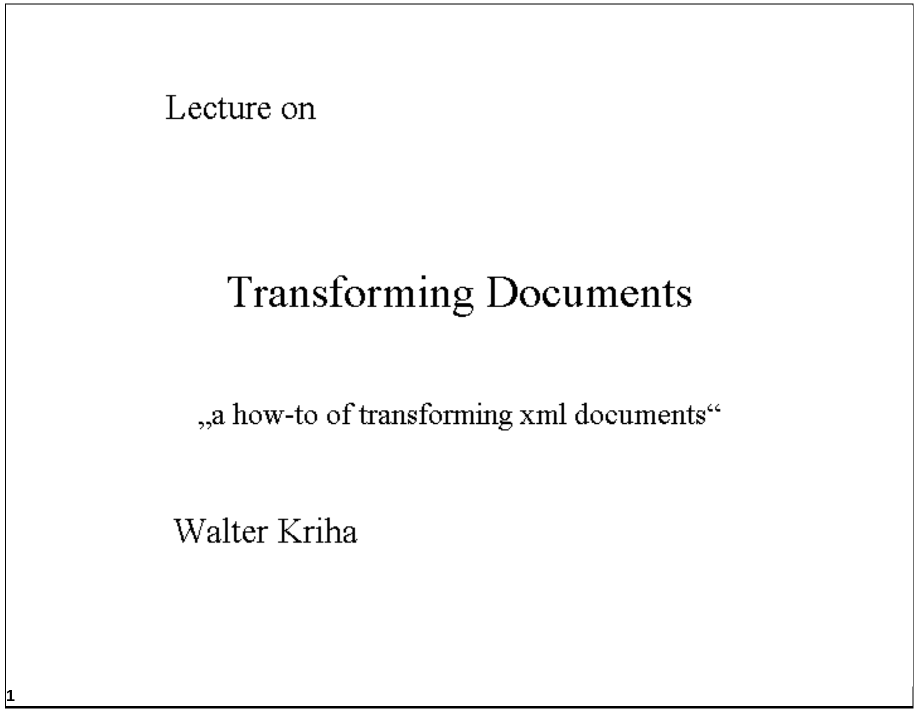
Transforming XML documents



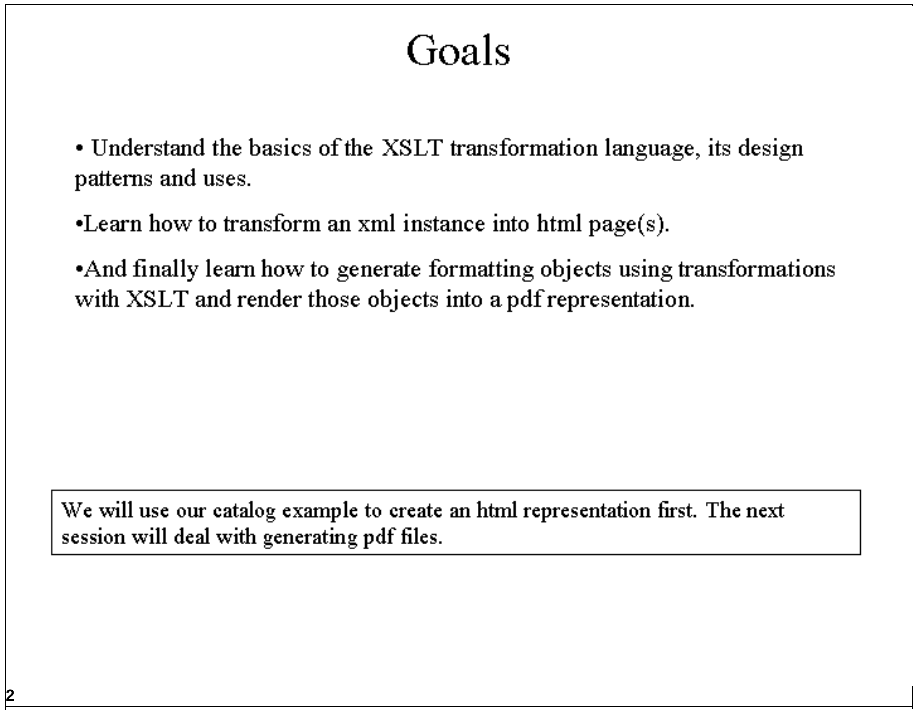
Lecture on

Transforming Documents

„a how-to of transforming xml documents“

Walter Kriha

1



Goals

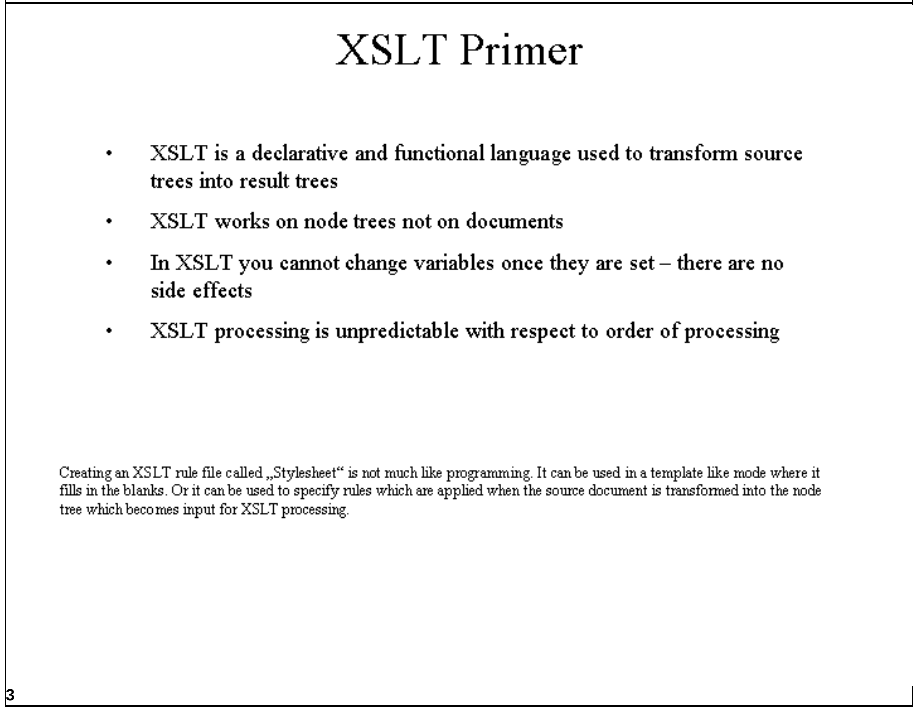
• Understand the basics of the XSLT transformation language, its design patterns and uses.

•Learn how to transform an xml instance into html page(s).

•And finally learn how to generate formatting objects using transformations with XSLT and render those objects into a pdf representation.

We will use our catalog example to create an html representation first. The next session will deal with generating pdf files.

2



XSLT Primer

- XSLT is a declarative and functional language used to transform source trees into result trees
- XSLT works on node trees not on documents
- In XSLT you cannot change variables once they are set – there are no side effects
- XSLT processing is unpredictable with respect to order of processing

Creating an XSLT rule file called „Stylesheet“ is not much like programming. It can be used in a template like mode where it fills in the blanks. Or it can be used to specify rules which are applied when the source document is transformed into the node tree which becomes input for XSLT processing.

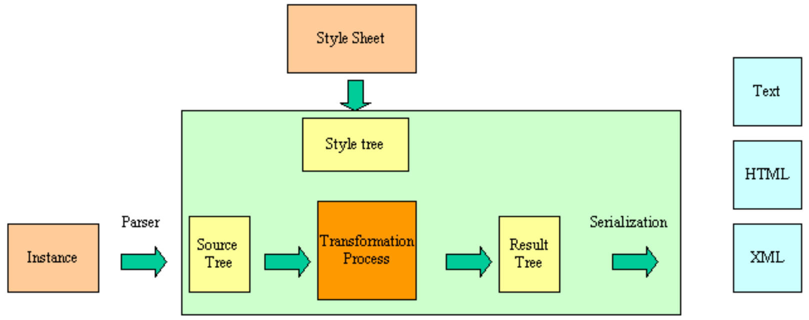
3

The structure of an XSLT Stylesheet

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:doc="http://nwalsh.com/xsl/documentation/1.0"
                exclude-result-prefixes="doc„ version='1.0'>
<xsl:output method="html„ encoding="ISO-8859-1„ indent="no"/>
<xsl:template match="catalog"> (The template rule)
          <html><head> <title>Catalog</title> </head> (literal result elements)
          <body> <div align="center"><h1>Catalog</h1></div>
          <div align="center"><h2><xsl:value-of select="./@ablaufdatum"/></h2></div>
          <xsl:apply-templates select="tableofcontent"/> (xsl instructions generating nodes)
          <xsl:apply-templates select="articles"/>
          <xsl:apply-templates select="orderform"/>
          <xsl:apply-templates select="index"/>
          </body> </html>
</xsl:template>
</xsl:stylesheet>
```

A xsl template matches nodes from the document input tree and performs operations on those nodes. Literal elements are just copied over into the result node tree.

How an XSLT processor works



The XSLT processer lets an XML parser transform the source document into the source tree consisting of nodes. It reads the stylesheet (rule) file and applies its rules to the source node tree, thereby transforming it into a result node tree. This tree is then serialized in whatever output format is requested. This processing can be changed because the result can be just another xml document, ready for input to a new XSLT transformation process with a different style sheet. This is not unlike SQL works with database tables as Michael Kay notes from whom this diagram comes. (see Resources)

How to use an XSLT processor

```
java -classpath ".;D:/xmltools/saxon.jar;" com.icl.saxon.StyleSheet -o catalog.html
catalog.xml catalog.xsl
```

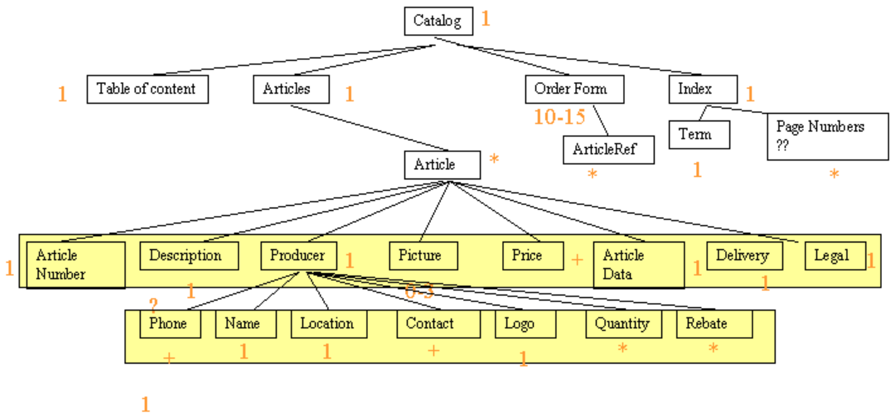
I am using the saxon xslt processor, a pure java implementation by Michael Kay. Alternatives are the Xalan processor from apache.org. Internet Explorer can also perform xslt transformation directly. Here –o catalog.html directs the processor to create an output file with the name catalog.html and to use catalog.xml as the input document and catalog.xsl as the stylesheet containing the transformation rules.

Important XSLT directives

- <xsl:template match=„somePattern“> Defines a template rule which will be processed when the processor finds a node that matches „somePattern“ in the source node tree.
- <xsl:apply-templates>
- <xsl:apply-templates select=„somePattern“> Selectively (if select is defined) continues processing of child nodes of the current node.
- <xsl:value-of select=„somePattern“> copy the text from the source nodes matching somePattern over into the result tree.
- XPath expressions to address nodes: ./article/@articleNumber selects an articleNumber attribute from a child node with name „article“ of the current node.

Some of the expressions can become quite complicated, depending on the structure of the source document, the target (result structure) and the amount of transformation necessary. It makes a difference if e.g. the result tree is very close in structure to the source tree.

Step one: understand how your xml instance is structured



We seem to have two blocks in our instance which look a little like they could be represented as tables: Article and Producer.

Step two: define your result tree

Article Header

Article Description

main article data in table form

Pictures and technical data

delivery information

Legal statements

complete producer information in table form

You will notice that the target structure differs from the source structure in several places. Where there is NO difference we will use the so called rule-based processing with XSLT. If the order changes, we will use the „fill-in-the-blanks“ pattern to selectively pull in content from the source tree into our result tree. And just in case the result tree needs a lot of processing we will use the „navigational pattern“ which uses subroutine calls between template rules and is closest to regular programming. Terms after M.Kay who also mentions another pattern: computational stylesheets.

Step three: create stylesheet template

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform version='1.0'>
<xsl:output method="html„ encoding="ISO-8859-1„ indent="no"/>
<!-- ================================================================== -->
<xsl:template match="*">
  <xsl:message>
    <xsl:text>No template matches </xsl:text>
    <xsl:value-of select="name(.)"/>
    <xsl:text>.</xsl:text>
  </xsl:message>
  <font color="red"> <xsl:text>&lt;</xsl:text>
    <xsl:value-of select="name(.)"/> <xsl:text>&gt;</xsl:text>
    <xsl:apply-templates/>
    <xsl:text>&lt;/</xsl:text> <xsl:value-of select="name(.)"/><xsl:text>&gt;</xsl:text> </font>
</xsl:template>
          <xsl:template match="text()"> <xsl:value-of select="."/>
</xsl:template>
<!-- ================================================================== -->
```

The header is standard but the rest is for debugging ONLY! It will show elements with no matching rule in red in the result tree so you know what you have missed yet.

Step four: apply simple rules

```
<xsl:template match="orderform">
          <div align="left"><h2>Order Form </h2></div>
            <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="index">
                    <div align="left"><h2>Index </h2></div>
            <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="description">
          <p> <xsl:value-of select="."/> </p>
</xsl:template>
<xsl:template match="logo">
          <p> <img src="{.}"></img> </p>
</xsl:template>
```

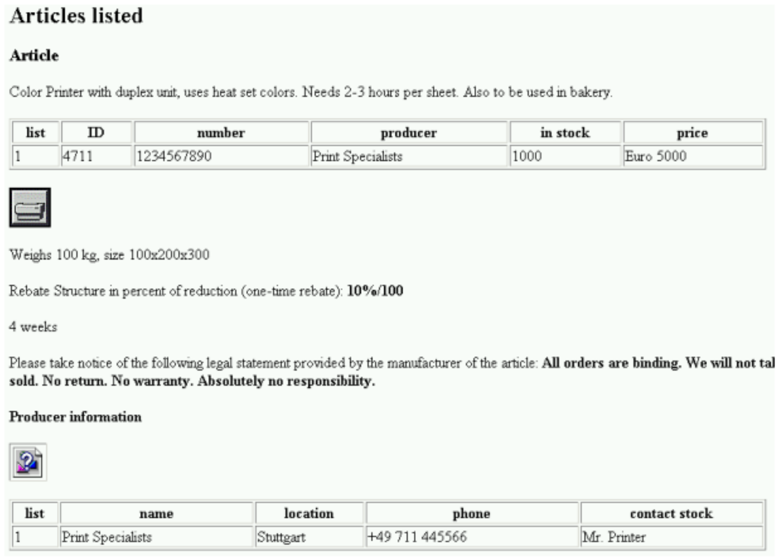
For those elements which we want to copy over from the source tree into the result tree we just create xsl:template match= ... comands, surrounded by html literals e.g. to create a header or an image link.

Step five: fill in the blanks

```
<xsl:template match="producer">
                    <div align="left"><h4>Producer information</h4></div>
<xsl:apply-templates select="logo"/>
<table border="1" width="800">
<tr> <th>list</th> <th>name</th> <th>location</th> <th>phone</th> <th>contact stock</th> </tr>
          <tr>
          <td><xsl:number/></td>
          <td><xsl:value-of select="name"/></td>
          <td><xsl:value-of select="location"/></td>
          <td><xsl:value-of select="phone"/></td>
          <td><xsl:value-of select="contact"/></td>
          </tr>
           </table>
</xsl:template>
```

The column values for our producer table are pulled in with xsl:value-of and a pattern to select. This makes our stylesheet independent of source document changes (e.g. order of producer child elements)

Step six: run the XSLT processor



Articles listed

Article

Color Printer with duplex unit, uses heat set colors. Needs 2-3 hours per sheet. Also to be used in bakery.

list	ID	number	producer	in stock	price
1	4711	1234567890	Print Specialists	1000	Euro 5000

Weighs 100 kg, size 100x200x300

Rebate Structure in percent of reduction (one-time rebate): **10%/100**

4 weeks

Please take notice of the following legal statement provided by the manufacturer of the article: **All orders are binding. We will not tal**
sold. No return. No warranty. Absolutely no responsibility.

Producer information

list	name	location	phone	contact stock
1	Print Specialists	Stuttgart	+49 711 445566	Mr. Printer

Resources (1)

- Michael Kay, XSLT 2nd Edition, Programmers Reference
- Charles Goldfarb, XML Handbook 4th edition: read about XSLT.