Chapter 8: XSLT
Extensible Stylesheet Language/Transformations

References:

- James Clark (Editor): XSL Transformations (XSLT), Version 1.0
  W3C Recommendation, 16 November 1999
  [https://www.w3.org/TR/xslt]
- Michael Kay (Editor): XSL Transformations (XSLT), Version 2.0
  W3C Recommendation, 23 January 2007
  [http://www.w3.org/TR/xslt20/]
- Michael Kay (Editor): XSL Transformations (XSLT), Version 3.0
  W3C Candidate Recommendation, 19 November 2015
  [http://www.w3.org/TR/xslt-30/]
- Michael Kay: XSLT 2.0 and XPath 2.0 Programmer's Reference (Programmer to Programmer) Wiley, 4th Ed. (June 3, 2008), ISBN-10: 0470192747, 1376 pages.
- Wikipedia (English): XSLT
  [https://en.wikipedia.org/wiki/XSLT]
- Robert Tolksdorf: Vorlesung XML-Technologien (Web Data and Interoperability), Kapitel 6: XSLT: Transformation von XML-Dokumenten.
  Freie Universität Berlin, AG Netzbasierte Informationssysteme, 2015.
  [http://blog.ag-nbi.de/wp-content/uploads/2015/05/06_XSLT.pdf]

Objectives

After completing this chapter, you should be able to:

- write transformations from XML to XML, or from XML to HTML as an XSLT stylesheet.

  This chapter also explains how a transformation from XML to LaTeX is done with XSLT.

- read and understand given XSLT stylesheets.

Overview

Introduction (1)

- XML is by itself only a data format:
  - ◇ It contains the data (content), but
  - ◇ does not specify how the elements should be printed or displayed in a browser or on paper.
- The output format is specified with style sheets:
  - ◇ Using Cascading Stylesheets (CSS).
  - ◇ Using XSLT to translate XML to HTML.

    The HTML is then typically formatted with CSS.
  - ◇ Using XSLT to translate XML to XSL-FO.

    For paper/PDF. One can also translate to LaTeX with XSLT.

Introduction (2)

- Many browsers support CSS, which is normally used for HTML web pages, also for XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css"
                 href="mystyle.css"?>
<GRADES-DB>
...
```

- However, this has many restrictions:
  - ◇ With CSS, the elements are formatted in the order in which they are written,
  - ◇ and there is only very limited filtering.

Introduction (3)

- The Extensible Stylesheet Language (XSL) consists of two parts:
  - ◇ XSLT (XSL Transformations) is a mechanism to transform XML documents into XSM documents (e.g., with other elements/tags).

    As explained below, the output is not necessarily XML. Even binary files can be generated.
  - ◇ XSL/FO (XSL Formatting Objects) is a set of element types/tags with a specified semantics for displaying them.

    "an XML vocabulary for specifying formatting semantics" [https://www.w3.org/Style/XSL/]

Introduction (4)

- So the idea is to
  - ◇ use XSLT to transform a custom XML file to XSL FO,
  - ◇ which is then displayed on screen or printed on paper.
- XSL-FO especially supports high-quality printout on paper (or as a PDF file).

  Thus, e.g. splitting a document into pages is important for XSL FO, whereas it is not important for displaying a web page in a browser. Also, hyphenation is treated. Where possible, properties from CSS2 where taken, and somtimes extended or split into several properties.

Introduction (5)

- XSL has its roots in DSSSL, the Document Style Semantics and Specification Language (for SGML).
- XSLT 1.0 became a W3C recommendation (official standard) on November 16, 1999.

  See [https://www.w3.org/TR/xslt]. The current version is XSLT 2.0 from Januar 23, 2007. [https://www.w3.org/TR/xslt20/].
- XSL 1.0 (which specifies XSL FO) became a W3C recommendation on October 15, 2001.

  See [https://www.w3.org/TR/2001/REC-xsl-20011015/]
  Current ver.: XSL 1.1 (Dec. 5, 2006) [https://www.w3.org/TR/xsl/]
  Draft: XSL 2.0 (Jan. 17, 2012) [https://www.w3.org/TR/xslfo20/]

Introduction (6)

- Quite often, XSLT is used without XSL-FO:
  - ◇ For instance, XML is transformed to HTML to be displayed in a browser.
  - ◇ Or XSLT is used to transform a given XML document into a differently structured XML document (with different element types/tags).

    In this way, one can adapt an XML file from a business partner to one's own XML structure. Or one can integrate XML files from different sources to a common XML vocabulary.

Introduction (7)

- For translating XML to HTML, XSLT can be used in two places:
  - ◇ Client: the web browser does the mapping,
  - ◇ Server: one uses an XSLT processor to translate XML to HTML, and puts the HTML files on the web server.

    Maybe in addition to the XML files. It is also possible that the HTTP server does the translation on demand: The web browser sends in the HTTP request a list of mime types it understands.
- It seems that browsers today still understand only XSLT 1.0 (which is based on XPath 1.0).

Introduction (8)

- Doing the XML to HTML mapping on Client or Server, continued:
  - ◇ If one does the translation in an intranet only for the employees of the company, one can at least rely on the knowledge which browser is used.
  - ◇ On the global internet, it might be that potential customers use old browsers which do not support XSLT or support it in incompatible ways.

    One can still put the XML file on the server in addition to the HTML file, in order to support semantic web applications (like price comparision services).

XSLT Implementations

- Saxon (from Michael Kay)

  M. Kay is editor of the XSLT 2.0 Spec. Basic version (without static type checking and XQuery→Java compiler) is open source. Supports XSLT 2.0, XPath 2.0, XQuery 1.0. [http://saxon.sourceforge.net/]
- Xalan (Apache) (Java and C++ versions)

  [http://xalan.apache.org/]
- XT (James Clark)

  [http://www.blnz.com/xt/index.html], [http://www.jclark.com]
- Sablotron

  [http://www.gingerall.org/sablotron.html]

Overview

Example XML File (1)

- Consider the grades DB with data in attributes:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl"
                 href="mystyle.xsl"?>
<GRADES-DB>
    <STUDENT SID='101'
             FIRST='Ann' LAST='Smith'
             EMAIL='smith@acm.org'/>
    <STUDENT SID='102'
             FIRST='Michael' LAST='Jones'/>
    ...
```


Example XML File (2)

- Grades DB (with data in attributes), continued:

```
    <EXERCISE CAT='H' ENO='1'
              TOPIC='Relational Algebra'
              MAXPT='10'/>
    ...
    <RESULT SID='101' CAT='H' ENO='1'
            POINTS='10'/>
    ...
</GRADES-DB>
```

- Note: The output of Firefox 43, Internet Explorer 11 and Microsoft Edge is simply empty if there is a typing error in the name of the style sheet.

Example XSLT Stylesheet (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xmlns="http://www.w3.org/1999/xhtml"
    exclude-result-prefixes="html">
```

- XSLT stylesheets are written in XML syntax, using the outermost element `stylesheet`.

`transform` is allowed as a synonym. The version number is mandatory.

Example XSLT Stylesheet (2)

- The namespace URI for XSLT elements must be

  `http://www.w3.org/1999/XSL/Transform`.

- In the example, a namespace for XHTML is declared in addition to the namespace for XSLT, and this is also the default namespace.

  So one can write XHTML tags without namespace prefix.

- With `exclude-result-prefixes`, it is specified that in the output of the transformation, the namespace prefix of XHTML tags should not be written.

Example XSLT Stylesheet (3)

```
<xsl:output
    method="xml"
    doctype-system=
      "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"
    doctype-public="-//W3C//DTD XHTML 1.1//EN" />
```

- This specifies how the resulting XDM tree should be printed/serialized (in this case, as XHTML).

  Alternative (classical HTML):

```
<xsl:output method="html"
    encoding="ISO-8859-1"
    doctype-public="-//W3C//DTD HTML 3.2 Final//EN"
    indent="yes" />
See: [https://www.w3.org/TR/xslt#output]
```


Example XSLT Stylesheet (4)

```
<xsl:template match="/">
    <html>
        <head><title>Students</title></head>
        <body>
            <h1>Student List</h1>
            <ul>
                <xsl:apply-templates
                     select="/GRADES-DB/STUDENT"/>
            </ul>
        </body>
    </html>
</xsl:template>
```


Example XSLT Stylesheet (5)

- A simple XSLT stylesheet is mainly a set of “templates” (transformation rules”).

  [https://www.w3.org/TR/xslt#rules]

- Each template describes a transformation from a subtree of the input (i.e. a start node and all its descendants) into a tree or list of trees.

- The output of the transformation for a given XML document is given by the rule for the root node “/” of the input tree.

  All other templates are used only if they are called (maybe indirectly) from the pattern for this root node “/” with “`apply-templates`”.

Example XSLT Stylesheet (6)

- Each transformation rule (template) consists mainly of two parts:
  - ◇ The attribute "`match`" defines, for which nodes this transformation rule is applicable.

    It is an XPath-expression.
  - ◇ The contents of "`xsl:template`" is a pattern for the output. It is mainly copied to the output tree, but contained XSLT elements are evaluated.

    In the example, the contents contains "`xsl:apply-templates`". Another typical tag used in the contents is "`xsl:value-of`".

Example XSLT Stylesheet (7)

- "`xsl:apply-templates`" will be replaced by the result of applying the transformation recursively to the node which is specified in the "`select`"-attribute.

  The contents of this attribute is an XPath expression. [https://www.w3.org/TR/xslt#section-Applying-Template-Rules]
- If it selects several nodes, the transformation results for all these the nodes are inserted into the output tree (in the same sequence).
- If the attribute "`select`" is omitted, all child nodes are transformed.

Example XSLT Stylesheet (8)

```
    <xsl:template match="STUDENT">
        <li>
            <xsl:value-of select="@LAST" />,
            <xsl:value-of select="@FIRST" />
        </li>
    </xsl:template>

</xsl:stylesheet>
```

- The result of the stylesheet is an HTML page which contains the student names, e.g. "Smith, Ann" as an unordered list.

Example XSLT Stylesheet (9)

- "`xsl:value-of`" is replaced by the value of the node which is selected by the XPath-expression in the "`select`"-attribute.

  [https://www.w3.org/TR/xslt#value-of]
- If several items are selected, only the first is chosen.

  I.e. the node, for which the template is applied, is the context node, and then the first result (in document order) of the XPath-expression specified under `select` is taken.
- This is converted to a string.

Semantics of match-Attribute

- A template with XPath-expression p in the attribute "match" is applicable to a node n
  - ◇ if there is some ancestor a of n
  - ◇ such that n is an element of the nodes selected by p evaluated with context node a.
- For instance, "/GRADES-DB/STUDENT/" matches
  - ◇ a STUDENT-node within the top GRADES-DB node,
  - ◇ not a GRADES-DB node with a STUDENT child node.
- There are priority rules if several templates match.

  [https://www.w3.org/TR/xslt#conflict]

Stylesheets are XML (1)

- Note that XSLT stylesheets must be well-formed XML. Thus, even if HTML is generated, one must e.g. write "
" and not simply "
".
- XML has only the five predefined entities "<", ">", "'", "&quote;", "&".
- To use other HTML entities (e.g. " "):
  - ◇ declare them in a local DTD part in the DOCTYPE declaration (see below), or
  - ◇ put them into a CDATA section (see below), or
  - ◇ write a character reference: for .

Stylesheets are XML (2)

- Solution with CDATA-Section:

```
<xsl:text disable-output-escaping="yes"
        ><![CDATA[ ]]></xsl:text>
```

- The xsl:text is needed so that the CDATA-section does not appear in the output (because the output escaping is disabled).

  xsl:text simply creates a text node.
  [https://www.w3.org/TR/xslt#section-Creating-Text]
  [https://www.w3.org/TR/xslt#disable-output-escaping]

- Obviously, this is only practical if there are very few entity references.

Stylesheets are XML (3)

```
<!DOCTYPE xsl:stylesheet [
    <!ENTITY Auml  "&#x00C4;">
    <!ENTITY auml  "&#x00E4;">
    <!ENTITY Ouml  "&#x00D6;">
    <!ENTITY ouml  "&#x00F6;">
    <!ENTITY Uuml  "&#x00DC;">
    <!ENTITY uuml  "&#x00FC;">
    <!ENTITY szlig "&#x00DF;">
    <!ENTITY nbsp  " ">
]>
```

The numbers can be taken from the HTML DTD or the Unicode standard or [http://www.w3.org/2003/entities/2007/w3centities-f.ent].