XSLT
(part II)

Mario Alviano

University of Calabria, Italy

A.Y. 2016/2017

Outline

Outline

Modalities

- Templates can specify a **modality** in the attribute **mode**
- Attribute `mode` can be used also in `<xsl:apply-templates>`
- This allows to have several templates for the same selection

Modalities

- Templates can specify a **modality** in the attribute **mode**
- Attribute `mode` can be used also in `<xsl:apply-templates>`
- This allows to have several templates for the same selection
- The default value of `mode` is `#default`
- Value `#all` can be used to apply **all modes!**

```
<xsl:apply-templates select="list-item" mode="idx" />
...
<xsl:template match="//list-item" mode="idx">
    ...
</xsl:template>
```

Outline

Variables

- A variable can be declared by `<xsl:variable>`

```
<xsl:variable name="foo" select="feed/link/@href" />
```

Variables

- A variable can be declared by `<xsl:variable>`

```
<xsl:variable name="foo" select="feed/link/@href" />
```

- To access the variable use $ followed by the name of the variable

```
<xsl:value-of select="$foo" />
```

Variables

- A variable can be declared by `<xsl:variable>`

```
<xsl:variable name="foo" select="feed/link/@href" />
```

- To access the variable use $ followed by the name of the variable

```
<xsl:value-of select="$foo" />
```

- **Warning!** Variables cannot be reassigned. Their value is fixed. Look at them as first-order variables!

Conditional constructs

The if element

```
<xsl:if test="...">
    ...
</xsl:if>
```

Conditional constructs

The if element

```
<xsl:if test="...">
    ...
</xsl:if>
```

The choose element (more or less a switch)

```
<xsl:choose>
    <xsl:when test="...">...</xsl:when>
    <xsl:when test="...">...</xsl:when>
    ...
    <xsl:otherwise>...</xsl:otherwise>
</xsl:chose>
```

Conditional constructs

The if element

```
<xsl:if test="...">
    ...
</xsl:if>
```

The choose element (more or less a switch)

```
<xsl:choose>
    <xsl:when test="...">...</xsl:when>
    <xsl:when test="...">...</xsl:when>
    ...
    <xsl:otherwise>...</xsl:otherwise>
</xsl:chose>
```

The if..then..else instruction

```
<xsl:value-of select="if $foo >= 0 then $foo else -1 * $foo" />
```

Iterate over sequences

- To iterate over a sequence use `<xsl:for-each>`

```
<xsl:for-each select="//list-item">
   <li><xsl:apply-templates /></li>
</xsl:for-each>
```

- Elements are processed in the order in which they are retrieved from the document

Iterate over sequences

- To iterate over a sequence use `<xsl:for-each>`

```
<xsl:for-each select="//list-item">
   <li><xsl:apply-templates /></li>
</xsl:for-each>
```

- Elements are processed in the order in which they are retrieved from the document
- A different order can be specified by `<xsl:sort>` (one or more instruction, processed in the given order)

```
<xsl:sort select="xpath expression" [order="descending"] />
```

Outline

Sorting

- Elements `<xsl:sort>` can also be used in `<xsl:perform-sort>`
- This allows to define sequences to be used later

```
<xsl:variable name="foo">
    <xsl:perform-sort select="//list-item">
        <xsl:sort select="@price" />
    </xsl:perform-sort>
</xsl:variable>
```

Sorting

- Elements `<xsl:sort>` can also be used in `<xsl:perform-sort>`
- This allows to define sequences to be used later

```
<xsl:variable name="foo">
   <xsl:perform-sort select="//list-item">
      <xsl:sort select="@price" />
   </xsl:perform-sort>
</xsl:variable>
```

- **Warning!** An engine for XSLT 2.0 is required!
  - `http://saxon.sourceforge.net/#F9.4HE`
  - `$ java -jar saxon9he.jar -xsl:foo.xsl -s:foo.xml`

Grouping

- Elements obtained by a selection can be grouped

```
<xsl:for-each-group select="..." group-by="...">
    ...
    <xsl:value-of select="current-grouping-key()" />
    ...
    <xsl:for-each select="current-group()">
        ...
    </xsl:for-each>
    ...
</xsl:for-each-group>
```

Grouping

- Elements obtained by a selection can be grouped

```
<xsl:for-each-group select="..." group-by="...">
    ...
    <xsl:value-of select="current-grouping-key()" />
    ...
    <xsl:for-each select="current-group()">
        ...
    </xsl:for-each>
    ...
</xsl:for-each-group>
```

- **Warning!** An engine for XSLT 2.0 is required!

Outline

Named templates

- Templates may specify a name by the attribute `name`
- To invoke a named template use `<xsl:call-template name="..." />`
- Named templates are like functions

Named templates

- Templates may specify a name by the attribute `name`
- To invoke a named template use `<xsl:call-template name="..." />`
- Named templates are like functions
- Parameters can be added by `<xsl:param>`

```
<xsl:param name="myParam" />
<xsl:param name="myParam" select="xpath expression" />
<xsl:param name="myParam">value</xsl:param>
```

- **Note:** a default value can be specified by `select` or in the content.

Named templates

- Templates may specify a name by the attribute `name`
- To invoke a named template use `<xsl:call-template name="..." />`
- Named templates are like functions
- Parameters can be added by `<xsl:param>`

```
<xsl:param name="myParam" />
<xsl:param name="myParam" select="xpath expression" />
<xsl:param name="myParam">value</xsl:param>
```

- **Note:** a default value can be specified by `select` or in the content.
- To specify that a parameter is required use `required="yes"`
- To invoke a template with parameter use `<xsl:with-param>` in `<xsl:call-template>`

Outline

Exercise 1

- Write an XSLT transforming the document **library.xml** to the HTML page shown below

Library Alpha

- Bravo
  1. Charlie: *Traveling with a poodle*
  2. Delta: *Mouth of the Mississippi*
- Echo
  1. Foxtrot: *Dance to four-quarters time*
  2. **Golf**
     1. Hotel: *Check in, but not out*
     2. India: *Indus to the Ganges*
- Juliet
  1. **Kilo**
     1. Lima: *Peru is here too*
     2. Mike: *Decorated Sistine Chapel*
  2. **November**
     1. Oscar: *Academy Awards*
     2. Papa: *To me he was so wonderful*

Book number	Book title	Part number	Part title	Chapter number	Chapter title	Chapter content
1	Bravo			1	Charlie	Traveling with a poodle
1	Bravo			2	Delta	Mouth of the Mississippi
2	Echo			1	Foxtrot	Dance to four-quarters time
2	Echo	1	Golf	1	Hotel	Check in, but not out
2	Echo	1	Golf	2	India	Indus to the Ganges
3	Juliet	1	Kilo	1	Lima	Peru is here too
3	Juliet	1	Kilo	2	Mike	Decorated Sistine Chapel
3	Juliet	2	November	1	Oscar	Academy Awards
3	Juliet	2	November	2	Papa	To me he was so wonderful

Exercise 2

- Retrieve document **bank-accounts.xml**
- Write an XSLT transforming the document to the HTML page shown below
  - For bank accounts, emphasize in bold negative balances
  - For deposit accounts, add the note shown in the table if the balance is >= 10000

Monte dei Paschi di Rende

Vista per conti

Conti corrente

ID conto	Cliente	Saldo
c1	Ben Richerdson	4025
c2	Ben Richerdson	**-125**
c3	Marc Wretcher	325

Conti deposito

ID conto	Cliente	Saldo	Tasso interesse	Interessi	Note
d1	Ben Richerdson	2500	3%	75	
d2	Angel Steady	15075	3%	452.25	Proporre opzione gold

Exercise 3

- Add a view for the customers
- Emphasize in bold the negative balances
- The result is shown on the right

Vista per clienti

Ben Richerdson

ID conto	Saldo
c2	**-125**
d1	2500
c1	4025
Totale	**6400**

Marc Wretcher

ID conto	Saldo
c3	325
Totale	**325**

Angel Steady

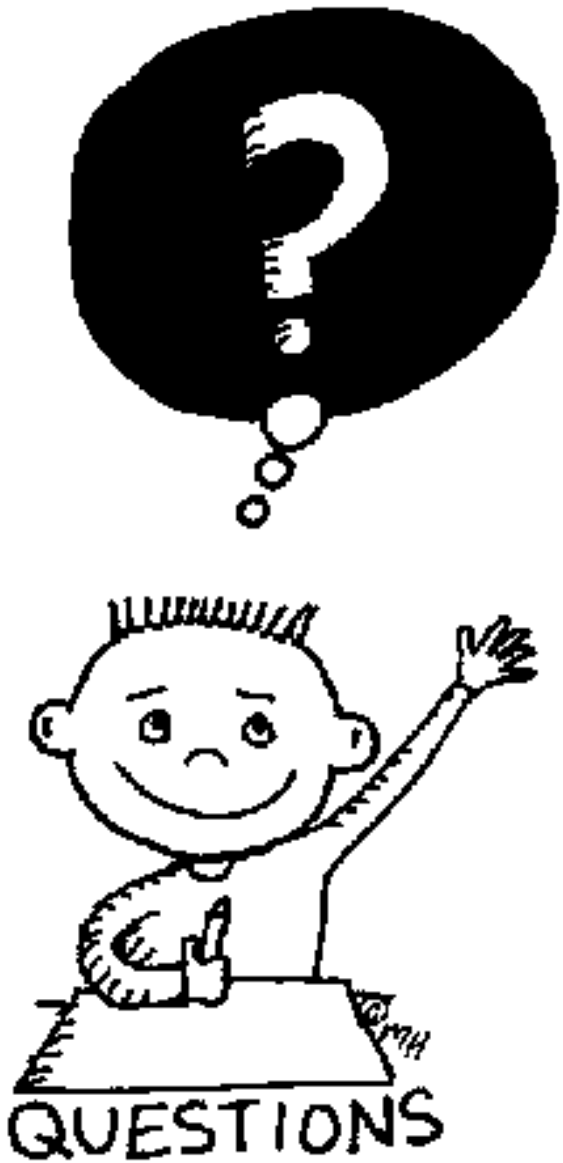
ID conto	Saldo
d2	15075
Totale	**15075**

Exercise 4

1. Write an XSLT transforming the document **ItemList.xml** sorting Item elements by ItemName
2. Write an XSLT grouping items by category (sort categories by CategoryName; the result is shown on the right)

Hint: Using XSLT 2.0 the last point above is simpler

```
<?xml version="1.0" encoding="utf-8"?>
<categories>
    <category id="1">
        <name>Book</name>
        <item>
            <name>ASP.NET</name>
            <details>ASP.NET12345</details>
        </item>
        <item>
            <name>PHP</name>
            <details>PHP12345</details>
        </item>
    </category>
    <category id="2">
        <name>Tool</name>
        <item>
            <name>ToolAbcde</name>
            <details>sth details</details>
        </item>
    </category>
</categories>
```



END OF THE LECTURE