An Introduction to XML and Web Technologies

Transforming XML Documents with XSLT

Anders Møller & Michael I. Schwartzbach

Objectives

- How XML documents may be rendered in browsers
- How the XSLT language transforms XML documents
- How XPath is used in XSLT

Presenting a Business Card

```
<card xmlns="http://businesscard.org">
  <name>John Doe</name>
  <title>CEO, Widget Inc.</title>
  <email>john.doe@widget.inc</email>
  <phone>(202) 555-1414</phone>
  <logo uri="widget.gif"/>
</card>
```

Using CSS

```
card  { background-color: #cccccc; border: none; width: 300;}
name  { display: block; font-size: 20pt; margin-left: 0; }
title { display: block; margin-left: 20pt;}
email { display: block; font-family: monospace; margin-left: 20pt;}
phone { display: block; margin-left: 20pt;}
```

- the information cannot be rearranged
- information encoded in attributes cannot be exploited
- additional structure cannot be introduced

Using XSLT

```
<?xml-stylesheet type="text/xsl" href="businesscard.xsl"?>
<card xmlns="http://businesscard.org">
  <name>John Doe</name>
  <title>CEO, Widget Inc.</title>
  <email>john.doe@widget.inc</email>
  <phone>(202) 555-1414</phone>
  <logo uri="widget.gif"/>
</card>
```

XSLT for Business Cards (1/2)

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:b="http://businesscard.org"
     xmlns="http://www.w3.org/1999/xhtml">

  <xsl:template match="b:card">
    <html>
      <head>
        <title><xsl:value-of select="b:name/text()"/></title>
      </head>
      <body bgcolor="#ffffff">
        <table border="3">
          <tr>
            <td>
              <xsl:apply-templates select="b:name"/><br/>
              <xsl:apply-templates select="b:title"/><p/>
              <tt><xsl:apply-templates select="b:email"/></tt><br/>
```

XSLT for Business Cards (2/2)

```
            <xsl:if test="b:phone">
              Phone: <xsl:apply-templates select="b:phone"/><br/>
            </xsl:if>
          </td>
          <td>
            <xsl:if test="b:logo">
              <img src="{b:logo/@uri}"/>
            </xsl:if>
          </td>
        </tr>
      </table>
    </body>
  </html>
</xsl:template>

<xsl:template match="b:name|b:title|b:email|b:phone">
  <xsl:value-of select="text()"/>
</xsl:template>

</xsl:stylesheet>
```

XSL-FO

- XSLT was originally design to target XSL-FO
- XSL-FO (Formatting Objects) in an XML language for describing physical layout of texts
- Widely used in the graphics industry
- Not supported by any browsers yet

XSL-FO for Business Cards

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:b="http://businesscard.org"
     xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="b:card">
    <fo:root>
      <fo:layout-master-set>
        <fo:simple-page-master master-name="simple"
                               page-height="5.5cm"
                               page-width="8.6cm"
                               margin-top="0.4cm"
                               margin-bottom="0.4cm"
                               margin-left="0.4cm"
                               margin-right="0.4cm">
          <fo:region-body/>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="simple">
        <fo:flow flow-name="xsl-region-body">
          <fo:table>
            <fo:table-column column-width="5cm"/>
            <fo:table-column column-width="0.3cm"/>
            <fo:table-column column-width="2.5cm"/>
            <fo:table-body>
              <fo:table-row>
                <fo:table-cell>
                  <fo:block font-size="18pt"
                            font-family="sans-serif"
                            line-height="20pt"
                            background-color="#A0D0FF"
                            padding-top="3pt">
                    <xsl:value-of select="b:name"/>
                  </fo:block>
                </fo:table-cell>
```

```
                <fo:table-cell/>
                <fo:table-cell>
                  <xsl:if test="b:logo">
                    <fo:block>
                      <fo:external-graphic src="url({b:logo/@uri})"
                                           content-width="2.5cm"/>
                    </fo:block>
                  </xsl:if>
                </fo:table-cell>
              </fo:table-row>
            </fo:table-body>
          </fo:table>
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
</xsl:stylesheet>
```

Overview

- Introduction
- **Templates and pattern matching**
- Sequence constructors
- Using XSLT

XSLT Stylesheets

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="2.0">
  ...
</xsl:stylesheet>
```

- XSLT is a domain-specific language for writing XML transformations (compare with e.g. JDOM)

- An XSLT stylesheet contains *template rules*
- Processing starts at the root node of the input document

Template Rules

```
<xsl:template match="...">
   ...
</xsl:template>
```

- Find the template rules that *match* the context node
- Select the *most specific* one
- *Evaluate* the body (a *sequence constructor*)

Use of XPath in XSLT

- Specifying *patterns* for template rules
- Selecting *nodes* for processing
- Computing *boolean* conditions
- Generating *text* contents for the output document

Evaluation Context

- A *context item* (a node in the source tree or an atomic value)
- A context *position* and *size*
- A set of *variable bindings* (mapping variable names to values)
- A *function library* (including those from XPath)
- A set of *namespace declarations*

The Initial Context

- The context item is the document root
- The context position and size both have value 1
- The set of variable bindings contains only global parameters
- The function library is the default one
- The namespace declarations are those defined in the root element of the stylesheet

Patterns and Matching

- A *pattern* is a restricted XPath expression
  - it is a union of path expressions
  - each path expression contains a number of steps separated by / or //
  - each step may only use the `child` or `attribute` axis
- A pattern *matches* a node if
  - starting from *some* node in the tree:
  - the given node is *contained* in the resulting sequence

```
rcp:recipe/rcp:ingredient//rcp:preparation
```

Names, Modes, Priorities

- Templates may have other attributes

- `name`: used to call templates like function
- `mode`: used to restrict the candidate templates
- `priority`: used to determine specificity

Overview

- Introduction
- Templates and pattern matching
- **Sequence constructors**
- Using XSLT

Sequence Constructors

- Element and attribute constructors
- Text constructors
- Copying nodes
- Recursive application
- Repetitions
- Conditionals
- Template invocation
- …

Literal Constructors

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
     <html>
       <head>
         <title>Hello World</title>
       </head>
       <body bgcolor="green">
         <b>Hello World</b>
       </body>
     </html>
  </xsl:template>
</xsl:stylesheet>
```

Explicit Constructors

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
     <xsl:element name="html">
        <xsl:element name="head">
          <xsl:element name="title">
            Hello World
          </xsl:element>
        </xsl:element>
        <xsl:element name="body">
          <xsl:attribute name="bgcolor" select="'green'"/>
          <xsl:element name="b">
            Hello World
          </xsl:element>
        </xsl:element>
     </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Computed Attributes Values (1/2)

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
     <xsl:element name="html">
        <xsl:element name="head">
          <xsl:element name="title">
            Hello World
          </xsl:element>
        </xsl:element>
        <xsl:element name="body">
          <xsl:attribute name="bgcolor" select="//@bgcolor"/>
          <xsl:element name="b">
            Hello World
          </xsl:element>
        </xsl:element>
     </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Computed Attribute Values (2/2)

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns="http://www.w3.org/1999/xhtml">
  <xsl:template match="/">
     <html>
       <head>
         <title>Hello World</title>
       </head>
       <body bgcolor="{//@bgcolor}">
         <b>Hello World</b>
       </body>
     </html>
  </xsl:template>
</xsl:stylesheet>
```

Text Constructors

- Literal text becomes character data in the output

```
here is some chardata
```

- Whitespace control requires a constructor:

```
<xsl:text>    </xsl:text>
```

- The (atomized) value of an XPath expression:

```
<xsl:value-of select=".//@unit"/>
```

Recursive Application

- The `apply-templates` element
  - finds some nodes using the `select` attribute
  - applies the entire stylesheet to those nodes
  - concatenates the resulting sequences
- The default `select` value is `child::node()`

- Processing is often (but not necessarily!) a simple recursive traversal down the input XML tree

Student Data

```
<students>
   <student id="100026">
      <name">Joe Average</name>
      <age>21</age>
      <major>Biology</major>
      <results>
        <result course="Math 101" grade="C-"/>
        <result course="Biology 101" grade="C+"/>
        <result course="Statistics 101" grade="D"/>
      </results>
   </student>
   <student id="100078">
      <name>Jack Doe</name>
      <age>18</age>
      <major>Physics</major>
      <major>XML Science</major>
      <results>
        <result course="Math 101" grade="A"/>
        <result course="XML 101" grade="A-"/>
        <result course="Physics 101" grade="B+"/>
        <result course="XML 102" grade="A"/>
      </results>
   </student>
</students>
```

```
<summary>
  <grades id="100026">
    <grade>C-</grade>
    <grade>C+</grade>
    <grade>D</grade>
  </grades>
  <grades id="100078">
    <grade>A</grade>
    <grade>A-</grade>
    <grade>B+</grade>
    <grade>A</grade>
  </grades>
</summary>
```

Generating Students Summaries

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="students">
    <summary>
      <xsl:apply-templates select="student"/>
    </summary>
  </xsl:template>

  <xsl:template match="student">
    <grades>
      <xsl:attribute name="id" select="@id"/>
      <xsl:apply-templates select=".//@grade"/>
    </grades>
  </xsl:template>

  <xsl:template match="@grade">
    <grade>
      <xsl:value-of select="."/>
    </grade>
  </xsl:template>
</xsl:stylesheet>
```

Using Modes, Desired Output

```
<summary>
  <name id="100026">Joe Average</name>
  <name id="100078">Jack Doe</name>
  <grades id="100026">
    <grade>C-</grade>
    <grade>C+</grade>
    <grade>D</grade>
  </grades>
  <grades id="100078">
    <grade>A</grade>
    <grade>A-</grade>
    <grade>B+</grade>
    <grade>A</grade>
  </grades>
</summary>
```

Using Modes (1/2)

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="students">
    <summary>
      <xsl:apply-templates mode="names" select="student"/>
      <xsl:apply-templates mode="grades" select="student"/>
    </summary>
  </xsl:template>

  <xsl:template mode="names" match="student">
    <name>
      <xsl:attribute name="id" select="@id"/>
      <xsl:value-of select="name"/>
    </name>
  </xsl:template>
```

Using Modes (2/2)

```
<xsl:template mode="grades" match="student">
    <grades>
      <xsl:attribute name="id" select="@id"/>
      <xsl:apply-templates select=".//@grade"/>
    </grades>
  </xsl:template>

  <xsl:template match="@grade">
    <grade>
      <xsl:value-of select="."/>
    </grade>
  </xsl:template>
</xsl:stylesheet>
```

Repetitions

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="students">
    <summary>
      <xsl:apply-templates select="student"/>
    </summary>
  </xsl:template>

  <xsl:template match="student">
    <grades>
      <xsl:attribute name="id" select="@id"/>
      <xsl:for-each select=".//@grade">
        <grade>
          <xsl:value-of select="."/>
        </grade>
      </xsl:for-each>
    </grades>
  </xsl:template>
</xsl:stylesheet>
```

Conditionals (if)

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="students">
    <summary>
      <xsl:apply-templates select="student"/>
    </summary>
  </xsl:template>

  <xsl:template match="student">
    <grades>
      <xsl:attribute name="id" select="@id"/>
      <xsl:for-each select=".//@grade">
        <xsl:if test=". ne 'F'">
          <grade><xsl:value-of select="."/></grade>
        </xsl:if>
      </xsl:for-each>
    </grades>
  </xsl:template>
</xsl:stylesheet>
```

Conditionals (choose)

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
     xmlns:b="http://businesscard.org"
  <xsl:template match="b:card">
     <contact>
       <xsl:choose>
         <xsl:when test="b:email">
           <xsl:value-of select="b:email"/>
         </xsl:when>
         <xsl:when test="b:phone">
           <xsl:value-of select="b:phone"/>
         </xsl:when>
         <xsl:otherwise>
           No information available
         </xsl:otherwise>
       </xsl:choose>
     </contact>
  </xsl:template>
</xsl:stylesheet>
```

Template Invocation (1/2)

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="students">
    <summary>
      <xsl:apply-templates select="student"/>
    </summary>
  </xsl:template>

  <xsl:template match="student">
    <grades>
      <xsl:attribute name="id" select="@id"/>
      <xsl:for-each select=".//@grade">
        <xsl:call-template name="listgrade"/>
      </xsl:for-each>
    </grades>
  </xsl:template>
```

Template Invocation (2/2)

```
  <xsl:template name="listgrade">
    <grade>
      <xsl:value-of select="."/>
    </grade>
  </xsl:template>
</xsl:stylesheet>
```

Built-In Template Rules

- What happens if no template matches a node?
- XSLT applies a *default* template rule
  - text is copied to the output
  - nodes apply the stylesheet recursively to the children

- A widely used default rule:
  for the document root node

Sorting

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="students">
    <enrolled>
      <xsl:apply-templates select="student">
        <xsl:sort select="age" data-type="number"
                                order="descending"/>
        <xsl:sort select="name"/>
      </xsl:apply-templates>
    </enrolled>
  </xsl:template>

  <xsl:template match="student">
    <student name="{name}" age="{age}"/>
  </xsl:template>
</xsl:stylesheet>
```

Copying Nodes

- The `copy-of` element creates *deep* copies
- The `copy` element creates *shallow* copies

- Give top-most HTML lists square bullets:

```
<xsl:template match="ol|ul">
   <xsl:copy>
      <xsl:attribute name="style"
                     select="'list-style-type: square;'"/>
      <xsl:copy-of select="*"/>
   </xsl:copy>
</xsl:template>
```

An Identity Transformation

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/|@*|node()">
    <xsl:copy>
      <xsl:apply-templates select="@*|node()"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Overview

- Introduction
- Templates and pattern matching
- Sequence constructors
- **Using XSLT**

XSLT 1.0 Restrictions

- Most browsers only support XSLT 1.0
- Can only use XPath 1.0
- No sequence values, only *result tree fragments*
- *…*

XSLT for Recipes (1/6)

```
<xsl:stylesheet version="2.0"
     xmlns="http://www.w3.org/1999/xhtml"
     xmlns:rcp="http://www.brics.dk/ixwt/recipes"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="rcp:collection">
    <html>
      <head>
        <title><xsl:value-of select="rcp:description"/></title>
        <link href="style.css" rel="stylesheet" type="text/css"/>
      </head>
      <body>
        <table border="1">
          <xsl:apply-templates select="rcp:recipe"/>
        </table>
      </body>
    </html>
  </xsl:template>
```

XSLT for Recipes (2/6)

```
<xsl:template match="rcp:recipe">
   <tr>
     <td>
       <h1><xsl:value-of select="rcp:title"/></h1>
       <i><xsl:value-of select="rcp:date"/></i>
       <ul><xsl:apply-templates select="rcp:ingredient"/></ul>
       <xsl:apply-templates select="rcp:preparation"/>
       <xsl:apply-templates select="rcp:comment"/>
       <xsl:apply-templates select="rcp:nutrition"/>
     </td>
   </tr>
 </xsl:template>
```

XSLT for Recipes (3/6)

```
<xsl:template match="rcp:ingredient">
   <xsl:choose>
     <xsl:when test="@amount">
       <li>
         <xsl:if test="@amount!='*'">
           <xsl:value-of select="@amount"/>
           <xsl:text> </xsl:text>
           <xsl:if test="@unit">
             <xsl:value-of select="@unit"/>
             <xsl:if test="number(@amount)>number(1)">
               <xsl:text>s</xsl:text>
             </xsl:if>
             <xsl:text> of </xsl:text>
           </xsl:if>
         </xsl:if>
         <xsl:text> </xsl:text>
         <xsl:value-of select="@name"/>
       </li>
     </xsl:when>
```

XSLT for Recipes (4/6)

```
    <xsl:otherwise>
      <li><xsl:value-of select="@name"/></li>
      <ul><xsl:apply-templates select="rcp:ingredient"/></ul>
      <xsl:apply-templates select="rcp:preparation"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

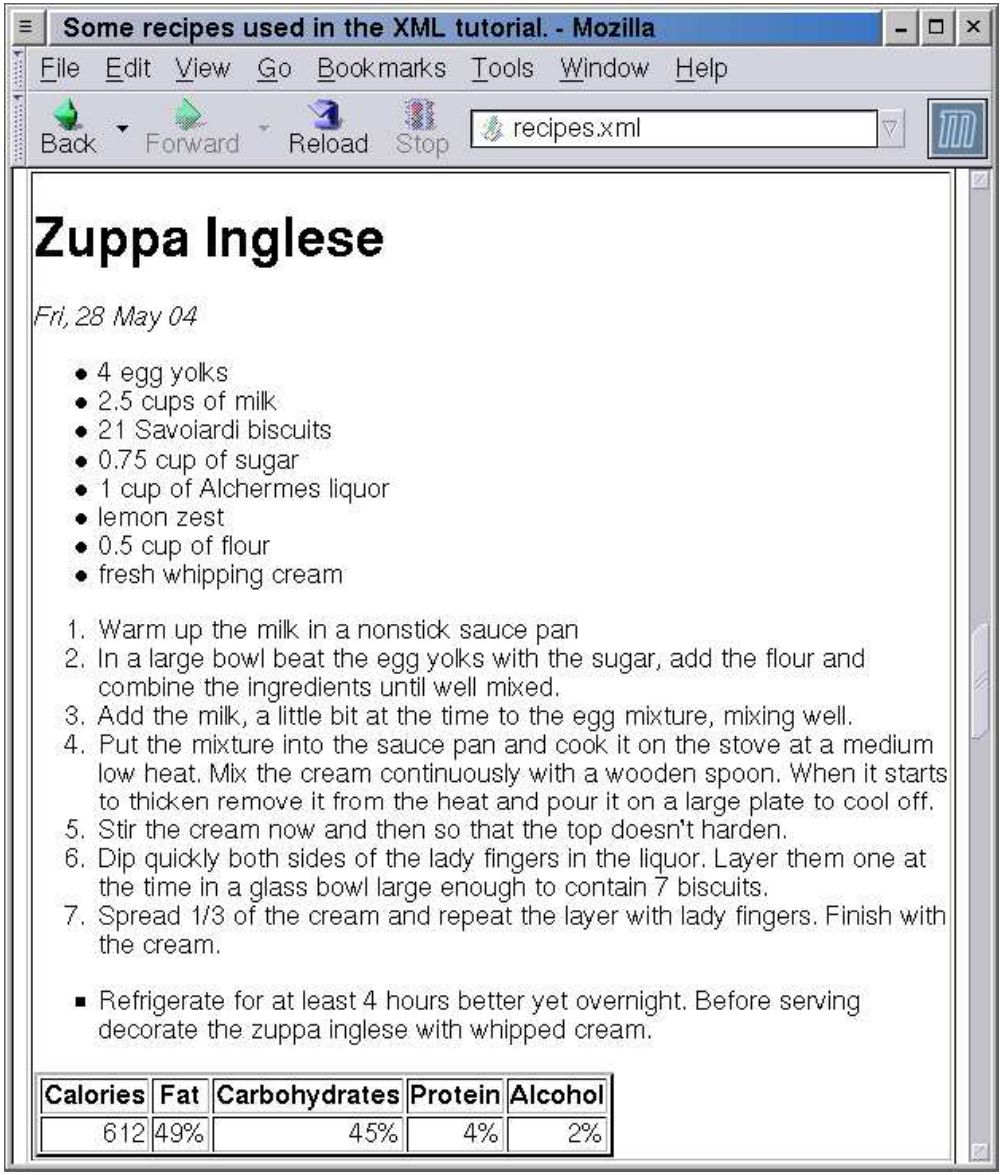
XSLT for Recipes (5/6)

```
<xsl:template match="rcp:preparation">
  <ol><xsl:apply-templates select="rcp:step"/></ol>
</xsl:template>

<xsl:template match="rcp:step">
  <li><xsl:value-of select="node()"/></li>
</xsl:template>

<xsl:template match="rcp:comment">
  <ul>
    <li type="square"><xsl:value-of select="node()"/></li>
  </ul>
</xsl:template>
```

XSLT for Recipes (6/6)

```
<xsl:template match="rcp:nutrition">
    <table border="2">
      <tr>
        <th>Calories</th><th>Fat</th><th>Carbohydrates</th><th>Protein</th>
        <xsl:if test="@alcohol">
          <th>Alcohol</th>
        </xsl:if>
      </tr>
      <tr>
        <td align="right"><xsl:value-of select="@calories"/></td>
        <td align="right"><xsl:value-of select="@fat"/></td>
        <td align="right"><xsl:value-of select="@carbohydrates"/></td>
        <td align="right"><xsl:value-of select="@protein"/></td>
        <xsl:if test="@alcohol">
          <td align="right"><xsl:value-of select="@alcohol"/></td>
        </xsl:if>
      </tr>
    </table>
  </xsl:template>
</xsl:stylesheet>
```

The Output


Some recipes used in the XML tutorial. - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
recipes.xml
Zuppa Inglese
Fri, 28 May 04
4 egg yolks
2.5 cups of milk
21 Savoiardi biscuits
0.75 cup of sugar
1 cup of Alchermes liquor
lemon zest
0.5 cup of flour
fresh whipping cream
1. Warm up the milk in a nonstick sauce pan
2. In a large bowl beat the egg yolks with the sugar, add the flour and combine the ingredients until well mixed.
3. Add the milk, a little bit at the time to the egg mixture, mixing well.
4. Put the mixture into the sauce pan and cook it on the stove at a medium low heat. Mix the cream continuously with a wooden spoon. When it starts to thicken remove it from the heat and pour it on a large plate to cool off.
5. Stir the cream now and then so that the top doesn't harden.
6. Dip quickly both sides of the lady fingers in the liquor. Layer them one at the time in a glass bowl large enough to contain 7 biscuits.
7. Spread 1/3 of the cream and repeat the layer with lady fingers. Finish with the cream.
Refrigerate for at least 4 hours better yet overnight. Before serving decorate the zuppa inglese with whipped cream.
Calories Fat Carbohydrates Protein Alcohol
612 49% 45% 4% 2%

A Different View

```
<xsl:stylesheet version="2.0"
     xmlns:rcp="http://www.brics.dk/ixwt/recipes"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="rcp:collection">
    <nutrition>
      <xsl:apply-templates select="rcp:recipe"/>
    </nutrition>
  </xsl:template>

  <xsl:template match="rcp:recipe">
    <dish name="{rcp:title/text()}"
          calories="{rcp:nutrition/@calories}"
          fat="{rcp:nutrition/@fat}"
          carbohydrates="{rcp:nutrition/@carbohydrates}"
          protein="{rcp:nutrition/@protein}"
          alcohol="{if (rcp:nutrition/@alcohol)
                    then rcp:nutrition/@alcohol else '0%'}"/>
  </xsl:template>
</xsl:stylesheet>
```

The Output

```
<nutrition>
  <dish name="Beef Parmesan with Garlic Angel Hair Pasta"
        calories="1167"
        fat="23%" carbohydrates="45%" protein="32%" alcohol="0%"/>
  <dish name="Ricotta Pie"
        calories="349"
        fat="18%" carbohydrates="64%" protein="18%" alcohol="0%"/>
  <dish name="Linguine Pescadoro"
        calories="532"
        fat="12%" carbohydrates="59%" protein="29%" alcohol="0%"/>
  <dish name="Zuppa Inglese"
        calories="612"
        fat="49%" carbohydrates="45%" protein="4%" alcohol="2%"/>
  <dish name="Cailles en Sarcophages"
        calories="8892"
        fat="33%" carbohydrates="28%" protein="39%" alcohol="0%"/>
</nutrition>
```

A Further Stylesheet

```
<xsl:stylesheet version="2.0"
     xmlns="http://www.w3.org/1999/xhtml"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="nutrition">
    <html>
      <head>
        <title>Nutrition Table</title>
      </head>
      <body>
        <table border="1">
          <tr>
            <th>Dish</th>
            <th>Calories</th>
            <th>Fat</th>
            <th>Carbohydrates</th>
            <th>Protein</th>
          </tr>
          <xsl:apply-templates select="dish"/>
        </table>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="dish">
    <tr>
      <td><xsl:value-of select="@name"/></td>
      <td align="right"><xsl:value-of select="@calories"/></td>
      <td align="right"><xsl:value-of select="@fat"/></td>
      <td align="right"><xsl:value-of select="@carbohydrates"/></td>
      <td align="right"><xsl:value-of select="@protein"/></td>
    </tr>
  </xsl:template>
</xsl:stylesheet>
```

The Final Output

Dish	Calories	Fat	Carbohydrates	Protein
Beef Parmesan with Garlic Angel Hair Pasta	1167	23%	45%	32%
Ricotta Pie	349	18%	64%	18%
Linguine Pescadoro	532	12%	59%	29%
Zuppa Inglese	612	49%	45%	4%
Cailles en Sarcophages	8892	33%	28%	39%

Other Language Features

- Variables and parameters
- Numbering
- Functions
- Sequence types
- Multiple input/output documents
- Dividing a stylesheet into several files
- Stylesheets that generate stylesheets as output

– see the book!

Essential Online Resources

- http://www.w3.org/TR/xslt20/
- http://saxon.sourceforge.net/
- http://www.w3.org/TR/xsl/
- http://xml.apache.org/fop/

Variables and Parameters

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template name="fib">
    <xsl:param name="n"/>
    <xsl:choose>
      <xsl:when test="$n le 1">
        <xsl:value-of select="1"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:variable name="f1">
          <xsl:call-template name="fib">
             <xsl:with-param name="n" select="$n -1"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="f2">
          <xsl:call-template name="fib">
             <xsl:with-param name="n" select="$n -2"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:value-of select="$f1+$f2"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
```

```
<xsl:template match="/">
    <xsl:call-template name="fib">
      <xsl:with-param name="n"
                      select="10"/>
    </xsl:call-template>
  </xsl:template>
</xsl:stylesheet>
```

Grouping

```
<xsl:stylesheet version="2.0"
     xmlns:rcp="http://www.brics.dk/ixwt/recipes"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="rcp:collection">
      <uses>
        <xsl:for-each-group select="//rcp:ingredient"
                            group-by="@name">
           <use name="{current-grouping-key()}"
                count="{count(current-group())}"/>
        </xsl:for-each-group>
      </uses>
   </xsl:template>
</xsl:stylesheet>
```

The Output

```
<uses>
  <use name="beef cube steak" count="1"/>
  <use name="onion, sliced into thin rings" count="1"/>
  <use name="green bell pepper, sliced in rings" count="1"/>
  <use name="Italian seasoned bread crumbs" count="1"/>
  <use name="grated Parmesan cheese" count="1"/>
  <use name="olive oil" count="2"/>
  <use name="spaghetti sauce" count="1"/>
  <use name="shredded mozzarella cheese" count="1"/>
  <use name="angel hair pasta" count="1"/>
  <use name="minced garlic" count="3"/>
  ...
</uses>
```

Numbering

```
<xsl:stylesheet version="2.0"
     xmlns:rcp="http://www.brics.dk/ixwt/recipes"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="rcp:ingredient">
     <rcp:ingredient>
       <xsl:attribute name="level">
         <xsl:number level="multiple" count="rcp:ingredient"/>
       </xsl:attribute>
       <xsl:apply-templates select="@*|*"/>
   </rcp:ingredient>
  </xsl:template>

  <xsl:template match="@*">
    <xsl:copy/>
  </xsl:template>

  <xsl:template match="*">
    <xsl:copy><xsl:apply-templates/></xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

Functions

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:local="http://www.w3.org/2004/07/xquery-local-functions">
  <xsl:function name="local:fib">
    <xsl:param name="n"/>
    <xsl:value-of select="if ($n le 1)
                          then 1
                          else local:fib($n -1)+local:fib($n -2)"/>
  </xsl:function>

  <xsl:template match="/">
    <xsl:value-of select="local:fib(10)"/>
  </xsl:template>
</xsl:stylesheet>
```

Multiple Input Documents

```
<xsl:stylesheet version="2.0"
     xmlns:rcp="http://www.brics.dk/ixwt/recipes"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="rcp:collection">
     <rcp:collection>
        <rcp:title>Selected Recipes</rcp:title>
        <xsl:apply-templates select="rcp:recipe"/>
     </rcp:collection>
  </xsl:template>

  <xsl:template match="rcp:recipe">
    <xsl:variable name="t" select="rcp:title/text()"/>
    <xsl:if test="not(doc('dislikes.xml')//
                      rcp:recipe[rcp:title eq $t])">
      <xsl:copy-of select="."/>
    </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Multiple Output Documents (1/2)

```
<xsl:stylesheet version="2.0"
     xmlns="http://www.w3.org/1999/xhtml"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="students">
     <xsl:result-document href="names.html">
        <html>
          <head><title>Students</title></head>
          <body>
             <xsl:apply-templates select="student" mode="name"/>
          </body>
        </html>
     </xsl:result-document>
     <xsl:result-document href="grades.html">
        <html>
          <head><title>Grades</title></head>
          <body>
             <xsl:apply-templates select="student" mode="grade"/>
          </body>
        </html>
     </xsl:result-document>
  </xsl:template>
```

Multiple Output Documents (2/2)

```
  <xsl:template match="student" mode="name">
    <a href="grades.html#{@id}"><xsl:value-of select="name"/></a>
    <br/>
  </xsl:template>

  <xsl:template match="student" mode="grade">
    <a name="{@id}"/>
    <xsl:value-of select="name"/>
    <ul>
      <xsl:apply-templates select="results/result"/>
    </ul>
  </xsl:template>

   <xsl:template match="result">
      <li>
        <xsl:value-of select="@course"/>:
        <xsl:text> </xsl:text>
        <xsl:value-of select="@grade"/>
      </li>
   </xsl:template>
</xsl:stylesheet>
```

The First Output

```
<html>
  <head><title>Students</title></head>
  <body>
    <a href="grades.html#100026">Joe Average</a>
    <br/>
    <a href="grades.html#100078">Jack Doe</a>
    <br/>
  </body>
</html>
```

The Second Output

```
<head>
  <title>Grades</title></head>
  <body>
    <a name="100026"/>Joe Average</a>
    <ul>
      <li>Math 101: C-</li>
      <li>Biology 101: C+</li>
      <li>Statistics 101: D</li>
    </ul>
    <a name="100078"/>Jack Doe</a>
    <ul>
      <li>Math 101: A</li>
      <li>XML 101: A-</li>
      <li>Physics 101: B+</li>
      <li>XML 102: A</li>
    </ul>
  </body>
</html>
```

Including a Stylesheet

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="howabout">
    <answer>
      I don't like <xsl:value-of select="text()"/>
    </answer>
  </xsl:template>
</xsl:stylesheet>
```

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:include href="negative.xsl"/>
  <xsl:template match="*">
    <answer>
      I'm crazy for <xsl:value-of select="text()"/>
    </answer>
  </xsl:template>
</xsl:stylesheet>
```

```
<howabout>Zuppa Inglese</howabout>
```

```
<answer>I don't like Zuppa Inglese</answer>
```

Importing a Stylesheet

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="howabout">
    <answer>
      I don't like <xsl:value-of select="text()"/>
    </answer>
  </xsl:template>
</xsl:stylesheet>
```

```
<xsl:stylesheet version="2.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="negative.xsl"/>
  <xsl:template match="*">
    <answer>
      I'm crazy for <xsl:value-of select="text()"/>
    </answer>
  </xsl:template>
</xsl:stylesheet>
```

```
<howabout>Zuppa Inglese</howabout>
```

```
<answer>I'm crazy for Zuppa Inglese</answer>
```

Multilingual Business Cards

```
<translate language="Danish">
  <card>kort</card>
  <name>navn</name>
  <title>titel</title>
  <email>email</email>
  <phone>telefon</phone>
  <logo>logo</logo>
</translate>
```

```
<translate language="French">
  <card>carte</card>
  <name>nom</name>
  <title>titre</title>
  <email>courriel</email>
  <phone>telephone</phone>
  <logo>logo</logo>
</translate>
```

Generating Stylesheets (1/2)

```
<xsl:stylesheet version="2.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:b="http://businesscard.org"
                xmlns:myxsl="foo">

  <xsl:namespace-alias stylesheet-prefix="myxsl" result-prefix="xsl"/>

  <xsl:template match="translate">
     <myxsl:stylesheet version="2.0">
        <xsl:namespace name=""
             select="concat('http://businesscard.org/',@language)"/>
        <myxsl:template match="b:card">
           <myxsl:element name="{card}">
              <myxsl:apply-templates/>
           </myxsl:element>
        </myxsl:template>
        <myxsl:template match="b:name">
           <myxsl:element name="{name}">
              <myxsl:value-of select="."/>
           </myxsl:element>
        </myxsl:template>
```

Generating Stylesheets (2/2)

```
<myxsl:template match="b:title">
          <myxsl:element name="{title}">
             <myxsl:value-of select="."/>
          </myxsl:element>
       </myxsl:template>
       <myxsl:template match="b:email">
          <myxsl:element name="{email}">
             <myxsl:value-of select="."/>
          </myxsl:element>
       </myxsl:template>
       <myxsl:template match="b:phone">
          <myxsl:element name="{phone}">
             <myxsl:value-of select="."/>
          </myxsl:element>
       </myxsl:template>
       <myxsl:template match="b:logo">
          <myxsl:element name="{logo}">
             <myxsl:attribute name="uri" select="@uri"/>
          </myxsl:element>
       </myxsl:template>
    </myxsl:stylesheet>
 </xsl:template>
```

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

Generated Stylesheet (1/2)

```
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:b="http://businesscard.org"
  xmlns="http://businesscard.org/French">
  <xsl:template match="b:card">
    <xsl:element name="carte">
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="b:name">
    <xsl:element name="nom">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="b:title">
    <xsl:element name="titre">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
```

Generated Stylesheet (2/2)

```
 <xsl:template match="b:email">
    <xsl:element name="courriel">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="b:phone">
    <xsl:element name="telephone">
      <xsl:value-of select="."/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="b:logo">
    <xsl:element name="logo">
      <xsl:attribute name="uri" select="@uri"/>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Business Card Translation

```
<card xmlns="http://businesscard.org">
  <name>John Doe</name>
  <title>CEO, Widget Inc.</title>
  <email>john.doe@widget.inc</email>
  <phone>(202) 555-1414</phone>
  <logo uri="widget.gif"/>
</card>
```

```
<carte xmlns="http://businesscard.org/French">
  <nom>John Doe</nom>
  <titre>CEO, Widget Inc.</titre>
  <courriel>john.doe@widget.inc</courriel>
  <telephone>(202) 555-1414</telephone>
  <logo uri="widget.gif"/>
</carte>
```

Red, Blue, and Sorted

- Transform this list of number to be:
  - sorted
  - alternatingly red and blue

```
<integerlist>
  <int>15</int>
  <int>12</int>
  <int>17</int>
  <int>25</int>
  <int>18</int>
  <int>17</int>
  <int>23</int>
</integerlist>
```

XSLT 2.0 Solution (1/2)

```
<xsl:template match="integerlist">
  <html>
    <head>
      <title>Integers</title>
    </head>
    <body>
      <xsl:variable name="sorted">
        <xsl:for-each select="int">
          <xsl:sort select="." data-type="number"/>
          <xsl:copy-of select="."/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:apply-templates select="$sorted"/>
    </body>
  </html>
</xsl:template>
```

XSLT 2.0 Solution (2/2)

```
<xsl:template match="int">
  <li>
    <font>
      <xsl:attribute name="color"
         select="if (position() mod 2 = 0) then 'blue'
                                           else 'red'"/>
      <xsl:value-of select="text()"/>
    </font>
  </li>
</xsl:template>
```

XSLT 1.0 Solution (1/3)

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

  <xsl:template match="integerlist">
    <xsl:copy>
      <xsl:apply-templates>
        <xsl:sort select="." data-type="number"/>
      </xsl:apply-templates>
    </xsl:copy>
  </xsl:template>

  <xsl:template match="int">
    <xsl:copy-of select="."/>
  </xsl:template>

</xsl:stylesheet>
```

XSLT 1.0 Solution (2/3)

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

  <xsl:template match="integerlist">
    <html>
      <head>
        <title>Integers</title>
      </head>
      <body>
        <xsl:apply-templates/>
      </body>
    </html>
  </xsl:template>
```

XSLT 1.0 Solution (3/3)

```
  <xsl:template match="int[position() mod 2 = 0]">
    <li>
      <font color="blue">
        <xsl:value-of select="text()"/>
      </font>
    </li>
  </xsl:template>

  <xsl:template match="int[position() mod 2 = 1]">
    <li>
      <font color="red">
        <xsl:value-of select="text()"/>
      </font>
    </li>
  </xsl:template>
</xsl:stylesheet>
```