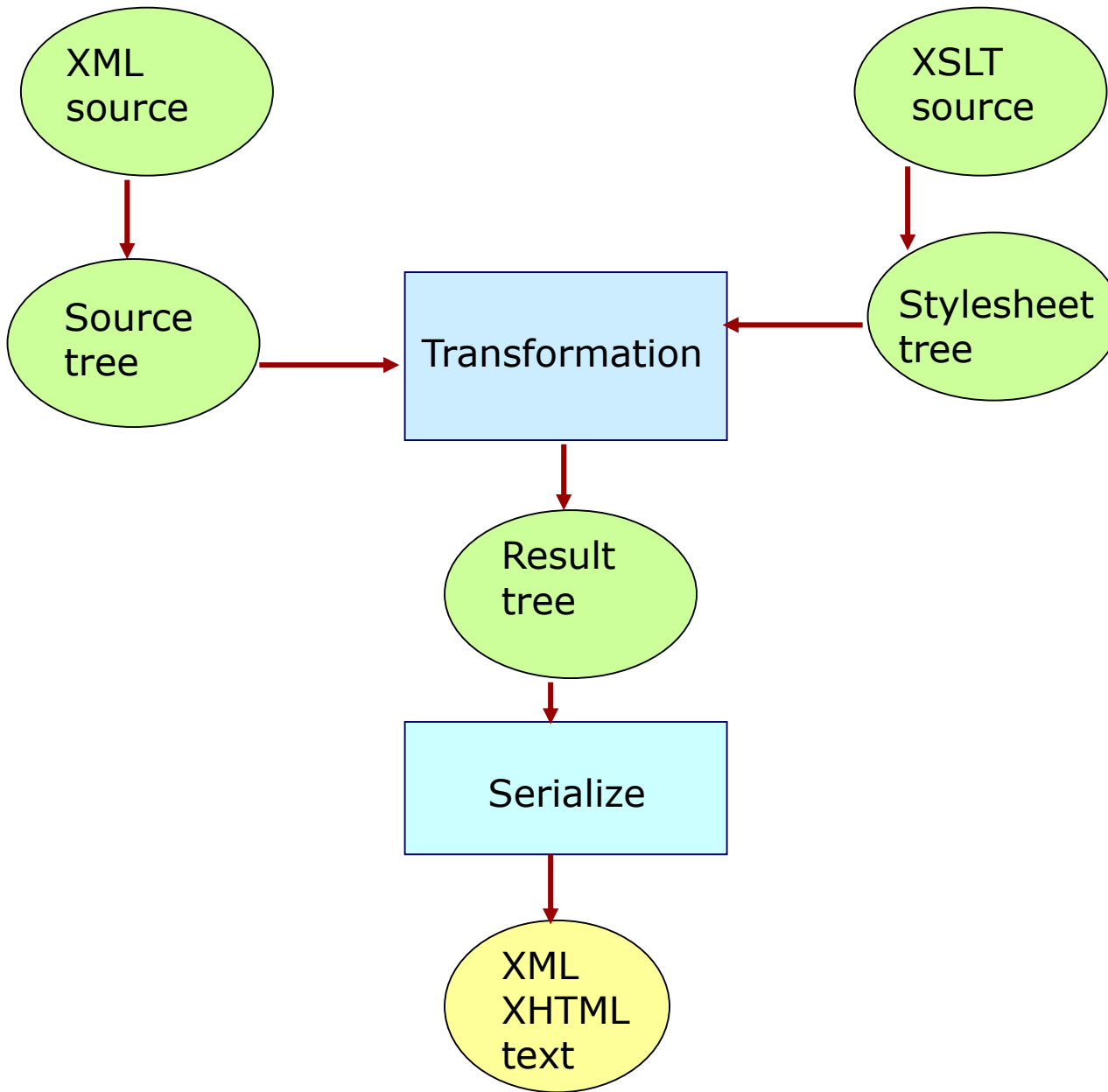
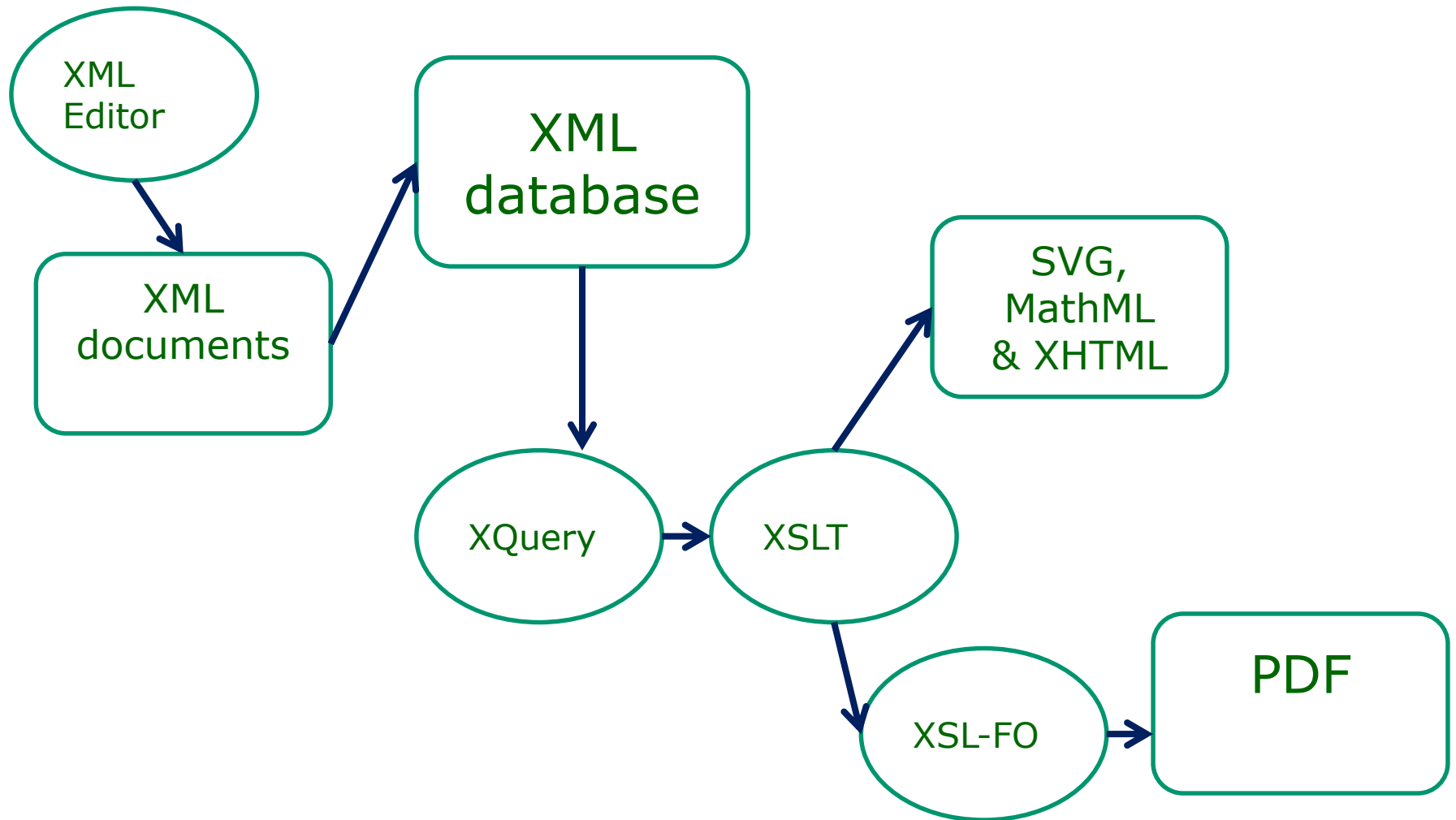


# **XSLT programming examples**

**Jaana Holvikivi**  
**Metropolia**



## Case: book printer's system



# Recursion: xml file

```
<?xml version='1.0'?>
<?xml-stylesheet type="text/xsl" href="books.xsl"?>
<library>
  <book>
    <title>Pennies from heaven</title>
    <author>Mae West</author>
  </book>
  <book>
    <title>Memories</title>
    <author>Bertrand Russel</author>
  </book>
  <book>
    <title>Investigations</title>
    <author>Ludwig Wittgenstein</author>
  </book>
</library>
```

# Recursion: books.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:template match="/">
<html><body>
  <xsl:element name="table">
    <xsl:apply-templates select="//book"/>
  </xsl:element>
</body></html>
</xsl:template>
▪ ...continues
```

## Recursion (continues)

```
<xsl:template match="*">  
  <xsl:if test="count(ancestor::* ) =1">  
    <xsl:element name="tr">  
      <xsl:apply-templates select="child::*"/>  
    </xsl:element>  
  </xsl:if>  
  <xsl:if test="count(ancestor::* ) !=1">  
    <xsl:element name="td">  
      <xsl:value-of select="."/>  
    </xsl:element>  
  </xsl:if>  
</xsl:template>  
</xsl:stylesheet>
```

## Sorting, second example

```
<?xml version="1.0"?>  
<?xml-stylesheet type="text/xsl" href="mooming.xsl"?>  
<crew>  
  <member name="Mamma" gear="handbag" cloth="apron"/>  
  <member name="Pappa" gear="pipe" cloth="hat"/>  
  <member name="Mymlan" gear="mirror" cloth="dress"/>  
</crew>
```

# xsl:sort

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

<xsl:template match="/">
<html>
<head><title>Moomin belongings sorted</title></head>
<body>
  <xsl:for-each select="//member">
    <xsl:sort select="@name" />
    <p><xsl:value-of select="@name"/>
    <ul>
      <li><xsl:value-of select="@gear"/></li>
      <li><xsl:value-of select="@cloth"/></li>
    </ul></p>
  </xsl:for-each>
</body></html>
</xsl:template>
</xsl:stylesheet>
```



# Generic process for copying

```
<?xml version="1.0"?>
```

```
<xsl:stylesheet version="1.0"
```

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

```
<xsl:output method="xml" />
```

```
<xsl:template match=" * | @* | processing-instruction() ">
```

```
  <xsl:copy>
```

```
    <xsl:apply-templates select=" * | @* | text | processing-instruction() "/>
```

```
  </xsl:copy>
```

```
</xsl:template>
```

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

## Note on selection syntax

Attribute values cannot contain "<" nor ">"

Consequently, the following is NOT valid:

**<Body bgcolor="<xsl:value-of select='/FitnessCenter/Member/FavoriteColor'/>">**

To extract the value of an XML element and use it as an attribute value you must use curly braces:

**<Body bgcolor="{/FitnessCenter/Member/FavoriteColor}">**

Want this:

<

>

=

<=

>=

!=

Use this:

&lt;

&gt;

=

&lt;=

&gt;=

!=

## Variables and parameters: the alphabet one by one

```
<xsl:template name="alphabetTemplate">
  <xsl:param name="alphabet" select="
  'ABCDEFGHIJKLMNOPQRSTUVWXYZ' " />
  <xsl:variable name="letter" select="substring($alphabet, 1, 1)" />
  <xsl:variable name="remainder" select="substring($alphabet, 2)" />
  ....
  <xsl:if test="$remainder">
    <xsl:call-template name="alphabetTemplate"/>
    <xsl:with-param name="alphabet" select=" remainder " />
    </xsl:call-template>
  </xsl:if>
</xsl:template>
```

# Another HTML example:

An XML document representing the results of a soccer tournament

```
<results group="A">
  <match>
    <date>10-Jun-1998</date>
    <team score="2">Brazil</team>
    <team score="1">Scotland</team>
  </match>
  <match>
    <date>10-Jun-1998</date>
    <team score="2">Morocco</team>
    <team score="2">Norway</team>
  </match>
  <match>
    <date>16-Jun-1998</date>
    <team score="1">Scotland</team>
    <team score="1">Norway</team>
  </match>
```

# Another HTML example, page 2:

An XML document representing the results of a soccer tournament

```
<match>
  <date>16-Jun-1998</date>
  <team score="3">Brazil</team>
  <team score="0">Morocco</team>
</match>
<match>
  <date>23-Jun-1998</date>
  <team score="1">Brazil</team>
  <team score="2">Norway</team>
</match>
<match>
  <date>23-Jun-1998</date>
  <team score="0">Scotland</team>
  <team score="3">Morocco</team>
</match>
</results>
```

# A basic style sheet for the soccer results

```
<xsl:transform
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="results">
    <html>
      <head><title>
        Results of Group <xsl:value-of select="@group"/>
      </title></head>
      <body><h1>
        Results of Group <xsl:value-of select="@group"/>
      </h1>
      <xsl:apply-templates/>
    </body></html>
  </xsl:template>
  <xsl:template match="match">
    <h2>
      <xsl:value-of select="team[1]"/> versus <xsl:value-of select="team[2]"/>
    </h2>
    <p>Played on <xsl:value-of select="date"/></p>
    <p>Result:
      <xsl:value-of select="team[1]"/>
      <xsl:value-of select="team[1]/@score"/>,
      <xsl:value-of select="team[2]"/>
      <xsl:value-of select="team[2]/@score"/>
    </p>
  </xsl:template>
</xsl:transform>
```

## A style sheet that computes team standings (part 1)

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

  <xsl:variable name="teams" select="//team[not(.=preceding::team)]"/>
  <xsl:variable name="matches" select="//match"/>

  <xsl:template match="results">

    <html><body>
      <h1>Results of Group <xsl:value-of select="@group"/></h1>

      <table cellpadding="5">
        <tr>
          <td>Team</td>
          <td>Played</td>
          <td>Won</td>
          <td>Drawn</td>
          <td>Lost</td>
          <td>For</td>
          <td>Against</td>
        </tr>
```

## A style sheet that computes team standings (part 2)

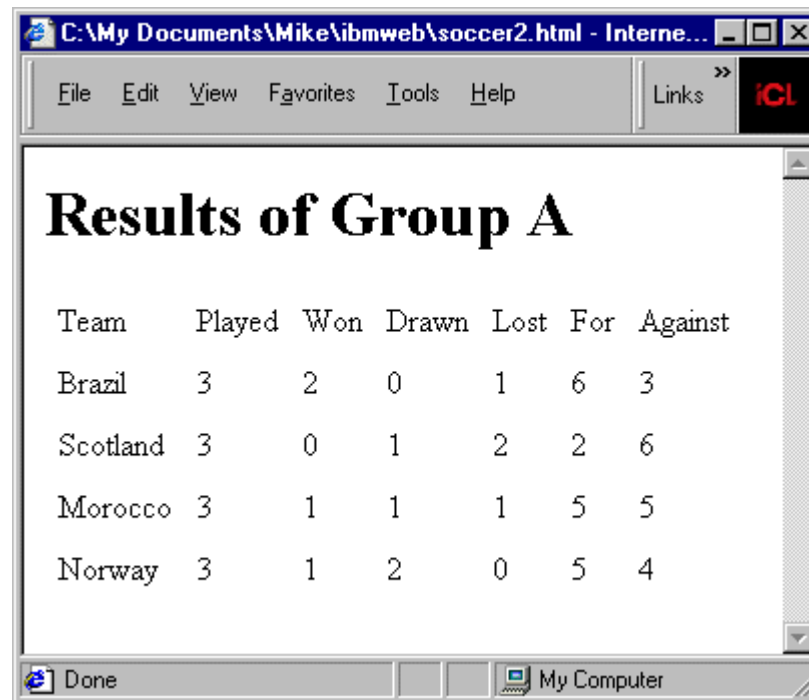
```
<xsl:for-each select="$teams">
  <xsl:variable name="this" select="."/>
  <xsl:variable name="played" select="count($matches[team=$this])"/>

  <xsl:variable name="won"
    select="count($matches[team[.=$this]/@score > team[!=$this]/@score])"/>
  <xsl:variable name="lost"
    select="count($matches[team[.=$this]/@score < team[!=$this]/@score])"/>
  <xsl:variable name="drawn"
    select="count($matches[team[.=$this]/@score = team[!=$this]/@score])"/>
  <xsl:variable name="for"
    select="sum($matches/team[.=current()]/@score)"/>
  <xsl:variable name="against"
    select="sum($matches[team=current()]/team/@score) - $for"/>
  <tr>
    <td><xsl:value-of select="."/></td>
    <td><xsl:value-of select="$played"/></td>
    <td><xsl:value-of select="$won"/></td>
    <td><xsl:value-of select="$drawn"/></td>
    <td><xsl:value-of select="$lost"/></td>
    <td><xsl:value-of select="$for"/></td>
    <td><xsl:value-of select="$against"/></td>
  </tr>
</xsl:for-each>
</table>
</body></html>
</xsl:template>
</xsl:transform>
```



# A style sheet that computes team standings: result

- The data are rearranged and processed:



The screenshot shows an Internet Explorer browser window with the address bar displaying 'C:\My Documents\Mike\ibmweb\soccer2.html - Interne...'. The menu bar includes 'File', 'Edit', 'View', 'Favorites', 'Tools', 'Help', and 'Links'. The main content area displays the title 'Results of Group A' and a table of soccer results. The table has seven columns: Team, Played, Won, Drawn, Lost, For, and Against. The data rows are for Brazil, Scotland, Morocco, and Norway. The status bar at the bottom shows 'Done' and 'My Computer'.

Team	Played	Won	Drawn	Lost	For	Against
Brazil	3	2	0	1	6	3
Scotland	3	0	1	2	2	6
Morocco	3	1	1	1	5	5
Norway	3	1	2	0	5	4

# Explanations

```
<xsl:variable name="teams" select="//team[not(.=preceding::team)]"/>
```

create a global variable *teams* (node-set which has all team elements, the same element cannot be selected twice)

this node set is processed in the loop on the next page

```
<xsl:for-each select="$teams">
```

 the dollar sign refers to variable *teams*

```
  <xsl:variable name="this" select="."/>
```

 declare a local variable *this*, that gets the value of the current node

```
count($matches[team=$this])
```

counts the number of those matches for which it is true that the name of this team is in the element value

```
/@score > team[!=$this]/@score
```

compare values of score attributes: has this team more (> &gt;) scores than the other; != means NOT

# XSL as a functional programming language: the factorial

```
<?xml version="1.0"?>
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <!-- Defining and Calling the Factorial Function in XSLT -->
    <!-- A separate file factorial-main.xml provides specific arguments -->
    <!-- call factorial on selected integer argument n -->
    <xsl:template match="/arguments/a1">
      <html>
        <head>
          <title>factorial(<xsl:value-of select="."/>)</title>
        </head>
        <body>
          <xsl:call-template name="factorial">
            <xsl:with-param name="n" select="."/>
          </xsl:call-template>
        </body>
      </html>
    </xsl:template>
  </xsl:stylesheet>
```

# XSL as a functional programming language: the factorial, cont.

- ```
<xsl:template name="factorial">
<xsl:param name="n"/>
<xsl:choose>
<xsl:when test="$n = 0">1</xsl:when>
<!-- factorial(0) = 1 -->
<xsl:when test="$n > 0"> <!-- factorial(n) = -->
<xsl:variable name="factor">
<xsl:call-template name="factorial">
<xsl:with-param name="n" select="$n - 1"/>
</xsl:call-template>
</xsl:variable>
<xsl:value-of select="$n * $factor"/> <!-- n*factorial(n-1) -->
</xsl:when>
</xsl:choose>
</xsl:template>
```