Basic XSLT

Code: xslt-basics

Author and version

- Daniel K. Schneider
- Email: Daniel.Schneider@tecfa.unige.ch

- Version: 0.9 (modified 29/4/07 by DKS)

Objectives

- Understand the purpose of XSLT
- Do simple transformations from XML to HTML
- Understand the most simple XPath expressions (tag names)

Prerequisites

- Editing XML (being able to use a simple DTD)
- XML namespaces (some)
- HTML and CSS (some)

Warning

- XSLT is a rather complex transformation language
- I believe that one could distinguish four levels of difficulty:
  - **These slides** are introductory (level 1)
  - Level 2 XSLT is more sophisticated template ordering, conditional expressions, loops, etc.
  - Level 3 is advanced XPath expressions
  - Level 4 is functional programming with templates

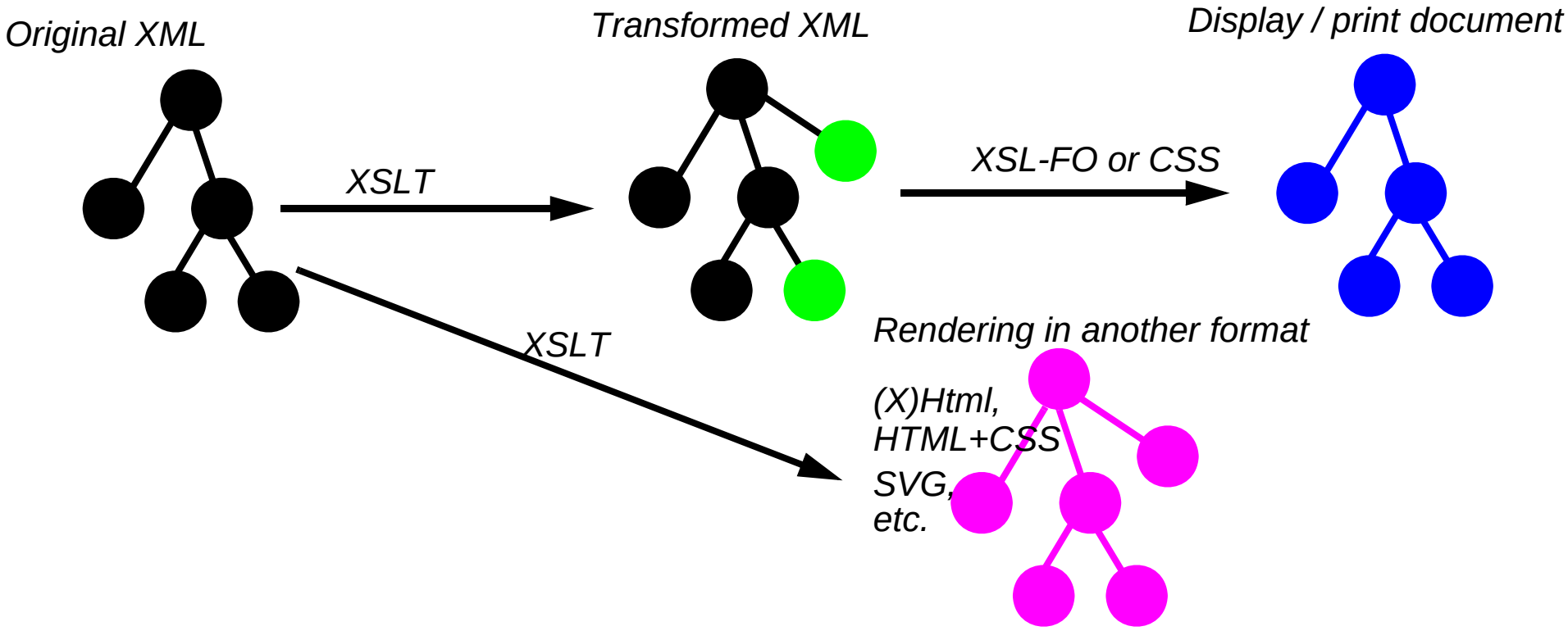
Disclaimer

- There may be typos (sorry) and mistakes (sorry again)
- Please also consult a textbook !

Contents

1. Introduction Extensible Stylesheet Language Transformations

- Goals of XSLT
- XSLT is a transformation language for XML
- XSLT is a W3C XML language (the usual XML well-formedness criteria apply)
- XSLT can translate XML into almost ***anything***, e.g.:
  - wellfomed HTML (closed tags)
  - any XML, e.g. yours or other XML languages like SVG, X3D
  - non XML, e.g. RTF (a bit more complicated)



1.1 History and specifications

Specification

- XSLT 1.0 was formalized as W3C Recommendation on 16/11/99

  url: **http://www.w3.org/TR/xslt**
- XSLT 2.0 is a W3C recommendation since 23 January 2007

  url: **http://www.w3.org/TR/xslt20/**
  - not implemented in current browsers, but in most good XSLT processors (e.g. Saxon and Xalan)

History

- Initially, XLS (XSL: eXtensible Stylesheet Language) was a project to replace CSS for both display and print media and to provide support for complex formatting and layout features (pagination, indexing, cross-referencing, recursive numbering, indexing, etc.
- XSLT (**Extensible Stylesheet Language Transformations**) was originally intended as a small part of the larger specification for XSL
- However, when the XSL specification draft became very large and complex it was decided to split the project into XSLT for transformations (that were urgently needed) and XSL for the rest (W3C recommendation of 2001)

Related languages

- XPath (XML Path language used by XSLT, XQuery, etc.), **http://www.w3.org/TR/xpath**
- XSL also called XSL/FO (the formatting language), **http://www.w3.org/TR/xsl/**
- XQuery (Query language for XML), **http://www.w3.org/TR/xquery/**

2. A first glance at XSLT

2.1 Root of an XSLT file stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
....
</xsl:stylesheet>
```

Mandatory elements

- XML declaration on top of the file
- A `stylesheet` root tag with the following version and namespace attributes:

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

- XSLT must be wellformed (and also obey the XSLT specification)
- XSLT files usually have the *.xsl extension and should have the `text/xsl` or `application/xml` mimetype when served by http.

2.2 Association of XML and an XSLT file

- An XSLT stylesheet is associated with a processing instruction (similar to a CSS stylesheet)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="project.xsl" type="text/xsl"?>
<yourxml> .... </yourxml>
```

2.3 Basic XSLT

Basic (!) use of XSLT means:

- writing **translation rules** (aka templates) for each XML tag we want to translate
- translating XML to HTML

A simple translation rule (called "template" in XSLT):

```
<xsl:template match="XML_tag_name">
  .... contents to produce (i.e. HTML)
  .... further instructions
</xsl:template>
```

Selector of XML elements to transform

A mix of contents to produce plus further instructions

Example 2-1: Translation of a title tag into HTML centered H1

XML Source we want to translate:

```
<title>Hello friend</title>
```

The XSLT rule that does it:

```
<xsl:template match="title">
   <h1 align="center">
       <xsl:apply-templates/>
   </h1>
</xsl:template>
```

Rule applies to "title"

HTML code produced, i.e. the translation !

Means "move on": find another rule or copy contents

2.4 A complete XSLT example

Example 2-2: Hello XSLT

XML file (source)

url: **hello.xml**

```
<?xml version="1.0"?>
<?xml-stylesheet href="hello.xsl" type="text/xsl"?>
<page>
 <title>Hello</title>
 <content>Here is some content</content>
 <comment>Written by DKS.</comment>
</page>
```

Wanted result document

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/REC-
html40/strict.dtd">
<html>
  <head>
    <title>Hello</title>
  </head>
  <body bgcolor="#ffffff">
    <h1 align="center">Hello</h1>
    <p align="center"> Here is some content</p>
    <hr><i>Written by DKS</i>
  </body>
</html>
```

The XSLT Stylesheet

url: **hello.xslt**

- See next slides for explanations ...

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="page">
   <html> <head> <title> <xsl:value-of select="title"/> </title> </head>
    <body bgcolor="#ffffff">
     <xsl:apply-templates/>
    </body>
   </html>
  </xsl:template>

  <xsl:template match="title">
   <h1 align="center"> <xsl:apply-templates/> </h1>
  </xsl:template>

  <xsl:template match="content">
   <p align="center"> <xsl:apply-templates/> </p>
  </xsl:template>

  <xsl:template match="comment">
   <hr/> <i><xsl:apply-templates/></i>
  </xsl:template>
</xsl:stylesheet>
```

Anatomy of a simple stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="page">
   .......
 <html>
   <head> <title>
       <xsl:value-of select="title"/>
   </title> </head>
   <body bgcolor="#ffffff">
     <xsl:apply-templates/>
   </body>
   </html>
</xsl:template>

<xsl:template match="title">
   <h1 align="center"> <xsl:apply-templates/> </h1>
</xsl:template>

...........

</xsl:stylesheet>
```

XSLT start

Rule dealing with the document's root element

Extract the value of the title element

An other rule

XSLT end

2.5 Rule execution order

1. The XSLT engine first looks at the XML file and tries to find a XLT rule that will match the root element
   - E.g. in the above example it will find "page" and then the template for page
2. The XSLT processor will then "move" inside the rule element and do further processing
   - HTML elements (or any other tags) will be copied to the output document
   - If an XSLT instruction is found, it will be executed
3. **`<xsl:apply-templates/>`** means: "go and look for other rules"

   E.g. in the above example
   - the processor dealing with root element "page" will first find a rule for "title" and execute it according to the same principle.
   - once it is done with "title" and its children, it then will find the rule for "content" and execute it
   - and so forth ....

More information

- **`<xsl:value-of select="title"/>`** will retrieve contents of the "title" child element.
  - In our example, it would only work in the template for "page", since only "page" has a "title" child
- You have to understand that XSLT works down "depth-first" the XML tree, i.e.
  - it first deals with the rule for the root element,
  - then with the first instruction within this rule.
  - If the first instruction says "find other rules" it will then apply the first rule found for the first child element and so forth...
  - The rule of the root element is also the last one be finished (since it must deal step-by-step with everything that is found inside) !!!

2.6 The procedure recapitulated

1. Create a XSLT stylesheet file: xxx.xsl
2. Copy/paste the XSLT header and root element below (decide encoding as you like)

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

</xsl:stylesheet>
```

3. Write a rule that deals with your XML root element
   - This rule must produce the root, head and body of the HTML (copy/paste this too, but replace "**page**")

```
<xsl:template match="page">
  <html>
  <head> <title> <xsl:value-of select="title"/> </title> </head>
  <body bgcolor="#ffffff">
     <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>
```

4. Write rules for **each** (!!) of your XML elements,
   - for each insert some HTML, sometimes some text, or sometimes nothing
   - make sure to place a <xsl:apply-templates> inside each rule (usually between some HTML) ... unless you wish to censor contents.
5. Associate this stylesheet with your XML file using:

```
<?xml-stylesheet href="xxx.xsl" type="text/xsl"?>
```

3. Tuning output with xsl:output and CSS

3.1 Output declarations

- So far, HTML output produced would display in a naviagor, but is not fully HTML compliant.

xsl:output

is an instruction that allows you to fine-tune XSLT translation output

```
Syntax: <xsl:output
     method = "xml" | "html" | "text"
     version = nmtoken
     encoding = string
     omit-xml-declaration = "yes" | "no"
     standalone = "yes" | "no"
     doctype-public = string
     doctype-system = string
     indent = "yes" | "no"
     media-type = string />
```

- To put in the beginning of the file (after xsl:stylesheet)

Example 3-1: Output in HTML ISO-latin encoded:

```
<xsl:output method="html"
    encoding="ISO-8859-1"
    doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN"/>
```

Example 3-2: Output in XHTML transitional with a namespace

- This is quite more complicated than producing simple HTML

```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/1999/xhtml">
<xsl:output
   method="xml"
   doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
   doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
   indent="yes"
   encoding="iso-8859-1" />
<xsl:template match="recipe">
   <html xmlns="http://www.w3.org/1999/xhtml">
    <head> ... </head> ... <body> ... </body>
</xsl:template>
```

Exemple 3-3: Your XML

```
<xsl:output
 method="xml" indent="yes"
 doctype-system="mydtd.dtd" />
```

Exemple 3-4: Output in SVG

```
<xsl:output
 method="xml"
 indent="yes"
 standalone="no"
 doctype-public="-//W3C//DTD SVG 1.0//EN"
 doctype-system="http://www.w3.org/TR/2001/PR-SVG-20010719/DTD/svg10.dtd"
 media-type="image/svg" />
```

3.2 CSS styling of HTHML

- Associating a CSS stylesheet with HTML output is trivial:
  - add a link tag in the "head" produced by the template for the root element
  - .... in the hello.css file you then have to define styles of HTML elements you generate

```
<xsl:template match="hello">
 <html>
  <head>
   <link href="hello.css" type="text/css" rel="stylesheet"/>
  </head>
  ......
</xsl:template match="hello">
```

Example 3-5: cooking

url: **cooking.xsl, cooking.xml and cooking-html.css**

```
<xsl:template match="recipe">
  <html xmlns="http://www.w3.org/1999/xhtml">
   <head>
     <title> <xsl:value-of select="title"/> </title>
     <link href="cooking-html.css" type="text/css" rel="stylesheet"/>
   </head>
    <body bgcolor="#ffffff">
    <xsl:apply-templates/>
   </body>
  </html>
 </xsl:template>
```

4. If things go wrong

4.1 Frequent problems and remediation

Style-sheet error !

- Validate the style-sheet in your XML editor
- If it provides XSLT support, it will help you find the error spots

XHTML doesn't display in Firefox !

- Firefox wants a namespace declaration in the XHMTL produced, do it (see above).

HTML doesn't seem to be right !

- Transform the XML document within your XML editor and look at the HTML

  In "Exchanger Lite", use **Transform** in the menu bar with the following parameters:

  - Transform->Execute Advanced XSLT
  - Input = current document
  - XSLT = Use Processing instructions
- You also may validate the output HTML !

There is various unformatted text in the output !

- See next slide

HTML still doesn't seem to be right !!

- Use a XSLT debugger/tracer to understand how your XSLT executes

4.2 The XSLT default rule

- When you test your first style sheet, it is likely that some of your contents will appear non-formatted.
- This is due to the fact that XSLT will apply a default rule to all XML elements for which it didn't find a rule.
  - If you forget to write a rule for a tag (or misspell tag names) this will happen .....
- The XSLT default rule simply copies all contents to the output.

A modified default rule that will help you find missing pieces

- simply cut/paste this to your XSLT (but remove it later on)

```
<xsl:template match="*">
 <dl><dt>Untranslated node:
     <strong><xsl:value-of select="name()"/></strong></dt>
 <dd>
  <xsl:copy>
    <xsl:apply-templates select="@*"/>
    <xsl:apply-templates select="node()"/>
  </xsl:copy>
</dd>
</dl>
</xsl:template>

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

5. Selective processing

5.1 Steering rule execution and information filtering

- Instead of letting XSL apply rules in "natural order", you can tell which rules to apply when.

Example 5-1: Hello without content

url: **hello2.xml and hello2.xsl**

- The rule for the root element will only "call" the rules for the "title" and the "comment" element
- Information within a content tag will not be displayed (since we don't let the processor find rules by itself, but only let it execute a rule for "title" and another for "comment").

```
<xsl:template match="page">
   <html> <head> </head>
    <body bgcolor="#ffffff">
     <xsl:apply-templates select="title"/>
     <xsl:apply-templates select="comment"/>
    </body> </html>
  </xsl:template>

  <xsl:template match="title">
   <h1 align="center"> <xsl:apply-templates/> </h1>
  </xsl:template>
  <xsl:template match="comment">
   <hr/> <i><xsl:apply-templates/></i>
  </xsl:template>
```

5.2 A short glance at Xpath (optional)

- XPath is a very powerful language to extract information from XML
- XPath was published at the same time as XSLT 1.0 (1999)
- Values of XSLT match and select attributes are XPath expressions

Simple XPath expressions

```
<xsl:template match="page">
  <xsl:apply-templates select="title" />
</xsl:template>
```

- XSLT beginners don't need to know a lot about XPath (so don't worry right now !).
  - Simply stick to the idea of writing a template for each XML tag, as explained before
- XPath expressions can be more complicated:

  `<xsl:apply-templates select="course/module[position()=1]/section[position()=2]"/>`
  means: "find rule for 2nd section of the first module of course"
- XPath also includes arithmetics and tests

  `"//Participant[string-length(Nom)>=8]"`

  means: "return all participant nodes with content of name longer than 7 characters"

For your information only: examples of a few simple XPath expressions (optional !)

- These should remind you of CSS selectors

Syntax elemen	(Type of path)	Example path	Example matches
tag	element name	project	***<project> ..... </project>***
/	separates children	project/title	<project>***<title> ... </title>***
		/	(root element)
//	descendant	project//title	<project><problem>***<title>....</title>***
		//title	<racine>...***<title>..</title>*** (any place)
*******	"wildcard"	*/title	<bla>***<title>..</title> and*** <bli>***<title>...</title>***
|	"or operator	title|head	<title>...</title> ***or*** <head> ...</head>
		|/|@	All elements: root, children and attributes
.	current element	.	
../	parent element	../problem	<project>
@	attribute name	@id	<xyz id="test">...</xyz>
element/@attr	attribute of child	project/@id	***<project id="test" ...> ... </project>***
@attr='type'	type of attribute	list[@type='ol']	***<list type="ol"> ..... </list>***

5.3 Basic value extraction

xsl:value-of

- inserts the value of an XPath expression and copies it to the output
- e.g. you can take contents of an element or attribute values and insert them in HTML table cells.

Example 5-2: Value-of

- Let's assume that we have an author element and that we would like to put this information on top of the page and that we should like to display the value of the revision attribute.

XML fragment

```
<page>
 <title>Hello</title>
 <content revision="10">Here is some content</content>
 <comment>Written by <author>DKS</author></comment>
</page>
```

XSLT rules

```
<xsl:template match="page">
    <P><xsl:value-of select="comment/author"/></P>
</xsl:template>

<xsl:template match="content">
    <P>Revision number: <xsl:value-of select="@revision"/></P>
</xsl:template>
```

5.4 Inserting a value inside a string

- If you want to insert information inside an HTML attribute value, things get a little bit tricky.

There is a special syntax:

Syntax: `{....}`

This is the equivalent of `<xsl:value-of select="..."/>` which can not be used here !!

Example 5-3: Building a href tag with an email

- We will use both the {...} and the value-or select constructs.

The XML information

```
<contact-info email="test@test">
```

The XSLT rule

```
<xsl:template match="contact-info">
....
  <a href="mailto:{@email}"><xsl:value-of select="@email"/></a>
...
```

The result

```
<a href="mailto:test@test">test@test</a>
```

5.5 Dealing with pictures

- There is no special "magic" for dealing with images, links, stylesheets etc. Simply:
  - look at your XML and figure out how to translate into equivalent HTML (or whatever else)
  - the following example demonstrates the use of value extraction
  - several other solutions than the one demonstrated exist ...

Example 5-4: Dealing with pictures

XML file with picture file names

url: **images.xml**

```
<?xml version="1.0"?>
<?xml-stylesheet href="images.xsl" type="text/xsl"?>
<page>
 <title>Hello Here are my images</title>
 <list>
   <!-- pictures are either contents or attribute values of elements -->

   <image>dolores_001.jpg</image>
   <image>dolores_002.jpg</image>

   <image3 source="dolores_002.jpg">Recipe image</image3>
 </list>
 <comment>Written by DKS.</comment>
</page>
```

Excerpts from the XSLT stylesheet

url: **images.xsl**

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

<!-- pictures are either contents or attribute values of elements -->
<xsl:template match="list">
   Images are element contents, apply a template to all image elements:
   <xsl:apply-templates select="image"/>

   Images are attribute values of an element, we do it differently:
   <xsl:apply-templates select="image3"/>
</xsl:template>
```

- This rule will insert the content of the image element into the value of src="".

```
<xsl:template match="image">
  <p> <img src="{.}"/> </p>
</xsl:template>
```

- This rule will insert the value of the source attribute into the value of src and also insert the contents of the the image3 element.

```
<xsl:template match="image3">
  <p> <img src="{@source}"/><br/>
  <!-- will insert text element contents -->
  <xsl:value-of select="."/> </p>
</xsl:template>
```

6. Next steps

6.1 Reading

Deitel Textbook chapter 20, section 20.9 Extensible Stylesheet Language (XSL) 665-673.

Recall that XSLT is a complex language (we only presented some basics !), so may turn to an XML or XSLT textbook for more ....

6.2 Homework

Take an XML file of your choice from module 6 example files and write an XSLT stylesheet

6.3 Next modules

Module 8

- Review of major Web Standards
- Webservers and server-side scripting (CGI)
- Scripting languages overview (ASP, PHP)
- Outlook: AJAX and web services (demonstration of "webtops/webOSs etc.")