

Processing “Computed” Texts

Jean-Michel HUFFLEN

LIFC — University of Franche-Comté

GUIT, 17th October 2009

Contents

What are “computed” texts?

Why to use XML?

Generating T_EX-like texts

Generating XSL-FO texts

Using ConT_EXt MkIV

Conclusion

TEX & Co.

Usually process texts typed by authors.

TEX & Co.

Usually process texts typed by authors.

But some texts may be extracted from a larger structure.

TEX & Co.

Usually process texts typed by authors.

But some texts may be extracted from a larger structure.

Example: ds.xml, a list of stories available as *pulps* and *pocket books*.

TEX & Co.

Usually process texts typed by authors.

But some texts may be extracted from a larger structure.

Example: ds.xml, a list of stories available as *pulps* and *pocket books*.

Very simple version of many actual examples.

Examples

Available at:

`http://lifc.univ-fcomte.fr/home/~jmhufflen/
texts/git-2009/`

Doing it in $(\mathbb{A})\text{T}_E\text{X}$?

Theoretically possible, but very tedious in practice.

Doing it in $(\mathbb{A})\text{T}_E\text{X}$?

Theoretically possible, but very tedious in practice.

T_EX : not suitable for neither handling data bases,

Doing it in $(\mathbb{A})\text{T}_E\text{X}$?

Theoretically possible, but very tedious in practice.

T_EX : not suitable for neither handling data bases,
nor functionalities related to programming: e.g.,
sorting.

Doing it in (\LaTeX) T_EX?

Theoretically possible, but very tedious in practice.

T_EX: not suitable for neither handling data bases, nor functionalities related to programming: e.g., sorting.

Complicated markup, complicated definitions.

XML

Structured texts, like trees.

XML

Structured texts, like trees.

Data bases.

XSLT

Now widely used.

XSLT

Now widely used.

This operation is actually a transformation of some information.

XSLT

Now widely used.

This operation is actually a transformation of some information.

The new version (2.0) allows *character maps* \implies (L^A)T_EX's special characters processed more easily.

XSLT

Now widely used.

This operation is actually a transformation of some information.

The new version (2.0) allows *character maps* \implies (L^A)T_EX's special characters processed more easily.

(Example.)

XSLT: the better choice?

No static checking except if you derive XML texts.

XSLT: the better choice?

No static checking except if you derive XML texts.

Balanced braces.

XSLT: the better choice?

No static checking except if you derive XML texts.

Balanced braces.

Balanced environments for L^AT_EX:

```
\begin{something} . . . \end{something}
```

XSLT: the better choice? (con'd)

Such test would be difficult to implement about texts processed by ConT_EXt:

`\startsomething ... \stopsomething`

(e.g., `\starttext ... \stoptext`)

XSLT: the better choice? (con'd)

Such test would be difficult to implement about texts processed by ConT_EXt:

`\startsomething ... \stopsomething`

(e.g., `\starttext ... \stoptext`)

Very partially done in `nbst` \iff `latex` mode.

XQuery

Less verbose.

XQuery

Less verbose.

Programming by *templates*, more than *applicative* programming.

XQuery

Less verbose.

Programming by *templates*, more than *applicative* programming.

(Example.)

XQuery (con'd)

Suitable for simple examples, but with the same drawbacks about static checking.

XQuery (con'd)

Suitable for simple examples, but with the same drawbacks about static checking.

Many standard features in XSLT—e.g., character maps—are implementation-dependent in XQuery.

An 'actual' programming language

DSSSL was used for SGML texts, but might be suitable for XML texts, especially if many features are related to 'pure' programming.

An ‘actual’ programming language

DSSSL was used for SGML texts, but might be suitable for XML texts, especially if many features are related to ‘pure’ programming.

$\text{T}_{\text{E}}\text{X}$ source texts are not directly specified, only *constructs* a DSSSL processor translates to $\text{T}_{\text{E}}\text{X}$.

An ‘actual’ programming language

DSSSL was used for SGML texts, but might be suitable for XML texts, especially if many features are related to ‘pure’ programming.

$\text{T}_{\text{E}}\text{X}$ source texts are not directly specified, only *constructs* a DSSSL processor translates to $\text{T}_{\text{E}}\text{X}$.

(Example.)

Generating xml-like texts

$\text{XML} \xrightarrow{\text{XSLT}} \text{XSL-FO}$

(Example.)

\LaTeX users can easily learn XSL-FO, but it is another language.

Generating xml-like texts

$\text{XML} \xrightarrow{\text{XSLT}} \text{XSL-FO}$

(Example.)

\LaTeX users can easily learn XSL-FO, but it is another language.

FO processors are almost complete, but in progress.

LuaTeX

Tasks related to ‘pure’ programming are *delegated* to external functions written using Lua.

LuaT_EX

Tasks related to ‘pure’ programming are *delegated* to external functions written using Lua.

ConT_EXt MkIV allows XML texts to be processed,

LuaT_EX

Tasks related to ‘pure’ programming are *delegated* to external functions written using Lua.

ConT_EXt MkIV allows XML texts to be processed, but has not reached stable state yet;

LuaT_EX

Tasks related to ‘pure’ programming are *delegated* to external functions written using Lua.

ConT_EXt MkIV allows XML texts to be processed, but has not reached stable state yet;

it uses XPath-like expressions, but not identical to ‘pure’ XPath’s.

Point of view

Simple transformation \implies XQuery.

Point of view

Simple transformation \implies XQuery.

More ambitious one \implies XSLT.

Point of view

Simple transformation \implies XQuery.

More ambitious one \implies XSLT.

Keep in touch with FO's processors' progress.

Point of view

Simple transformation \implies XQuery.

More ambitious one \implies XSLT.

Keep in touch with FO's processors' progress.

Scrutinise ConT_EXt MKIV's development, ask his team for more development.