

# XSLT - Introductory

Code: `xml-xslt`

## Author and version

- Daniel K. Schneider
- Email: [Daniel.Schneider@tecfa.unige.ch](mailto:Daniel.Schneider@tecfa.unige.ch)
- Version: 1.1 (modified 14/11/10 by DKS)

## Prerequisites

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

## Availability (including example files)

url: <http://tecfa.unige.ch/guides/te/files/xml-xslt.pdf>



## Objectives

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

## Warning

- XSLT is a rather complex 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	5
1.1 History and specifications	6
2. A first glance at XSLT	7
2.1 Root of an XSLT file stylesheet	7
2.2 Association of XML and an XSLT file	7
2.3 Basic XSLT	8
Example 2-1: Translation of a title tag into HTML centered H1	8
2.4 A complete XSLT example	9
Example 2-2:Hello XSLT	9
2.5 Rule execution order	12
2.6 The procedure recapitulated	13
3. Tuning output with xsl:output and CSS	14
3.1 Output declarations	14
Example 3-1:Output in HTML ISO-latin encoded	14
Example 3-2:Output in XHTML transitional with a namespace	15
Example 3-3:Your XML	16
Exemple 3-4:Output in SVG	16
3.2 CSS styling of HTHTML	17
Example 3-5:cooking	17
4. If things go wrong	18
4.1 Frequent problems and remediation	18
4.2 The XSLT default rule	19
5. Selective processing	20
5.1 Steering rule execution and information filtering	20
Example 5-1>Hello without content	20
5.2 A short glance at Xpath	21
5.3 Basic value extraction	23
Example 5-2:Value-of	23
5.4 Inserting a value inside a string	24

Example 5-3:Building a href tag with an email 24

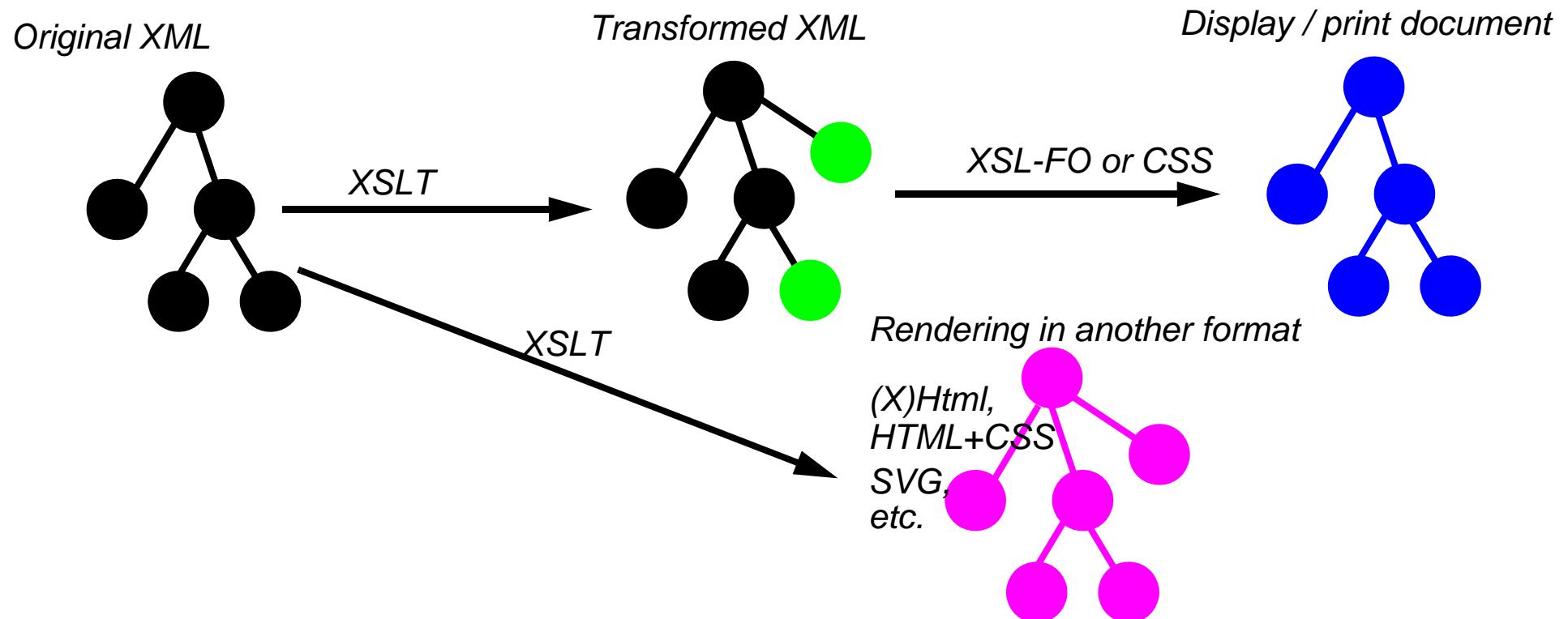
## 5.5 Example - dealing with pictures

25

Example 5-4:Dealing with references to pictures 25

# 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 anything you like, e.g.:
  - wellfomed HTML (closed tags)
  - any XML, e.g. yours or other XML languages like SVG, X3D
  - any other symbol format, e.g. RTF (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">           ← Selector of XML elements  
    .... contents to produce (i.e. HTML)      ← to transform  
    .... further instructions  
</xsl:template>
```

*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">           ← Rule applies to "title"  
    <h1 align="center">                 ← HTML code produced,  
        <xsl:apply-templates/>          ← i.e. the translation !  
    </h1>  
</xsl:template>
```

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

## 2.4 A complete XSLT example

### Example 2-2: Hello XSLT

url: <http://tecfa.unige.ch/guides/xml/examples/xsl-simple/hello.xml>

url: <http://tecfa.unige.ch/guides/xml/examples/xsl-simple/hello.xsl>

#### XML file (source)

```
<?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

- 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">
```

XSLT start

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

Rule dealing  
with the document's  
root element

Extract the value  
of the title element

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

An other rule

```
.....
</xsl:stylesheet>
```

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 root 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 (see next item)
3. `<xsl:apply-templates/>` means: "**go and look for other rules that can deal with each of the child elements of the current context**"
  - E.g. in our 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" ?>
<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)

```
<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. XSLT will look for rules that apply to each of the **current** child elements

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"  
            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>
```

## Example 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 HTMHTML

- Associating a CSS stylesheet with HTML output is trivial:
  - add a link tag in the "head" produced by the template for the root element
  - .... a link tag is just like any other tag that the translation will add

```
<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: <http://tecfa.unige.ch/guides/xml/examples/dtd-examples/cuisine.xml>

url: <http://tecfa.unige.ch/guides/xml/examples/dtd-examples/cuisine.xsl>

url: <http://tecfa.unige.ch/guides/xml/examples/dtd-examples/cuisine.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 XHTML 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
    - Output = to new file or to new document
  - You also may validate the output HTML with an HTML validator !

#### 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

```
<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

### Example 5-1: Hello without content

- The rule for the root element will only "call" the rules for the "title" and the "comment" element
- Information within a "content" tag would not be displayed, since we tell the processor to execute only two rules: one for "title" and another for "comment".

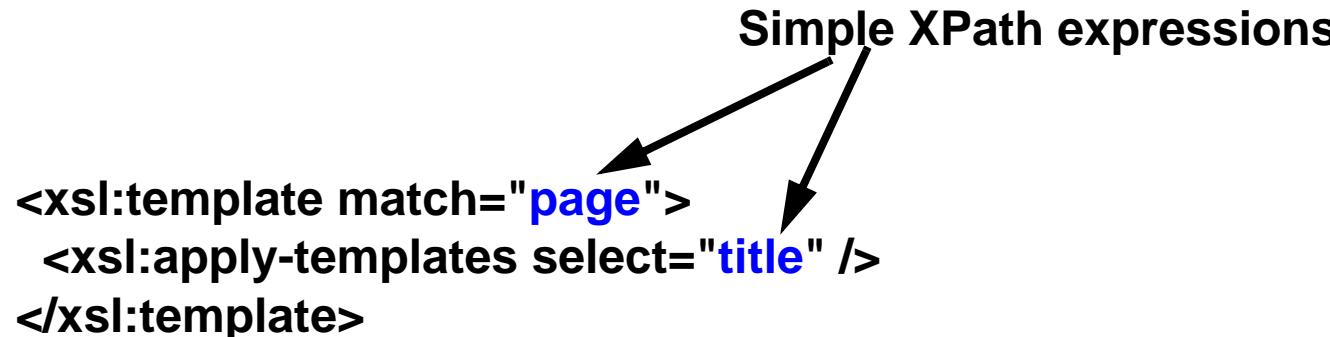
```
<xsl:template match="page">
  <html> <head> </head>
    <body bgcolor="#ffffff">
      <xsl:apply-templates select="title"/>
      <xsl:apply-templates select="comment"/>
    </body>
  </html>
</xsl:template>

<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

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



- 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
  - If you want to censor / change rule execution order, simply do it from the parent element like this:  
`<xsl:apply-templates select="title" />`
- 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"

## FYI: Examples of a few simple XPath expressions

- These should remind you of CSS selectors

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

## 5.3 Basic value extraction

### xsl:value-of

- inserts the value of an XPath expression and copies it to the output
- e.g. contents of an element or attribute values

#### 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

There is a special syntax:

Syntaxe: {....}

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

#### 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 Example - dealing with pictures

- There is **nothing special about pictures**, we just apply the principles we have seen so far.  
If your XML includes URLs or file names for pictures in one way or another, you need to translate these to an appropriate HTML tag.

### Example 5-4: Dealing with references to pictures

url: <http://tecfa.unige.ch/guides/xml/examples/xsl-simple/images.xml>

url: <http://tecfa.unige.ch/guides/xml/examples/xsl-simple/images.xsl>

The XML. As you can see we got a list of picture file names inserted in various ways

```
<?xml version="1.0"?>
<?xml-stylesheet href="images.xsl" type="text/xsl"?>
<page>
  <title>Picture list</title>
  <list>
    <image>dolores_001.jpg</image>
    <image>dolores_002.jpg</image>
    <image>dolores_002.jpg</image>
    <image2>scrolls.jpg </image2>
    <image2>scrolls.jpg </image2>
    <image3 source="dolores_002.jpg">Figure5: Good food</image3>
  </list>
  <comment>Written by DKS.</comment>
</page>
```

The XSLT stylesheet extracts these files names and creates appropriate HTML tags. "image2" pictures are not handled correctly.

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
Transform">
  <xsl:output method="html"
    doctype-public="-//W3C//DTD HTML 4.01 Transitional//EN"/>

  <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="list">
  The following three pictures were found within &lt;image&gt; tags:
  <xsl:apply-templates select="image"/>
  <p>
    The next picture was found within the first &lt;image2&gt; tag: </p>
  <p>  </p>
```

```
<p>The next image was not found by XSLT:</p>

<p>This last picture was found within an attribute of the "image3" tag:</p>
<xsl:apply-templates select="image3"/>
</xsl:template>

<xsl:template match="image">
  <p align="center">  </p>
</xsl:template>

<xsl:template match="image3">
  <p align="center">  </p>
  <p align="center"> <xsl:value-of select=".." /> </p>
</xsl:template>

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

</xsl:stylesheet>
```

