# XML

## e**X**tensible **M**arkup **L**anguage

Reda Bendraou
Reda.Bendraou@lip6.fr

http://pagesperso-systeme.lip6.fr/Reda.Bendraou/Enseignements.htm

# Xpath

# Xpath: what it is for?

- **Language for (only) finding information in an XML document**
  - **You can't add new elements**
- **Used to navigate through elements and attributes in an XML document**
- **A major element in the W3C's XSLT standard - and XQuery and XPointer are both built on XPath expressions**
- **Understanding XPath is fundamental to a lot of advanced XML usage**

# What is XPath?

- XPath is a syntax for defining parts of an XML document

- XPath uses path expressions to navigate in XML documents
  - Uses path expressions to select nodes or node-sets in an XML document
  - Path expressions look very much like the expressions in traditional computer file system

- XPath contains a library of standard functions
  - Over 100 built-in functions (string values, numeric values, date and time comparison, etc.)

- **XML documents are treated as trees of nodes**
  - seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document (root) nodes

# Relationship of Nodes

- **Parent**
  - Each element and attribute has one parent
- **Children**
  - Element nodes may have zero, one or more children
- **Siblings**
  - Nodes that have the same parent
- **Ancestors**
  - A node's parent, parent's parent, etc
- **Descendants**
  - A node's children, children's children, etc

# XPath Syntax : Selecting Nodes

- **The node is selected by following a path or steps**

| Expression | Description |
|---|---|
| *nodename* | Selects all child nodes of the current node |
| / | Selects from the root node |
| // | Selects nodes in the document from the current node that match the selection no matter where they are |
| . | Selects the current node |
| .. | Selects the parent of the current node |
| @ | Selects attributes |

# Selecting Nodes :Example !

```
<?xml version="1.0" encoding="ISO-
    8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning
     XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

| Path Expression | Result |
|---|---|
| **bookstore** | **Selects all the child nodes of the bookstore element** |
| **/bookstore** | **Selects the root element bookstore Note: If the path starts with a slash ( / ) it always represents an absolute path to an element!** |
| **bookstore/book** | **Selects all book elements that are children of bookstore** |
| **//book** | **Selects all book elements no matter where they are in the document** |
| **bookstore//book** | **Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element** |
| **//@lang** | **Selects all attributes that are named lang** |

# XPath Syntax : Predicates

- Predicates are used to find a specific node or a node that contains a specific value

- Predicates are always embedded in square brackets

# XPath Predicates : **Example!**

```
<?xml version="1.0" ?>
<bookstore>
  <book>
    <title lang="fr">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>
```

| Path Expression | Result |
|---|---|
| **/bookstore/book[1]** | **Selects the first book element that is the child of the bookstore element** |
| **/bookstore/book[last()]** | **Selects the last book element that is the child of the bookstore element** |
| **/bookstore/book[last()-1]** | **Selects the last but one book element that is the child of the bookstore element** |
| **/bookstore/book[position()<3]** | **Selects the first two book elements that are children of the bookstore element** |
| **//title[@lang]** | **Selects all the title elements that have an attribute named lang** |
| **//title[@lang='eng']** | **Selects all the title elements that have an attribute named lang with a value of 'eng'** |
| **/bookstore/book[price>35.00]** | **Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00** |
| **/bookstore/book[price>35.00]/ title** | **Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00** |

# XPath Syntax : Selecting Unknown Nodes

| Wildcard | Description |
|---|---|
| * | Matches any element node |
| @* | Matches any attribute node |
| node() | Matches any node of any kind |

**Example**

| Path Expression | Result |
|---|---|
| /bookstore/* | Selects all the child nodes of the bookstore element |
| //* | Selects all elements in the document |
| //title[@*] | Selects all title elements which have any attribute |

# Location Path Expression

- A location path can be absolute or relative
  - An absolute location path starts with a slash ( / ) and a relative location path does not
    - An absolute location path:
      **/step/step/...**
    - A relative location path:
      **step/step/...**
    - Each step is evaluated against the nodes in the current node-set

- A step consists of:
  - an axis (defines the tree-relationship between the selected nodes and the current node)
  - a node-test (identifies a node within an axis)
  - zero or more predicates (to further refine the selected node-set)

- Step Syntax
  **axisname::nodetest[predicate]**

# XPath Axes

- An axis defines a node-set relative to the current node

| AxisName | Result |
|---|---|
| ancestor | Selects all ancestors (parent, grandparent, etc.) of the current node |
| ancestor-or-self | Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself |
| attribute | Selects all attributes of the current node |
| child | Selects all children of the current node |
| descendant | Selects all descendants (children, grandchildren, etc.) of the current node |
| descendant-or-self | Selects all descendants of the current node and the current node itself |
| following | Selects everything in the document after the closing tag of the current node |
| following-sibling | Selects all siblings after the current node |
| namespace | Selects all namespace nodes of the current node |
| parent | Selects the parent of the current node |
| preceding | Selects everything in the document that is before the start tag of the current node |
| preceding-sibling | Selects all siblings before the current node |
| self | Selects the current node |

# XPath Axes : Examples

| Example | Result |
|---|---|
| child::book | Selects all book nodes that are children of the current node |
| attribute::lang | Selects the lang attribute of the current node |
| child::* | Selects all children of the current node |
| attribute::* | Selects all attributes of the current node |
| child::text() | Selects all text child nodes of the current node |
| child::node() | Selects all child nodes of the current node |
| descendant::book | Selects all book descendants of the current node |
| ancestor::book | Selects all book ancestors of the current node |
| ancestor-or-self::book | Selects all book ancestors of the current node - and the current as well if it is a book node |
| child::*/child::price | Selects all price grandchildren of the current node |

# XPath expression Examples

## Syntax (full)

- **/descendant::olist/child::item** selects all "item" elements having "olist" as parent of the current node
- **child::para[position()=last()-1]** Selects the Last-1 "para" child of the current node
- **following-sibling::chapter[position()=1]** selects the first "chapter" sibling of the current node
- **/child::doc/child::chapter[position()=5]/child::section[position()=2]** selects the second "section" of the 5th "chapter" element of "doc" document's element
- **child::para[attribute::type='warning'][position()=5]** selects le 5th child "para" of the current node having an attribute "type" with value set to "warning"
- **child::chapter[child::title='Introduction']** selects the child "chapter" of the current node having one or more Childs "title" with a text content equals to 'Introduction'
- **child::*[self::chapter or self::appendix][position()=last()]** selects the last child "chapter" or "appendix" of the current node

# Abbreviations

- child may be **implicit**
  - child::div/child::para => div/para
- attribute may be replaced by @
- /descendant-or-self::node()/ may be replaced by **//**
- self.node() may be replaced by **.**
- parent::node() may be replaced by **..**

# XPath expression Examples

## Abbreviated Syntax (1)

- ***** selects all childs of the current node
- **text()** selects child nodes of text type of the current node
- **@name** selects the attribute "name" of the current node
- **@*** selects all attributes of the current node
- **para[1]** selects the first child "para" of the current node
- **para[last()]** selects the last child "para" of the current node
- ***/para** selects all "para" descendants of the current node
- **/doc/chapter[5]/section[2]** selects the second "section" of the 5th "chapter" of "doc"
- **chapter//para** selects all descendants "para" of "chapter", child of the current node
- **//para** selects all "para" descendants of the current node

# XPath expression Examples

## Abbreviated Syntax (2)

- **.** selects the current node
- **.//para** selects "para" descendants of the current node
- **..** selects the parent of the current node
- **para[@type="warning"][5]** selects le 5th child "para" of the current node having a "type" attribute with value set to warning
- **para[5][@type="warning"]** selects le 5th child "para" of the current node having an attribute "type" with value set to "warning"
- **chapter[title="Introduction"]** selects the child "chapter" of the current node having one or more Childs "title" with a text content equals to 'Introduction'
- **employee[@secretary and @assistant]** selects all childs "employee" of the current node having both secretary and assistant attributes

# Exercise

- Given the following XML document

```
<AAA>
  <BBB/>
  <BCD/>
  <CCC>
    <BBB name='titi' />
    <BBB name='toto' />
  </CCC>
</AAA>
```

- Give the XPath expression for the selection of all <BBB> tags
- Give the XPath expression for the selection of the <BBB> tags having their "name" attribute value equals to “toto”

- Your feedback for XPath

# XML Document Transformations with XSLT

Reda.Bendraou@Lip6.fr

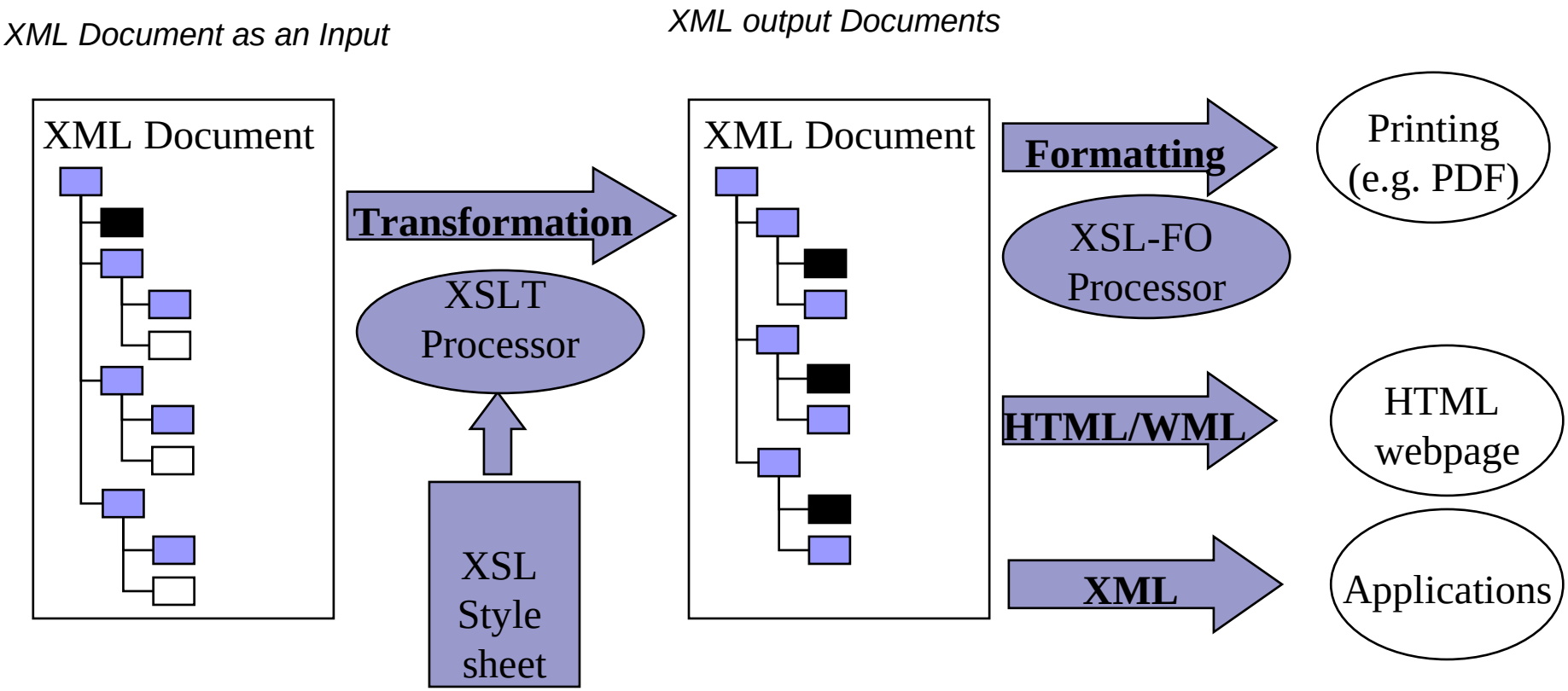
# XSLT ?

- XSLT stands for XSL Transformations
  - A Style Sheet = {Transformation Rules}
- An independent part of XSL (eXtensible Style Language)
- Used to transform an XML document into another XML document or HTML, XHTML documents
- **XML-based Syntax**
- Uses a **pattern-matching** approach
- Uses **Xpath** to navigate through elements and attributes in XML documents
- W3C Recommendation November 1999

# What you can do with XSLT

- Add/remove elements and attributes to or from the output file
- Rearrange and sort elements
- Perform tests and make decisions about which elements to hide and display
- And a lot more.

# How it works!




**XSLT transforms an XML source-tree into an XML result-tree**

# How to Declare  XSL/XSLT Style Sheets

- File extension :  .xsl or .xslt

- Declaration:

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

# Link the XSL Style Sheet to the XML Document

Add an XSL style sheet reference **within your XML document**

```
<? xml version = "1.0" ?>
<?xml-stylesheet  type="text/xsl"  href= "document.xsl"?>

<MyXMLDocumentRoot>
.....
.....
</MyXMLDocumentRoot>
```

*path to your XSLT file*

# XSLT rules

- An XSL style sheet consists of one or more set of rules that are called **Templates**
- Each **Templates** contains rules to apply when a specified node is matched:
  - Source Element (XML Tag) Identification thanks to XPath
  - Processing (applying the rule) ➔ a new XML or HTML (or both) as output

# XSLT rules : Example

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

# The <xsl:template> Element

- The **<xsl:template>** element is used to build templates

- The **match** attribute is used to associate a template with an XML element

- The value of the match attribute is an XPath expression
  - match="/" defines the whole document

# XSLT Template : Structure

<xsl:template

match = *pattern* *(XPath expression)*

name = *qname* *(In order to call this template from other Templates)*

priority= *number* *(Priority Rules )*

mode = *qname* *(In case you have different alternatives for the same match)* >

<!-- Content -->

</xsl:template>

# The <xsl:template> Element : Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
  <body>
     <h2>My CD Collection</h2>
     <table border="1">
            <tr bgcolor="#9acd32">
                    <th>Title</th> <th>Artist</th>
            </tr>
            <tr> <td>.</td> <td>.</td> </tr>
     </table>
  </body>
 </html>
 </xsl:template>
</xsl:stylesheet>
```

Result

# XSLT <xsl:value-of> Element

- Used to extract the value of a selected node add it to the output stream of the transformation
- Example

```
<xsl:template match="/">
<html>
 <body>
    <h2>My CD Collection</h2>
      <table border="1">
          <tr bgcolor="#9acd32">
              <th>Title</th> <th>Artist</th>
          </tr>
          <tr>
              <td><xsl:value-of select="catalog/cd/title"/></td>
              <td><xsl:value-of select="catalog/cd/artist"/></td>
          </tr>
          </table>
 </body>
</html>
</xsl:template>
```

Result

My CD Collection

| Title | Artist |
|---|---|
| Empire Burlesque | Bob Dylan |

Xpath expression

# XSLT <xsl:for-each> Element

- Allows you to do looping in XSLT
- Can be used to select every XML element of a specified node-set

```
<xsl:template match="/">
<html>
 <body>
    <h2>My CD Collection</h2>
      <table border="1">
          <tr bgcolor="#9acd32">
              <th>Title</th> <th>Artist</th>
          </tr>
          <xsl:for-each select="catalog/cd">
          <tr>
             <td><xsl:value-of select="title"/></td>
             <td><xsl:value-of select="artist"/></td>
          </tr>
          </xsl:for-each>
          </table>
 </body> </html>
</xsl:template>
```

Result

My CD Collection

| Title | Artist |
|---|---|
| Empire Burlesque | Bob Dylan |
| Hide your heart | Bonnie Tyler |
| Greatest Hits | Dolly Parton |
| Still got the blues | Gary More |
| Eros | Eros Ramazzotti |
| One night only | Bee Gees |
| Maggie May | Rod Stewart |
| Romanza | Andrea Bocelli |
| When a man loves a woman | Percy Sledge |
| Black angel | Savage Rose |
| For the good times | Kenny Rogers |
| Big Willie style | Will Smith |
| Tupelo Honey | Van Morrison |
| The very best of | Cat Stevens |
| Stop | Sam Brown |
| Bridge of Spies | T'Pau |
| Private Dancer | Tina Turner |
| Midt om natten | Kim Larsen |
| Pavarotti Gala Concert | Luciano Pavarotti |
| The dock of the bay | Otis Redding |
| Picture book | Simply Red |
| Red | The Communards |
| Unchain my heart | Joe Cocker |

# XSLT <xsl:sort> Element

- The <xsl:sort> element is used to sort the output
  - simply add an <xsl:sort> element inside the <xsl:for-each> element in the XSL file

Example

```
<xsl:for-each select="catalog/cd">
  <xsl:sort select="artist"/>
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
</xsl:for-each>
```

Result

My CD Collection

| Title | Artist |
|---|---|
| Romanza | Andrea Bocelli |
| One night only | Bee Gees |
| Empire Burlesque | Bob Dylan |
| Hide your heart | Bonnie Tyler |
| The very best of | Cat Stevens |
| Greatest Hits | Dolly Parton |
| Sylvias Mother | Dr.Hook |
| Eros | Eros Ramazzotti |
| Still got the blues | Gary Moore |
| Unchain my heart | Joe Cocker |
| Soulsville | Jorn Hoel |
| For the good times | Kenny Rogers |
| Midt om natten | Kim Larsen |
| Pavarotti Gala Concert | Luciano Pavarotti |
| 1999 Grammy Nominees | Many |
| The dock of the bay | Otis Redding |
| When a man loves a woman | Percy Sledge |
| Maggie May | Rod Stewart |
| Black angel | Savage Rose |
| Picture book | Simply Red |
| Red | The Communards |
| Private Dancer | Tina Turner |
| Tupelo Honey | Van Morrison |
| Big Willie style | Will Smith |

# XSLT <xsl:if> Element

- Used to put a conditional test against the content of the XML file
- **Syntax**

```
<xsl:if test="expression">
... ...some output if the expression is true.....
</xsl:if>
```

Example

```
<xsl:for-each select="catalog/cd">
 <xsl:if test="price &gt; 10">
   <tr>
     <td><xsl:value-of select="title"/></td>
     <td><xsl:value-of select="artist"/></td>
   </tr>
 </xsl:if>
</xsl:for-each>
```

Result

My CD Collection

| Title | Artist |
|---|---|
| Empire Burlesque | Bob Dylan |
| Still got the blues | Gary Moore |
| One night only | Bee Gees |
| Romanza | Andrea Bocelli |
| Black Angel | Savage Rose |
| 1999 Grammy Nominees | Many |

# XSLT <xsl:apply-templates> Element

- Applies a template to the current element or to the current element's child nodes

- If we add a **select** attribute to the **<xsl:apply-templates>** element => will process only the child element that matches the value of the attribute

**Syntax**

<xsl:apply-templates
select = *path* ➔ *Process nodes selected by path*
mode = *name* > ➔ *apply templates with mode name.*
….Content
</xsl:apply-templates>

- We can use the select attribute to specify the order in which the child nodes are processed

# XSLT <xsl:apply-templates> Element

```
<xsl:template match="/">
    <html>
        <body>
            <h2>My CD Collection</h2>
             <xsl:apply-templates/>
        </body>
    </html>
</xsl:template>

<xsl:template match="cd">
<p> <xsl:apply-templates select="title"/>
    <xsl:apply-templates select="artist"/> </p>
</xsl:template>

<xsl:template match="title">
    Title: <span style="color:#ff0000">
        <xsl:value-of select="."/></span> <br />
</xsl:template>

<xsl:template match="artist">
    Artist: <span style="color:#00ff00">
        <xsl:value-of select="."/></span> <br />
</xsl:template>
```

Result

**My CD Collection**

Title: Empire Burlesque
Artist: Bob Dylan

Title: Hide your heart
Artist: Bonnie Tyler

Title: Greatest Hits
Artist: Dolly Parton

Title: Still got the blues
Artist: Gary Moore

Title: Eros
Artist: Eros Ramazzotti

Title: One night only
Artist: Bee Gees

Title: Sylvias Mother
Artist: Dr.Hook

# XSLT <xsl:apply-templates> : Modes

- Modes : alternative processing.
  - declaration

```
<xsl:template match="Cd" mode="m1">
    ...
<xsl:template match="Cd" mode="m2">
```

  - Use

```
<xsl:apply-templates mode="m2">
```

# XSLT <xsl:call-template> Element

- The **<xsl:call-template>** element calls a named template
- A named template can't have a **match** attribute
- **How to call a Template**

  **<xsl:call-template name=*qname*>**

E.g.:

```
<xsl:template match="car">
        <xsl:call-template name="description"/>
</xsl:template>


<xsl:template name=" description ">
        .....
</xsl:template>
```

# XSLT <xsl:variable> Element

- The <xsl:variable> element is used to declare a local or global variable
  - The variable is global if it's declared as a top-level element, and local if it's declared within a template
  - Once you have set a variable's value, you cannot change or modify that value!

**Syntax**

```
<xsl:variable name="name" select="expression">
    <!-- Content:template -->
</xsl:variable>
```

- You can add a value to a variable by the content of the <xsl:variable> element OR by the select attribute!
- **Example**

```
<xsl:variable name="color" select= "@name" />
```

Or

```
<xsl:variable name="header">
        <tr> <th>Element</th> <th>Description</th> </tr>
</xsl:variable>
```

- **To Call** a Variable, **Use** the Variable name preceeded by the **$** signe.

*eg. <xsl:apply-templates select="categorie/logiciel{$num}">*

# XML elements creation

- Element Creation
  - <xsl:element name = *qname*>
- Attribute Creation
  - <xsl:attribute name = *qname*> *value*
- Texte Creation
  - <xsl:text > *texte*
- Processing Instruction Creation
  - <xsl:processing-instruction name=*qname*>
- Comments Creation
  - <xsl-comment> *text* </xsl-comment>

# Template Import

- Two mechanisms for combining style sheets

  - <xsl:include href=*uri-reference*/>
    Style sheet Copy/Paste

  - <xsl:import href=*uri-reference*/>
    Link into the referenced style sheet.
    - Has less priority in case of conflict with the current style sheet

# Other XSLT Functions

- Operations upon numbers
- Opérations upon strings
- Sending of message
- …

# XSLT Engines

- IE 6.0 (Microsoft)
- Xalang (Apache)
- Turbine (Apache)

# Exercise

```
<A>
    <B>
        <E>4</E>
        <E>3</E>
    </B>
    <B>
        <E>1</E>
    </B>
    <C>
        <F>a</F>
    </C>
    <E>2</E>
</A>
```

- xml1.xml

```
<A>
    <B>
        <E>4</E>
    </B>
    <B/>
    <C>
        <F>a</F>
    </C>
    <C>
        <F>b</F>
    </C>
</A>
```

xml2.xml

Give the XSLT style sheet used to generate xml2.xml from xml1.xml

# One possible solution:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match='/'>
      <xsl:apply-templates select='A'/>
</xsl:template>
<xsl:template match='A'>
   <A>
     <xsl:apply-templates select='B[1]'/>
    <B></B>
     <xsl:apply-templates select='C'/>
    </A>
</xsl:template>

<xsl:template match='B'>
        <B>
        <E> <xsl:value-of select='E'/> </E>
        </B>
</xsl:template>

<xsl:template match='C'>
        <C>
        <F> <xsl:value-of select='F'/> </F>
        </C>
        <C>
        <F>b</F>
        </C>
</xsl:template>
</xsl:stylesheet>
```

# Exercise

- Given the following XML document

```
<AAA>
  <BBB/>
  <BCD/>
  <CCC>
    <BBB nom='titi' />
    <DDD nom='toto' />
  </CCC>
</AAA>
```

- Propose an  XSLT that transforms the given document towards another one that contains only AAA, BBB, CCC tags

# XSLT Conclusion

- XML-based Syntax (simple)
- Approach: **Pattern/Matching**
- Client or Server Transformations
- Recommendation W3C November 1999

# XSL/FO: Basics

**XSL Specification**
*url:* http://www.w3.org/TR/xsl/

**Purpose**
- Formatting an XML document
- Result of a high-quality (equivalent to LaTeX text edition)

**Usage**
1. XML + XSLT => XML + XSL/FO -> printable/displayable format

**Status**
• XSL/FO a W3C recommendation since 2001
• XSL-FO does not work in usual browsers (you need viewers)
• Some XML editors as **Oxygen** provide support for XSL/FO
*url:* **http://xmlgraphics.apache.org/fop/ (Processur XSL/FO de Apache)**

# fo:root

**The root of a formatted document:**

**fo:root**

**A formatted document start by :**

```
<fo:root
   xmlns:fo="http://www.w3.org/1999/XSL/Format">
```

# fo:root

fo:root contains :

- One element : fo:layout-master-set
- 0 or 1 element : fo:declarations
- 1 or more element : fo:page-sequence

# A simple XSLT example with XSL-FO

```
<xsl:template match="page">
<fo:root>
    <fo:layout-master-set>
    <!-- Definition of a single master page. It is simple (no headers etc.) -->
        <fo:simple-page-master master-name="first" >
            <!-- required element body -->
            <fo:region-body/>
        </fo:simple-page-master>
    </fo:layout-master-set>
    <!-- Definition of a page sequence -->
    <fo:page-sequence master-reference="first">
        <fo:flow flow-name="xsl-region-body" font-size="14pt" line-height="14pt">
          <xsl:apply-templates/>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
</xsl:template>
```

# The usual « Hello World! » example

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format" >
    <fo:layout-master-set>
        <fo:simple-page-master master-name="LetterPage" page-width="8.5in“ page-height="11in" >
            <fo:region-body region-name="PageBody" margin="0.7in"/>
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="LetterPage">
        <fo:flow flow-name="PageBody">
            <fo:block>Hello World</fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
```

Result => Hello World

# fo:layout-master-set

fo:layout-master-set defines the layout of the document

Contains 1 or more elements:

- fo:simple-page-master

ou

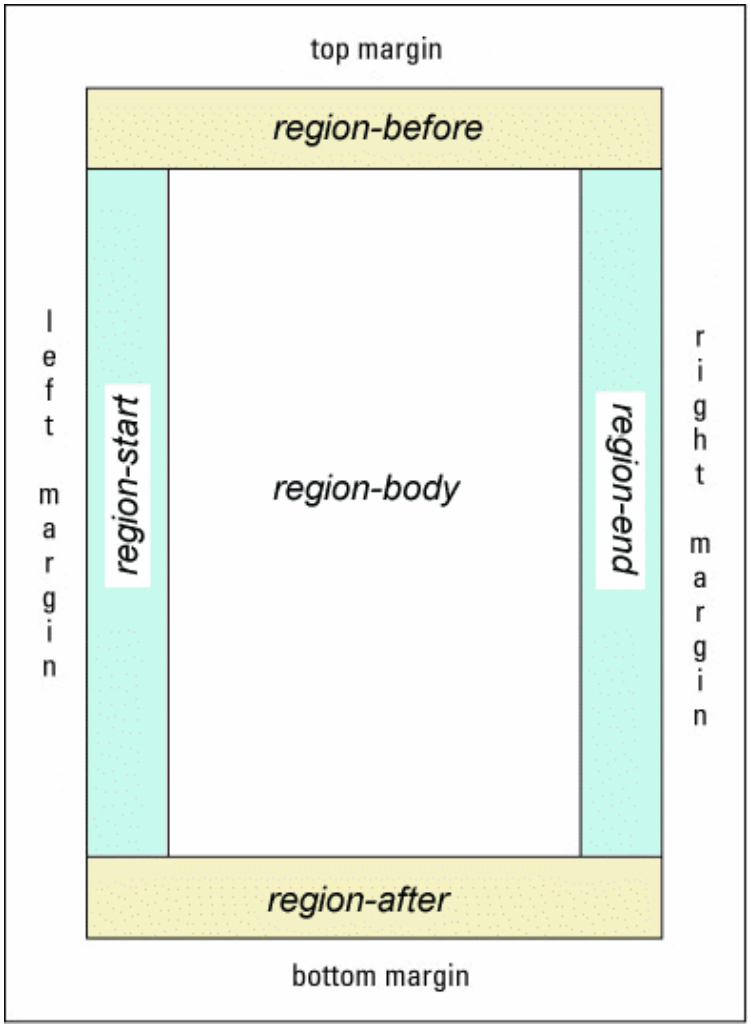
- fo:page-sequence-master

# fo:simple-page-master

Defines the layout of the pages

```
<fo:simple-page-master master-name="simple"
    page-height="29.7cm"
    page-width="21cm"
    margin-top="1cm"
    margin-bottom="2cm" margin-left="2.5cm" margin-
    right="2.5cm">
      <fo:region-body margin-top="3cm"/>
      <fo:region-before extent="3cm"/>
      <fo:region-after extent="1.5cm"/>
</fo:simple-page-master>
```

# Layout: the reference



# fo:page-sequence

fo:page-sequence defines the content of the pages

Contains :

0 or 1 element fo:title

0 or more element fo:static-content

1 element fo:flow

# fo:flow

fo:flow defines the data flow contained in the pages

Contains element of type block :

fo:block

fo:block-container

fo:table-and-caption

fo:table

fo:list-block

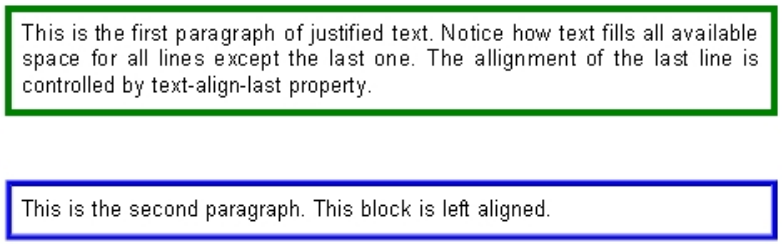
# fo:block

fo:block => paragraphe

Contains text

# Les paragraphes: exemple

```
<?xml version="1.0" encoding="utf-8"?><fo:root
    xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
    <fo:simple-page-master master-name="LetterPage"  page-width="6in" page-height="5in">
        <fo:region-body region-name="PageBody" margin="0.7in"/>
    </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="LetterPage">
    <fo:flow flow-name="PageBody" font-family="Arial" font-size="12pt" >
    <fo:block text-align="justify" space-after="0.5cm" border="0.5pt solid green" >  C'est le
        premier paragraphe du texte justifié. Remarquez comment le texte remplit tout l'espace
        disponible. La bordure environnante est de 0.5 points de large, couleur verte et pleine.
        Ce paragraphe a un espace-après égale à 0.5 centimètres.
    </fo:block>
    <fo:block text-align="justify" space-before="2cm" border="0.5pt dotted red" >
    C'est le deuxième paragraphe du texte justifié. Cette fois la bordure est pointillée et rouge.
        Ce paragraphe a un espace-avant égale à 2 centimètres.
    </fo:block>
    </fo:flow>
</fo:page-sequence>
</fo:root>
```



# XSL FO

- Offers large possibilities!
- Only 10% presented here ☺
- Feel free to look at the standard specification