XSLT
(part I)

Mario Alviano

University of Calabria, Italy

A.Y. 2016/2017

Outline

Outline

What is XSLT?

- XSLT is a (Turing complete) functional language

What is XSLT?

- XSLT is a (Turing complete) functional language
- It used for transforming XML documents to other XML documents (or text document in general)
- Often used to transform XML data to web pages

What is XSLT?

- XSLT is a (Turing complete) functional language
- It used for transforming XML documents to other XML documents (or text document in general)
- Often used to transform XML data to web pages
- Modern web browsers implement XSLT
  - To link an XSLT stylesheet to an XML document use a **processing instruction**

```
<?xml-stylesheet href="file_name.xsl" type="text/xsl" ?>
```

Output format

- XSLT stylesheets are XML files
  - XSLT is an XML application
- Root element `xsl:stylesheet` (or `xsl:transform`)

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">
    ...
</xsl:stylesheet>
```

- By default, XSLT produces in output an XML file
- Different formats can be obtained by specifying `<xsl:output>`

```
<xsl:output method="html" encoding="UTF-8" indent="yes" />
```

Outline

Templates

- XSLT works with templates which are applied on elements selected by XPath expressions

Templates

- XSLT works with templates which are applied on elements selected by XPath expressions
- Templates are defined by `<xsl:template>`
  - Attribute `match` specifies an XPath expression to select nodes of the XML document in input
  - The content defines what to produce in output for any matched node

Templates

- XSLT works with templates which are applied on elements selected by XPath expressions
- Templates are defined by `<xsl:template>`
  - Attribute `match` specifies an XPath expression to select nodes of the XML document in input
  - The content defines what to produce in output for any matched node
- The simplest output is **literal**

Example. Literal output

```
<xsl:template match="/">
    <html>
        <body>
            Hello, World!!!
        </body>
    </html>
</xsl:template>
```

Read input

- It is possible to write in output data read from input
- Use the element `<xsl:value-of>`
  - Attribute `select` specifies what to select in the input file (an XPath expression!)

Example. Use of value-of

```
<xsl:template match="/">
    <html>
        <head>
            <title>
                <xsl:value-of select="reference/body/title" />
            </title>
        </head>
        <body>
            Hello, World!!!
        </body>
    </html>
</xsl:template>
```

Apply other templates

- In the content of a template it is possible to specify the application of other templates
- Use the element `<xsl:apply-templates>`
  - Attribute `select` specifies what to select (again, an XPath expression!)

Example. Use of apply-templates

```
<body>
   <xsl:apply-templates select="reference/body" />
</body>
```

Apply other templates

- In the content of a template it is possible to specify the application of other templates
- Use the element `<xsl:apply-templates>`
  - Attribute `select` specifies what to select (again, an XPath expression!)

Example. Use of apply-templates

```
<body>
   <xsl:apply-templates select="reference/body" />
</body>
```

- For each element in the sequence the most appropriate template will be used

Apply other templates

- In the content of a template it is possible to specify the application of other templates
- Use the element `<xsl:apply-templates>`
  - Attribute `select` specifies what to select (again, an XPath expression!)

Example. Use of apply-templates

```
<body>
   <xsl:apply-templates select="reference/body" />
</body>
```

- For each element in the sequence the most appropriate template will be used
- If no template matches a selected node, XSLT produces in output the textual value of the node and calls `apply-templates` to all children

Apply other templates

- In the content of a template it is possible to specify the application of other templates
- Use the element `<xsl:apply-templates>`
  - Attribute `select` specifies what to select (again, an XPath expression!)

Example. Use of apply-templates

```
<body>
   <xsl:apply-templates select="reference/body" />
</body>
```

- For each element in the sequence the most appropriate template will be used
- If no template matches a selected node, XSLT produces in output the textual value of the node and calls `apply-templates` to all children
- **Note:** if `select` is omitted then all children of the current node are selected

Outline

Attributes

- Use `<xsl:attribute>` to add an attribute to an element
  - The name of the attribute is specified in `name`
  - The value of the attribute is specified in the content

```
<p>
    <xsl:attribute name="align">
        right
    </xsl:attribute>
    ...
</p>
```

is equivalent to

```
<p align="right">
    ...
</p>
```

Groups of attributes

- To create a group of attributes (to be used later)

```
<xsl:attribute-set name="row">
    <xsl:attribute name="border">1</xsl:attribute>
</xsl:attribute-set>
```

Groups of attributes

- To create a group of attributes (to be used later)

```
<xsl:attribute-set name="row">
   <xsl:attribute name="border">1</xsl:attribute>
</xsl:attribute-set>
```

- Attribute `xsl:use-attribute-sets` can be used to specify an attribute-set to be added to an element

```
<td xsl:use-attribute-sets="row">
```

Outline

Elements copy

- `<xsl:copy>` allows to copy a node in output

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

Elements copy

- `<xsl:copy>` allows to copy a node in output

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

- **Warning!** `<xsl:copy>` only produces in output opening and closing tags of the selected node
- For this reason we used `<xsl:apply-templates />` inside `<xsl:copy>`

Elements copy

- `<xsl:copy>` allows to copy a node in output

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

- **Warning!** `<xsl:copy>` only produces in output opening and closing tags of the selected node
- For this reason we used `<xsl:apply-templates />` inside `<xsl:copy>`
- An alternative is `<xsl:copy-of>`

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

Outline

How to apply XSLT stylesheet

- XSLT with libxml
  - `xsltproc XSLTfile XMLfile`

How to apply XSLT stylesheet

- XSLT with libxml
  - `xsltproc XSLTfile XMLfile`
- XSLT with Eclipse EE
  - Select XML and XSLT files
  - Right-click, then **Run as | XSL Transformation**

Exercises (1)

Exercise 1

- Write an XML document with content `<hello>World</hello>`
- Write an XSLT stylesheet transforming the XML document in an HTML file with title “Hello ” followed by the content of the tag `<hello>`
- Moreover, the XSLT must add in the body of the HTML a tag `<h1>` with the same content of the element title
- Apply the transformation
- Link the XSLT to the XML document, then open the XML document with a browser

Exercise 2

- Retrieve file **library.xml**
- Write an XSLT transforming element and attribute names from English to Italian
- **Note:** use `<xsl:value-of>` to obtain attribute values
- **Hint:** define a group of attributes with `<xsl:attribute-set>`

Exercise 3

- 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*

Exercise 4

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

Library Alpha

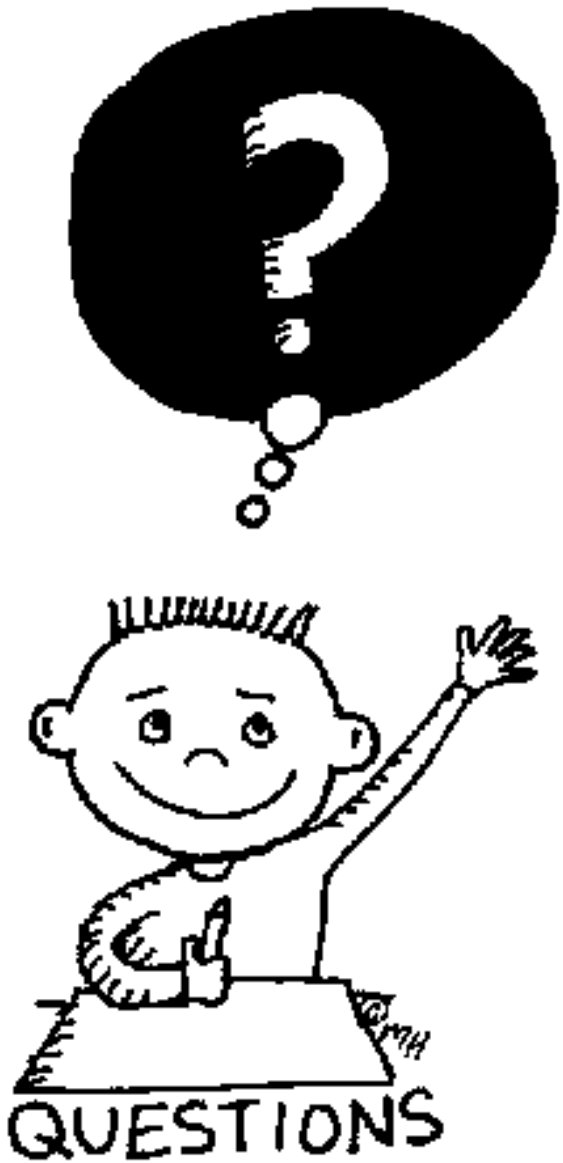
Book title	Part title	Chapter title	Chapter content
Bravo		Charlie	Traveling with a poodle
Bravo		Delta	Mouth of the Mississippi
Echo		Foxtrot	Dance to four-quarters time
Echo	Golf	Hotel	Check in, but not out
Echo	Golf	India	Indus to the Ganges
Juliet	Kilo	Lima	Peru is here too
Juliet	Kilo	Mike	Decorated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

Exercise 5

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

Library Alpha

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



END OF THE LECTURE