
<sname>Kathman</sname>
```

A text example

Take this

```
<recipe n="34">
 <title>Pasta for beginners</title>
 <ingredients>
  <item>Pasta</item>
  <item>Grated cheese</item>
 </ingredients>
 <cook>Cook the pasta and mix with the cheese</cook>
</recipe>
```

and make this

```
<html>
 <h1>34: Pasta for beginners</h1>
 <p>Ingredients: Pasta Grated cheese</p>
 <p>Cook the pasta and mix with the cheese</p>
</html>
```

How do you express that in XSL?


```
<xsl:stylesheet version="1.0">
 <xsl:template match="recipe">
  <html>
   <h1>
    <xsl:value-of select="@n"/>:
   <xsl:value-of select="title"/>
   </h1>
   <p>Ingredients:
   <xsl:apply-templates select="ingredients/item"/>
   </p>
   <p>
    <xsl:value-of select="cook"/>
   </p>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Structure of an XSL file


```
<xsl:stylesheet version="1.0">
 <xsl:template match="div">
<!-- .... do something with div elements....-->
 </xsl:template>
 <xsl:template match="p">
<!-- .... do something with p elements....-->
 </xsl:template>
</xsl:stylesheet>
```

The `div` and `p` are *XPath expressions*, which specify which bit of the document is matched by the template.

Any element not starting with xsl: in a template body is put into the output.

The Golden Rules of XSLT

1. If there is no template matching an element, we process the elements inside it
2. If there are no elements to process by Rule 1, any text inside the element is output
3. Children elements are not processed by a template unless you explicitly say so:
4. `xsl:apply-templates select="XX"` looks for templates which match element "XX"; `xsl:value-of select="XX"` simply gets any text from that element
5. The order of templates in your program file is immaterial
6. You can process any part of the document from any template
7. Everything is well-formed XML. Everything!

A simple test file


```
<TEI>
 <text>
  <front>
   <p>Material up front</p>
  </front>
  <body>
   <div>
    <head>Introduction</head>
    <p rend="it">Some sane words</p>
    <p>Rather more surprising words</p>
   </div>
  </body>
  <back>
   <p>Material in the back</p>
  </back>
 </text>
</TEI>
```

Feature: apply-templates


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

```
<xsl:template match="tei:TEI">
 <xsl:apply-templates select="tei:text"/>
</xsl:template>
```

```
<xsl:template match="tei:text">
 <h1>FRONT MATTER</h1>
 <xsl:apply-templates select="tei:front"/>
 <h1>BODY MATTER</h1>
 <xsl:apply-templates select="tei:body"/>
</xsl:template>
```

Feature: `value-of`

Templates for paragraphs and headings:

```
<xsl:template match="tei:p">
 <p>
  <xsl:apply-templates/>
 </p>
</xsl:template>
<xsl:template match="tei:div">
 <h2>
  <xsl:value-of select="tei:head"/>
 </h2>
 <xsl:apply-templates/>
</xsl:template>
<xsl:template match="tei:div/tei:head"/>
```

Notice how we avoid getting the heading text twice.

Why did we need to qualify it to deal with just <head> inside ?