

# XSLT

## MPRI 2.26.2: Web Data Management

---

Antoine Amarilli<sup>a</sup>

Wednesday, December 11th



---

<sup>a</sup>Based on slides from the Webdam book by Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, Pierre Senellart  
<https://webdam.inria.fr/Jorge/files/slxmlt.pdf>

## What is XSLT?

XSLT = a specialized language for transforming an XML document into another XML document.

Main principles:

- An XSLT program, or **stylesheet**, consists of rules, or **templates**.
- A template applies to a specific kind of node of the input document, and produces a fragment of the output document.
  - ▶ by creating **literal nodes**,
  - ▶ by **copying values** and **fragments** from the input document,
  - ▶ by **instantiating** (= calling) other templates.
- Execution model: initially, a template is applied to the **root node** of the input document  
⇒ this first template may initiate a traversal of the input document.

### Remark

An XSLT stylesheet is an XML document! XSLT element names are prefixed by (typically) `xsl:` that refers to the XSLT namespace.

# A Hello World! Stylesheet

```
<?xml version="1.0" encoding="utf-8"?>

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

  <xsl:output method="xml" encoding="utf-8" />

  <xsl:template match="/">
    <hello>world</hello>
  </xsl:template>

</xsl:stylesheet>
```

General structure of a stylesheet:

- A top-level `<xsl:stylesheet>` element
- Some **declarations** (all elements except `<xsl:template>` ones)
- Some **template rules**, in this case a template that applies to the root node.

## Invocation of an XSLT Stylesheet

An XSLT stylesheet may be invoked:

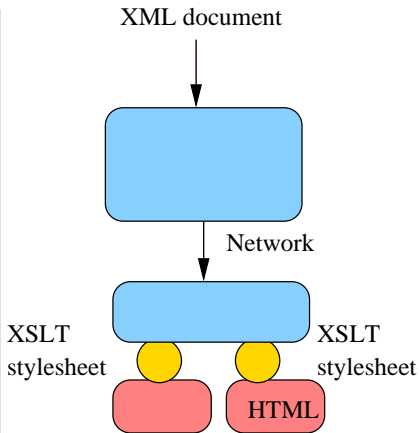
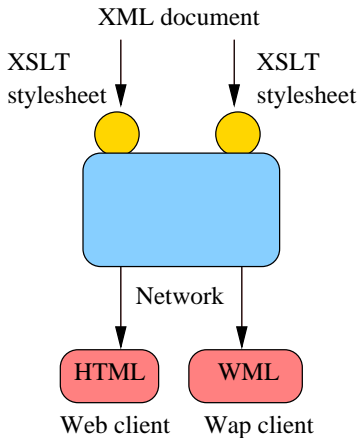
- **Programmatically**, through one of the various XSLT libraries.
- Through a **command line** interface.
- In a Web Publishing context, by including a styling processing instruction in the XML document

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

<doc>
  <titi />
</doc>
```

- ▶ the transformation can be processed on the **server side** by a CGI, PHP, ASP, JSP... script
- ▶ or on the **client side** through the XSLT engines integrated to most browsers.

# Web Publishing with XSLT



## The `<xsl:template>` Element

```
<xsl:template match="book">
  The book title is:
    <xsl:value-of select="title" />

  <h2>Authors list</h2>
  <ul>
    <xsl:apply-templates select="authors/name" />
  </ul>
</xsl:template>
```

A template consists of

**A pattern** an XPath expression (restricted) which determines the node to which the template applies.

The pattern is the value of the `match` attribute.

**A body** an XML fragment (well-formed!) which is inserted in the output document when the template is instantiated.

## XPath patterns in XSLT

The role of the XPath expression of the `match` attribute is quite specific: it describes the nodes which can be the target of a template instantiation. Those expressions are called **patterns**. They must comply to the following requirements

- **A pattern always denotes a node set.**

Example: `<xsl:template match='1'>` is incorrect.

- **It must be easy to decide whether a node is denoted or not by a pattern.**

Example: `<xsl:template match='preceding::*[12]'>` is meaningful, but quite difficult to evaluate.

### Patterns syntax

A pattern is a valid XPath expression which uses only the `child` and `@` axes, and the abbreviation `//`. Predicates are allowed.

## Pattern examples

Recall: a pattern is interpreted as the nodes to which a template applies.

- `<xsl:template match='B'>`  
applies to any **B** element.
- `<xsl:template match='A/B'>`  
applies to any **B** element, child of an **A** element.
- `<xsl:template match='@att1'>`  
applies to any **att1** attribute, whatever its parent element.
- `<xsl:template match='A//@att1'>`  
applies to any **att1** attribute, if its parent element is a descendant of an **A** element.

### General rule

Given an XML tree  $T$ , a pattern  $P$  **matches** a node  $N$  if there exists a node  $C$  (the **context node**) in  $T$  such that  $N \in P(T, C)$ .



## Content of a template body

Basically, the content of `<xsl:template>` may consist of:

- **Literal elements and text.**

Example: `<h2>Authors</h2>` . This creates in the output document an element `h2`, with a **Text** child node 'Authors'.

- **Values and elements from the input document.**

Example: `<xsl:value-of select='title' />` . This inserts in the output document a node set, result of the XPath expression `title`.

- **Call to other templates.**

Example: `<xsl:apply-templates select='authors' />` .  
Applies a template to each node in the node set result of the XPath expression `authors`.

### Remark

Only the basic of XSLT programming! Many advanced features (modes, priorities, loops and tests) beyond this core description.

## Instantiation of a `<xsl:template>`

### Main principles:

- **Literal elements** (those that don't belong to the XSLT namespace) and **text** are simply copied to the output document.
- **Context node**: A template is always instantiated in the context of a node from the input document.
- **XPath expressions**: all the (relative) XPath expression found in the template are evaluated with respect to the context node (an exception: `<xsl:for-each>` ).
- **Calls with `xsl:apply-templates`** : find and instantiate a template for each node selected by the XPath expression `select` .
- **Template call substitution**: any call to other templates is eventually replaced by the instantiation of these templates.

## The `xsl:apply-templates` element

```
<xsl:apply-templates select="authors/name" />
```

**select** an XPath expression which, if relative, is interpreted with respect to the context node,

**Note:** the default value is `child::node()`, i.e., select all the children of the context node

**mode** a label which can be used to specify which kind of template is required.

## The `xsl:apply-templates` mechanism

```
<xsl:template match="book">
  <ul><xsl:apply-templates
    select="authors/name" /></ul>
</xsl:template>
```

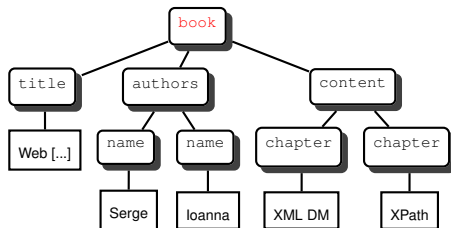
```
<xsl:template match="name">
  <li><xsl:value-of select="." /></li>
</xsl:template>
```

```
<book>
...
<authors>
  <name>Serge</name>
  <name>Ioana</name>
</authors>
</book>
```



```
<ul>
  <li>Serge</li>
  <li>Ioana</li>
</ul>
```

# Combined templates instantiation



```

<xsl:template match="book">
Title: <xsl:value-of
      select="title" />

<h2>Authors</h2>
  <ul><xsl:apply-templates
    select="authors/name" />
  </ul>
</xsl:template>
  
```

→

```

Title: Web[...]

<h2>Authors</h2>
  <ul>
    <li>Serge</li>
    <li>Ioanna</li>
  </ul>
  
```

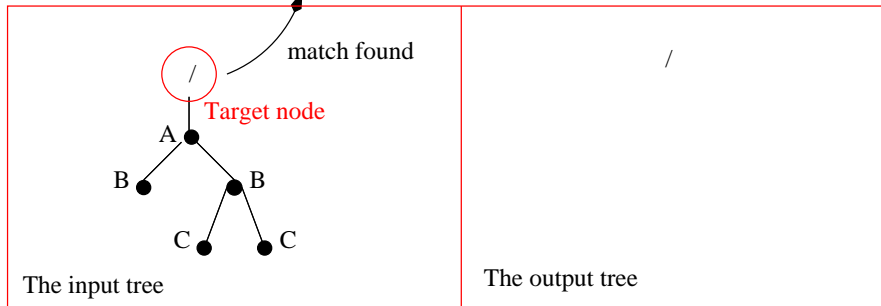
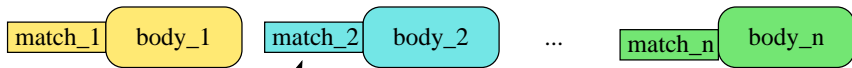
## The execution model of XSLT

An XSLT stylesheet consists of a set of templates. The transformation of an input document  $I$  proceeds as follows:

- 1 The engine considers the **root node**  $R$  of  $I$ , and selects the template that applies to  $R$ .
- 2 The template body is copied in the output document  $O$ .
- 3 Next, the engine considers all the `<xsl:apply-templates>` that have been copied in  $O$ , and evaluate the **select** XPath expression, taking  $R$  as context node.
- 4 For each node result of the XPath evaluation, a template is selected, and its body replaces the `<xsl:apply-templates>` call.
- 5 The process iterates, as new `<xsl:apply-templates>` may have been inserted in  $O$ .
- 6 The transform terminates when  $O$  is free of **xsl:** instructions.

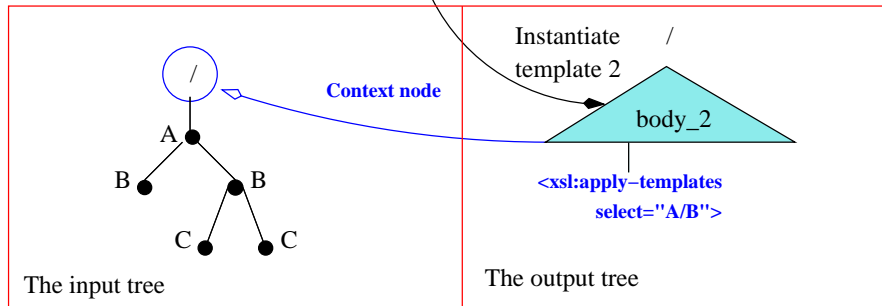
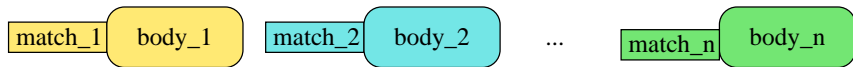
# The execution model: illustration

The templates list



# The execution model: illustration

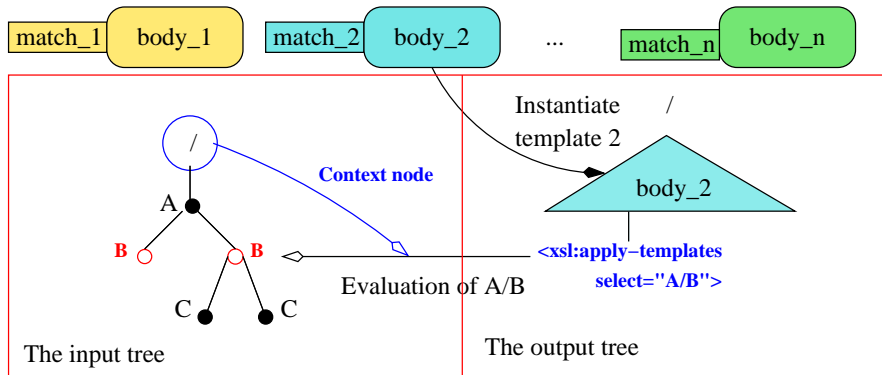
## The templates list





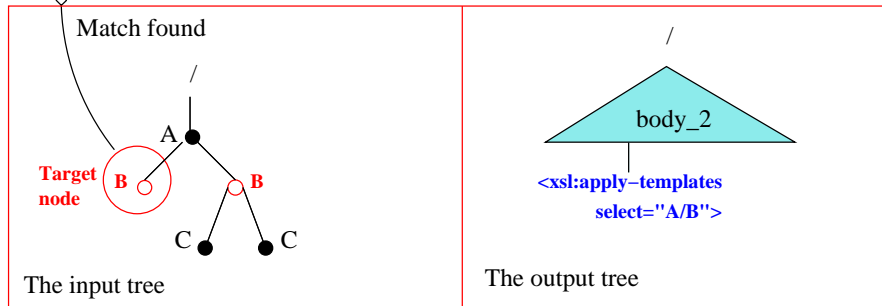
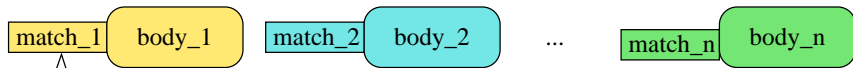
## The execution model: illustration

## The templates list



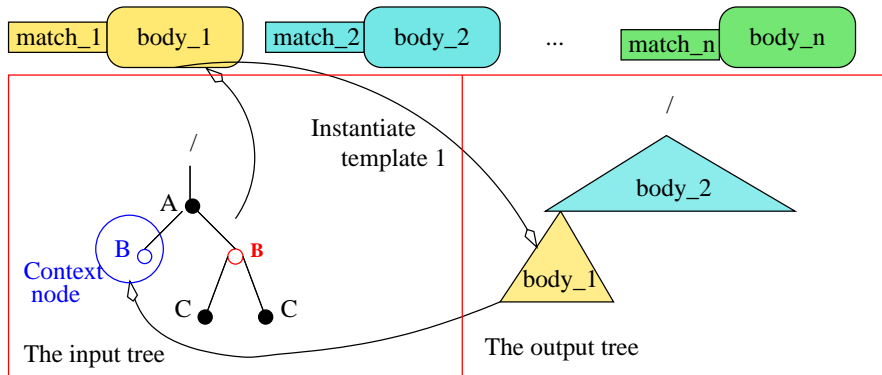
# The execution model: illustration

## The templates list



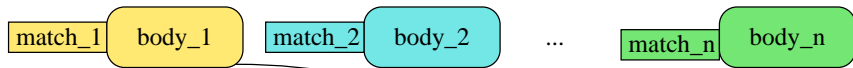
# The execution model: illustration

## The templates list

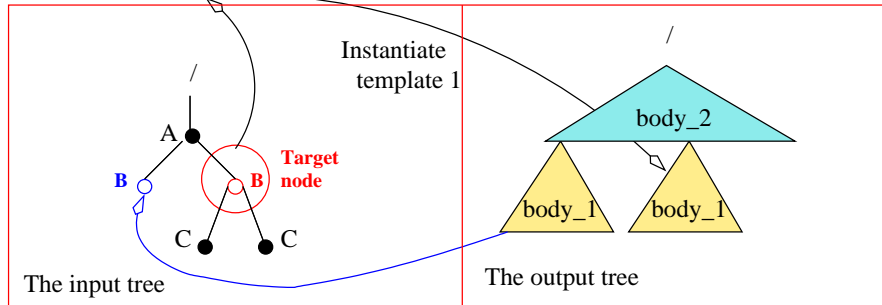


# The execution model: illustration

The templates list



Instantiate  
template 1

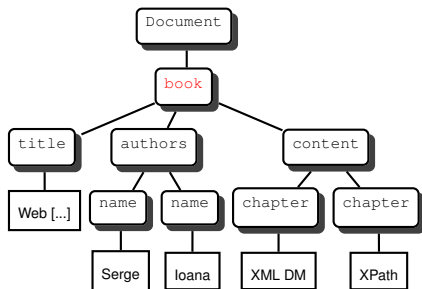


# The input document in serialized and tree forms

```

<?xml version="1.0"
  encoding="utf-8"?>
<book>
  <title>Web [...]</title>
  <authors>
    <name>Serge</name>
    <name>Ioana</name>
  </authors>
  <content>
    <chapter id="1">
      XML data model
    </chapter>
    <chapter id="2">
      XPath
    </chapter>
  </content>
</book>

```



## The XSLT template that matches the root node

```
<xsl:template match="/">
  <html>
    <head>
      <title>
        <xsl:value-of select="/book/title"/>
      </title>
    </head>
    <body bgcolor="white">

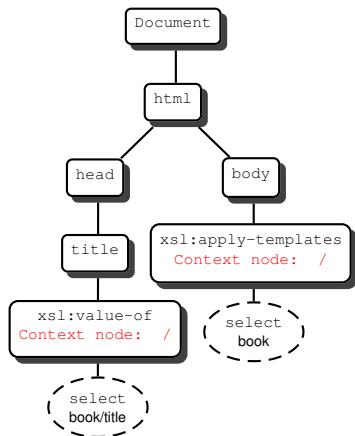
      <xsl:apply-templates select="book"/>

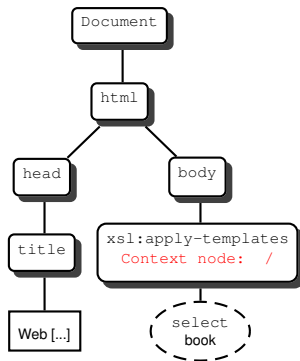
    </body>
  </html>
</xsl:template>
```

### Remark

Typical of Web publishing templates.

# The output document after instantiating the template



The output document after evaluation of `<xsl:value-of>`



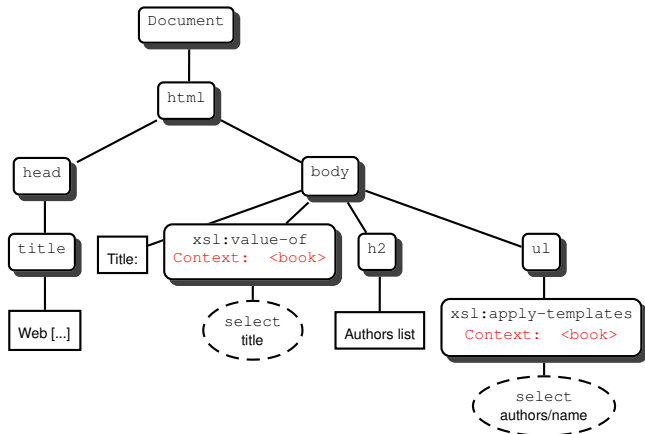
## The remaining templates

```
<xsl:template match="book">
  Title:
    <xsl:value-of select="title" />

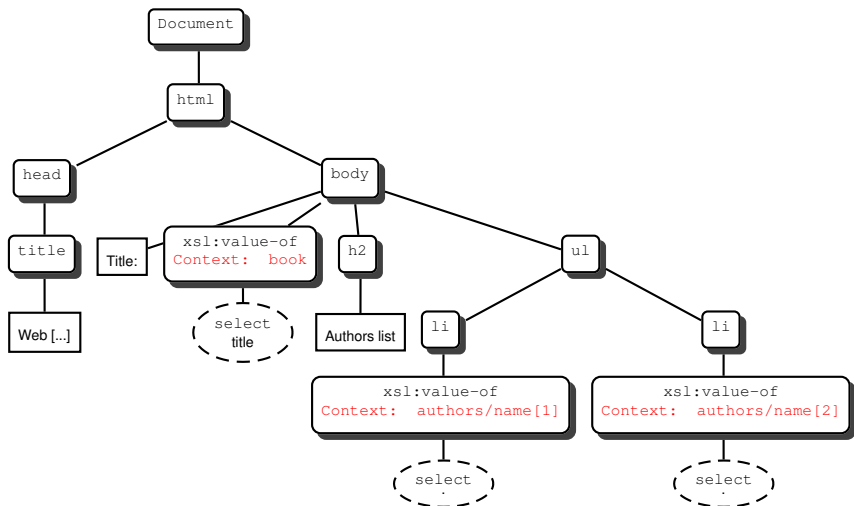
  <h2>Authors list</h2>
  <ul>
    <xsl:apply-templates select="authors/name" />
  </ul>
</xsl:template>

<xsl:template match="authors/name">
  <li><xsl:value-of select="."/ ></li>
</xsl:template>
```

The `<book>` element is selected  $\Rightarrow$  the `book` template is instantiated



The `authors/name` template is instantiated twice, one for each author



# The final output document

