XSLT, a *functional* approach to XML document transformation

Ludovic Stordeur

June 7, 2011

Outline

Outline

Context

1. Performing my first Open-MX release;
2. Tarball generation, GForge pushing, ...

Context

1. Performing my first Open-MX release;
2. Tarball generation, GForge pushing, ...
3. and finally ... website update !
   - a HTML handwritten code;
   - Need to add the Open-MX release news at several places;
     - On the *Releases* page;
     - On the *News* page;
     - On the main page in the "hot" news section.

A lot of code duplication !

Need for a more maintainable solution.

Outline

Goals

Idea

Split “cold” part and “hot” part of the website.

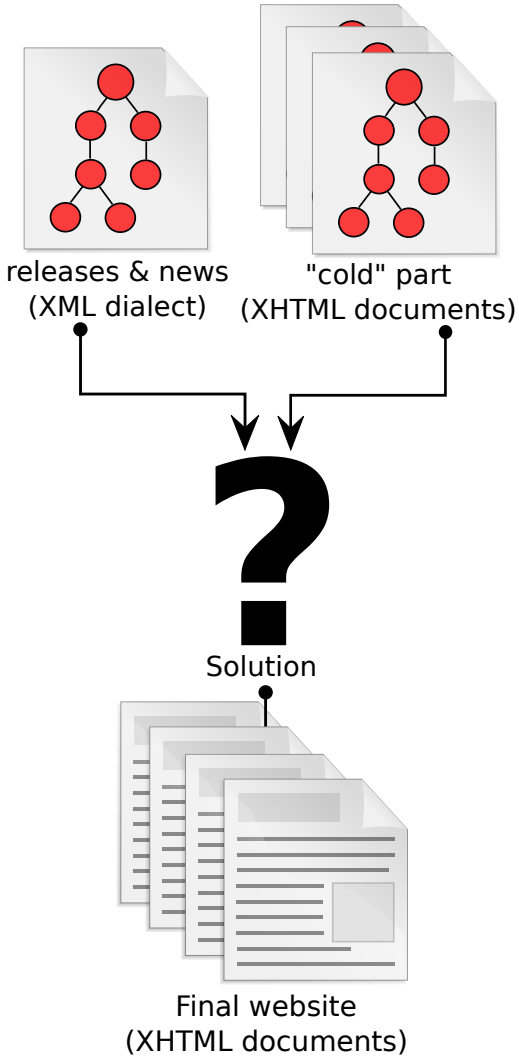
1. Inputs

- “cold” part: several XHTML documents;
- “hot” part: our own XML dialect;
  - Releases;
  - News.

2. Processing phase

3. Output

The final website: a collection of XHTML documents.



Existing approaches

Based on a “classical” XML parsing . . .

- SAX implementations;
- Expat;
- libxml2;
- . . .

. . . using general purpose programming languages

- PHP;
- Python;
- Java;
- C/C++;
- . . .

Existing approaches

Based on a “classical” XML parsing . . .

- SAX implementations;
- Expat;
- libxml2;
- . . .

. . . using general purpose programming languages

- PHP;
- Python;
- Java;
- C/C++;
- . . .

Error-prone solutions !

XSLT: a different point of view

XSLT: Extensible Stylesheet Language Transformations

- XML-based language;
- Higher level of abstraction (declarative model);
- Functional paradigm;
- XML dedicated processing.

Outline

History

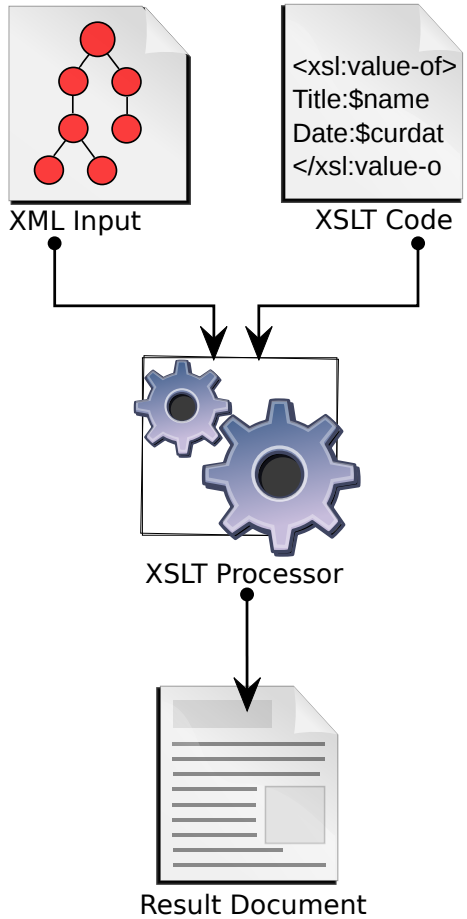
- Developed by the *World Wide Web Consortium* (W3C);
- Originally part of the *Extensible Stylesheet Language* (XSL) project;
  - XSL Formatting Objects;
  - XPath.
- Development effort initiated in 1998;
- XSLT 1.0: first recommandation on 16 November 1999 (James Clark);
- XSLT 2.0: 23 January 2007 (Michael Kay);
- Direct successor of DSSSL (SGML processing).

Outline

Overview

Involved entities

- XML inputs (generally one document);
- XSLT code inputs (generally one module);
- XSLT processor;
- Outputs (generally one document);
  - XML (XHTML);
  - HTML;
  - Plain text;
  - Binary.



source: *Wikipedia*

Philosophy

- Declaration of a collection of rules;
- A rule define how to handle a given input node;
  - Using pattern matching through *XPath*.

Outline

XML description of the releases

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

<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>

    <release ver="1.3.4" date="2010/12/16"
             md5="4565e6516610a2776e4a926529885f82"
             href="28070/open-mx-1.3.4.tar.gz"/>

    <release ver="1.3.3" date="2010/10/15"
             md5="253b46bc0151d1c6f3ea347006dab85b"
             href="27637/open-mx-1.3.3.tar.gz"/>

    ...
  </releases>
</open-mx>
```

Wished output

```
...
<tr>
  <td> <a
     href="http://gforge.inria.fr/frs/download.php/28070/open-mx-1.3.4.tar.gz">
     Open-MX 1.3.4
  </a> </td>
  <td>2010/12/16</td>
  <td>4565e6516610a2776e4a926529885f82</td>
 </tr>
 <tr>
  <td> <a
     href="http://gforge.inria.fr/frs/download.php/27637/open-mx-1.3.3.tar.gz">
     Open-MX 1.3.3
  </a> </td>
  <td>2010/12/16</td>
  <td>253b46bc0151d1c6f3ea347006dab85b</td>
</tr>
...
```

The minimal XSLT skeleton

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

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

 <xsl:output method="xml"
             version="1.0"
             encoding="utf-8"
             indent="yes"
             doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"
             doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"/>

 <!-- XSLT code here -->

</xsl:stylesheet>
```

Introduction to templates

```
<xsl:stylesheet>

  <xsl:output ...>

  <xsl:template match="/open-mx/releases/release">
    <tr>
      <td>
        <a>
          <xsl:attribute name="href">
            <xsl:value-of select="../base_url"/>/<xsl:value-of select="@href"/>
          </xsl:attribute>
          Open-MX <xsl:value-of select="@ver"/>
        </a>
      </td>
      <td> <xsl:value-of select="@date"/> </td>
      <td> <xsl:value-of select="@md5"/> </td>
    </tr>
  </xsl:template>

</xsl:stylesheet>
```

Core concept

1. The XSLT processor permanently have a context in the XML input;
2. Context = Node.

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. **Starting context:** the root node;

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. **Starting context:** the root node;
2. Looking for a matching rule . . .

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. **Starting context:** the root node;
2. Looking for a matching rule . . . Failed !
3. **Context change:** go down to the child node.

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. Looking for a matching rule . . .

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. Looking for a matching rule . . . Failed !
2. **Context change:** go down to the child node . . .

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. Which ones ?

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. Which ones ?
2. all of them but in a completely specified order;
3. **New definition:** Context = Nodeset.

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. Matching!
2. Executing the template code;

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

1. Matching!
2. Executing the template code;
3. The next child will also match the same template.

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Core concept

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release"> --> XPath expression
  <tr>
    <td>
...
```

XPath

Goal

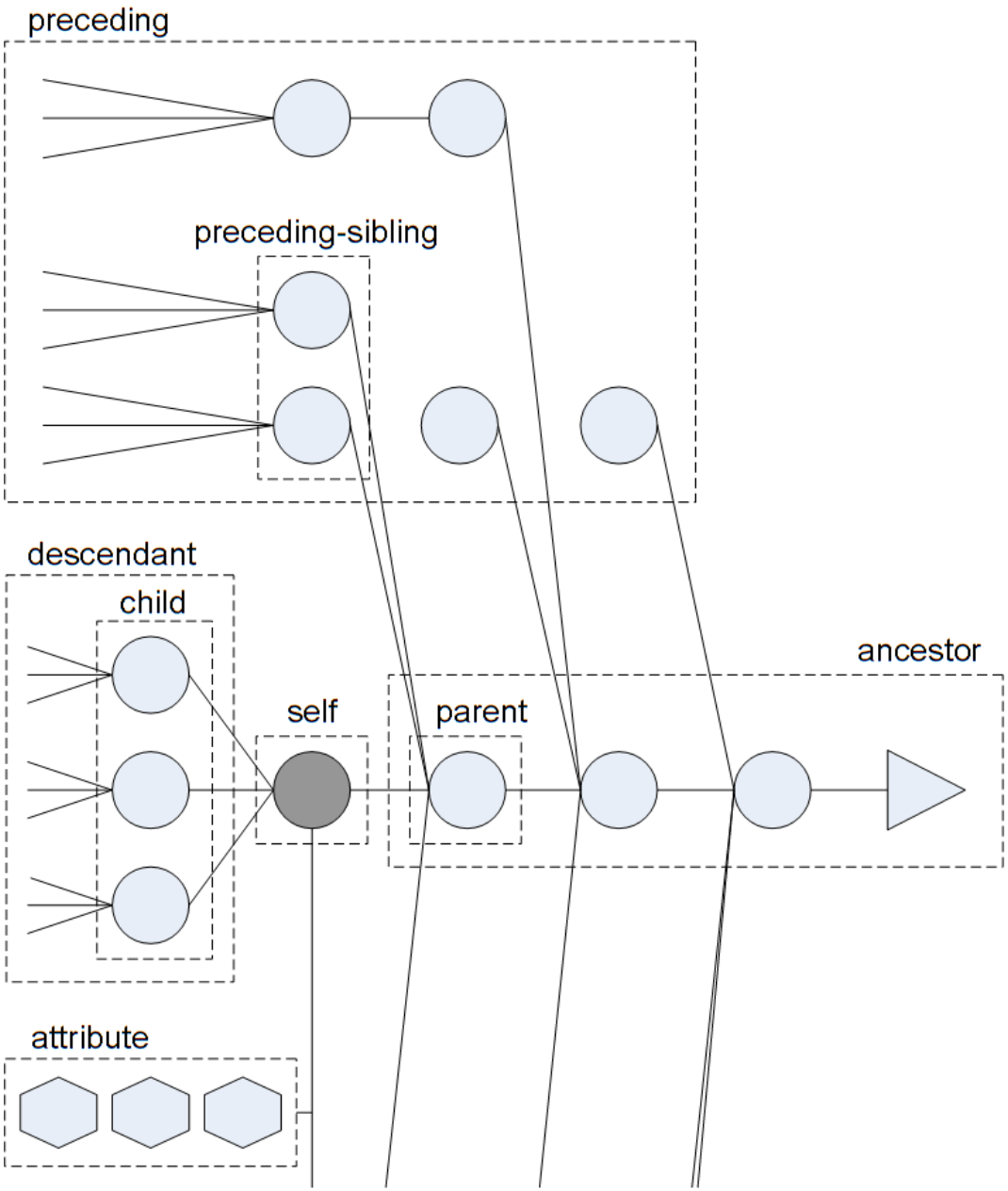
1. Addressing a part of the XML tree;
2. Short syntax;
3. Powerful language.

Recommendation

XPath 1.0 16 November 1999;

XPath 2.0 23 January 2007.

XML tree


preceding
preceding-sibling
descendant
child
ancestor
self
parent
attribute

The XPath concept: XPath path

Definition

A step sequence:

```
[/]step1/step2/.../stepN
```

- A step is evaluated in a nodeset context;
- A step evaluation provides the context for the next step;
- The last step evaluation returns the final nodeset.

Initial context

Absolute path the root node;

Relative path the current node (current context in the XML document).

Path union

Using the operator: |

```
/foo | /bar/baz
```

XPath steps

Definition

```
axe::filter[predicate]*
```

axe	scanning direction;
filter	first-level filtering;
predicates	second-level filtering.

Some filters

QName	filter nodes having the name *QName*;
*	nodes of the *main* type;
node()	any node;
text()	text nodes.

XPath syntactic sugars

`child::` can be omitted

```
child::releases/child::release
        --> releases/release
```

`attribute::` can be replaced by `@`

```
child::release[attribute::ver="1.3.3"]/attribute::md5
        --> release[@ver="1.3.4"]/@md5
```

`descendant-or-self::node()` can be replaced by `//`

```
descendant-or-self::node()/@md5
       --> //@md5
```

Template application

```
<xsl:stylesheet>
  ...
  <xsl:template match="/open-mx/releases/release">
    <tr>
      <td>
        <a>
         <xsl:attribute name="href">
           <xsl:value-of select="../base_url"/>/<xsl:value-of select="@href"/>
         </xsl:attribute>
         Open-MX <xsl:value-of select="@ver"/>
        </a>
      </td>
      <td> <xsl:value-of select="@date"/> </td>
      <td> <xsl:value-of select="@md5"/> </td>
    </tr>
  </xsl:template>
  ...
</xsl:stylesheet>
```

Template application

- **Non XSLT elements:** inlined in the output document.

```
<xsl:stylesheet>
  ...
  <xsl:template match="/open-mx/releases/release">
    <tr>
      <td>
        <a>
         <xsl:attribute name="href">
           <xsl:value-of select="../base_url"/>/<xsl:value-of select="@href"/>
         </xsl:attribute>
         Open-MX <xsl:value-of select="@ver"/>
        </a>
      </td>
      <td> <xsl:value-of select="@date"/> </td>
      <td> <xsl:value-of select="@md5"/> </td>
    </tr>
  </xsl:template>
  ...
</xsl:stylesheet>
```

Template application

xsl:attribute

- XSLT function;
- Create a computed attribute in the output document.

```
<xsl:stylesheet>
  ...
  <xsl:template match="/open-mx/releases/release">
    <tr>
      <td>
        <a>
         <xsl:attribute name="href">
           <xsl:value-of select="../base_url"/>/<xsl:value-of select="@href"/>
         </xsl:attribute>
         Open-MX <xsl:value-of select="@ver"/>
        </a>
      </td>
      <td> <xsl:value-of select="@date"/> </td>
      <td> <xsl:value-of select="@md5"/> </td>
    </tr>
  </xsl:template>
  ...
</xsl:stylesheet>
```

Template application

xsl:value-of

- Extract a node value.

```
<xsl:stylesheet>
  ...
  <xsl:template match="/open-mx/releases/release">
    <tr>
      <td>
        <a>
         <xsl:attribute name="href">
           <xsl:value-of select="../base_url"/>/<xsl:value-of select="@href"/>
         </xsl:attribute>
         Open-MX <xsl:value-of select="@ver"/>
        </a>
      </td>
      <td> <xsl:value-of select="@date"/> </td>
      <td> <xsl:value-of select="@md5"/> </td>
    </tr>
  </xsl:template>
  ...
</xsl:stylesheet>
```

Some XSLT functions

`xsl:element` Create a computed **element** node;

`xsl:text` Create a computed text node;

`xsl:comment` Create a computed comment node;

`xsl:if` Conditional execution;

`xsl:choose` *aka* "switch. . . case";

`xsl:for-each` Iteration over a nodeset.

`...` and a lot more !

Force a new context (xsl:apply-templates)

Goal

- We just want to generate the release “1.3.4”;
- **Idea:** manually build the next context.

XML input:

```
<open-mx>
  <releases> <-- Current context
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Force a new context (xsl:apply-templates)

Solution: apply-templates

- Create the next context, using an XPath expression;
- **Push** it as the new context.

XML input:

```
<open-mx>
  <releases> <-- Current context
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases">
  <xsl:apply-templates select="release[@ver='1.3.4']">
</xsl:template>

<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Force a new context (xsl:apply-templates)

Solution: apply-templates

- Create the next context, using an XPath expression;
- **Push** it as the new context.

XML input:

```
<open-mx>
  <releases> <-- Stacked context
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ... <-- New current context
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases">
  <xsl:apply-templates select="release[@ver='1.3.4']">
</xsl:template>

<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Default templates

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...
```

Default templates

- Default "context pushing" is in fact performed using an implicit template;
- **Template priority:** the first one matches !

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...

<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

Default templates

- Default "context pushing" is in fact performed using an implicit template;
- **Template priority:** the first one matches !

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...

<xsl:template match="*|/">
  <xsl:apply-templates/>
</xsl:template>
```

Default templates

- Default "context pushing" is in fact performed using an implicit template;
- **Template priority:** the first one matches !

XML input:

```
<open-mx>
  <releases>
    <base_url>http://gforge.inria.fr/frs/download.php</base_url>
    <release ver="1.3.4" ...
    <release ver="1.3.3" ...
...
```

XSLT code:

```
<xsl:template match="/open-mx/releases/release">
  <tr>
    <td>
...

<xsl:template match="*|/">
  <xsl:apply-templates select=node()/>
</xsl:template>
```

Outline

Using a standalone program (aka "Server side")

Concept: two-step cycle

1. Editing XML and/or XSLT documents;
2. Generation by manually calling the XSLT processor.

Syntax example

```
$ xsltproc -o baz.xhtml foo.xsl bar.xml
```

Some XSLT processors

Xalan *Apache fundation*. API C, Java, Perl. XSLT 1.0;

libxslt *GNOME project*. API C. XSLT 1.0;

Saxon *Michael Kay*. API Java, .NET. XSLT 1.0 & 2.0;

...

Using a web browser (aka "Client side")

Concept: "on-the-fly" processing

- No longer processing phase on the server side;
- The web browser fetches the XSLT code when requesting an XML resource and performs the process on its side.

Additional annotation in the XML document

Indicate to the browser where is the XSLT stylesheet bound to the document:

```
<?xml-stylesheet href="example2.xsl" type="text/xsl" ?>
```

Limitations

- Supported by most popular web browsers . . . but often partially;
- Less flexibility than the "Server side" approach.

Outline

Conclusion

The essential

- A XML-dedicated manipulation language;
- A very powerful language (nearly everything is possible !);
- Perfectly suited for web domain . . . but not only . . .

Some drawbacks

- Support lacking especially in web browsers;
- XSLT 2.0 not yet widely supported (and so used);
- Language verbosity.

Thanks !
Any questions ?