XSLT

Rasmus Pagh

An Introduction to XML and Web Technologies

Transforming XML Documents with XSLT

based on slides by
Anders Møller & Michael I. Schwartzbach

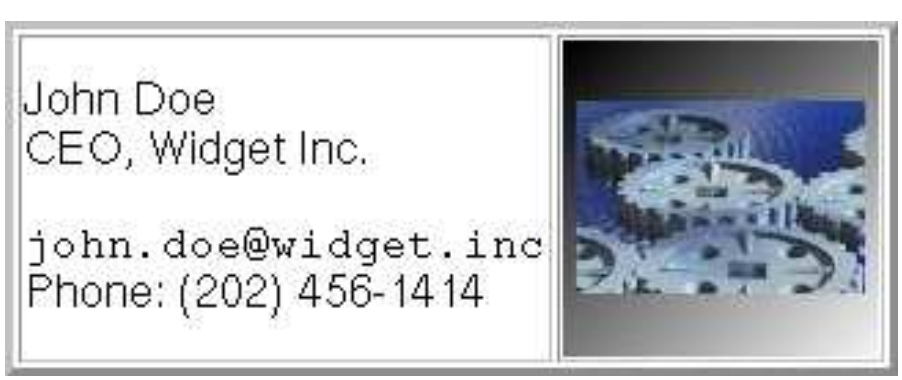
Presenting a Business Card

```
<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>
```

```
- <card>
    <name>John Doe</name>
    <title>CEO, Widget Inc.</title>
    <email>john.doe@widget.inc</email>
    <phone>(202) 456-1414</phone>
    <logo uri="widget.gif"/>
  </card>
```

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>
```



XHTML

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>
```

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
- 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

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 //, possibly followed by a filter […].
  - steps 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
```

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 Transformation

```
<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>
```

XSLT for student transformation

```
<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>
```

Sorting

```
<xsl:template match="students">
    <summary>
    <xsl:apply-templates select="student">
       <xsl:sort select=”age"/>
    </xsl:apply-templates>
</summary>
</xsl:template>
```

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

Acknowledgement

- Thanks to Anders Møller, co-author of *An Introduction to XML and Web Technologies* for allowing me to use his slides **without** forcing students to buy his book!
- But if you want an in-depth XML book, the book is recommended.
  - Now also in Italian!