User's manual for

Writer$\vec{2}$LaTeX

Writer2LaTeX, Writer2BibTeX and Writer2xhtml

version 0.3.2

Table of Contents

1 Introduction

1.1 What is Writer2LaTeX?

Writer2LaTeX is a utility to convert OpenOffice.org (or StarOffice 6/7) Writer documents – in particular documents containing formulas – into other formats, Actually it is 3 converters in one:

- **Writer2LaTeX** converts to LaTeX 2e.
- **Writer2BibTeX** extracts bibliographic data from a document and converts it to BibTeX format.
- **Writer2xhtml** converts to XHTML 1.0 strict or XHTML 1.1 + MathML 2.0, using CSS2 to convert style information.

You can use Writer2LaTeX

- ...as a command line utility, independent of OpenOffice.org/StarOffice.
- ...as an export filter for OpenOffice.org 1.1 or StarOffice 7.
- ...from another Java program.

This user manual will explain how to install and use Writer2LaTeX.

Writer2LaTeX is a Java application, and thus should work on any platform that supports Java. You need Sun's Java 2 Virtual Machine (Runtime Environment), **version 1.4**. You can download this from `http://java.sun.com/getjava/download.html`. AFAIK Writer2LaTeX doesn't run (unmodified) under any other Java interpreter.

Note: In this manual OOo is used as an abbreviation of OpenOffice.org/StarOffice.

2 Installation

2.1 How to install Writer2LaTeX for command line usage

Writer2LaTeX can work as a standalone command line utility (that is without OOo).

Installation for Microsoft Windows

To install Writer2LaTeX under Microsoft Windows follow these instructions:

1. Unzip `writer2latex032.zip` into some directory. This will create a subdirectory `writer2latex`.

2. Add this directory to your PATH environment variable.

3. Open the file `w2l.bat` with a text editor and replace the path at the top of the file with the full path to Writer2LaTeX, for example

   ```
   set W2LPATH="c:\program files\writer2latex"
   ```

   At a command line type `java -version` to verify that the Java executable is in your path. If this is not the case or you have several Java versions installed you should edit the next line to contain the full path to the Java executable, eg.

   ```
   set JAVAEXE="C:\j2sdk1.4.0_01\bin\java’’
   ```

To use the LaTeX files generated by Writer2LaTeX, you will need the LaTeX package `writer.sty`, see section 3.1.

Installation for Unix and friends

1. Unzip `writer2latex032.zip` into some directory. This will create a subdirectory `writer2latex`.

2. Add this directory to your PATH environment variable or create a symbolic link to the file `w2l` from some directory in yout PATH.

3. Open the fle `w2l` with a text editor and replace the path at the top of the file with the full path to Writer2LaTeX, eg.

   ```
   W2LPATH="/home/username/writer2latex"
   ```

   Open a command shell and type `java -version` to verify that the Java executable is in your path. If this is not the case or you have several Java versions installed you should edit the next line to contain the full path to the Java executable, ie.

   ```
   set JAVAEXE="/path/to/java/executable/’’
   ```

4. Add execute permissions to `w2l` as follows:

   ```
   chmod +x w2l
   ```

To use the LaTeX files generated by Writer2LaTeX, you will need the LaTeX package `writer.sty`, see section 3.1.

2.2 How to install Writer2LaTeX as an export filter

Writer2LaTeX can work as an export filter for OOo Writer. This requires OpenOffice.org 1.1 or StarOffice 7. It does *not* work with OpenOffice.org 1.0 or StarOffice 6.0.

The following instructions covers all operating systems.

Uninstalling previous versions of Writer2LaTeX

If you have installed a previous version of Writer2LaTeX you will have to undo the changes you made to the file `TypeDetection.xcu`:

- If you *copied* `TypeDetection.xcu` into your user settings you can delete or (to be safe) rename the file.
- If you *edited* an existing version of `TypeDetection.xcu` you should restore the backup copy of the file. If you forgot to take a backup, you will have to delete the additions by hand.

When you restart OOo, the filters should have disappeared.

Installation

Note: If you have made a -net (multiuser) installation of OOo, you will probably need to log in as root/administrator to install Writer2LaTeX.

Before you start, you need an installation of OOo where

- You have set up OOo to use Java. If you didn't do that during installation, you can run `<OOo install>/program/jvmsetup`. (Of course this requires that you have installed Java on your system).
- You must have the *Mobile Device Filters installed*. If you didn't install these during installation (it's not part of a standard installation!), you can run OOo setup, choose Modify and add the filters. This will install a framework for Java based filters in OOo (known as xmerge), which is also used by Writer2LaTeX (despite the fact that it has nothing to do with mobile devices).

Then the installation proceeds as follows:

1. Copy `writer2latex.jar`, `xmergefix.jar` and `writer2latex.xml` into the classes directory

   ```
   <OOo install>/program/classes/
   ```

2. Rename the existing `xmerge.jar` to `oldxmerge.jar` (or whatever you like; this is only to have a backup of the old version).
3. Rename `xmergefix.jar` to `xmerge.jar`.
4. Copy `w2lfilter.zip` into the directory

   ```
   <OOo install>/share/uno_packages
   ```

5. Make sure that no OOo processes are running: Close all document windows and (under MS Windows) the Quick Starter.

6. From a command shell navigate to the directory

   ```
   <OOo install>/program
   ```

   and type

   ```
   pkgchk --shared
   ```

   This will register Writer2LaTeX as a filter in OOo. If it works, there will be no messages on the screen.

7. Now restart OOo.

To use the LaTeX files generated by Writer2LaTeX, you will need the LaTeX package `writer.sty`, see section 3.1.

2.3 Uninstall Writer2LaTeX

To remove the Writer2LaTeX filters from your OOo installation, you should proceed as follows:

1. Delete `w2lfilter.zip from` the directory

   ```
   <OOo install>/share/uno_packages
   ```

2. Make sure that no OOo processes are running: Close all document windows and (under MS Windows) the Quick Starter.

3. From a command shell navigate to the directory

   ```
   <OOo install>/program
   ```

   and type

   ```
   pkgchk --shared
   ```

   This will remove the registration of Writer2LaTeX from OOo. If it works, there will be no messages on the screen.

4. Now restart OOo.

You may also undo the changes you have made to OOo's `classes` directory, but this is not required.

3 Using Writer2LaTeX and Writer2BibTeX

Writer2LaTeX is quite flexible: It can take advantage of several LaTeX packages, such as `hyperref`, `pifont`, `ulem`. It supports 16 different languages, including greek and russian.

The flexibility makes it possible to use Writer2LaTeX from several philosophies:

- You can use LaTeX as a typesetting engine for your OOo documents: Writer2LaTeX will create a LaTeX document with the same look and feel as the original (this is the default behavior). Note that the resulting LaTeX source will be readable, but not very clean.

- If you for some reason need to continue the work on your document in LaTeX your primary interest may be the content rather than the formatting. Writer2LaTeX can be configured to produce a LaTeX document which strips most of the formatting and hence produces a clean LaTeX source from any source document.

- If you don't like to write LaTeX code by hand, you may use OOo as a simple graphical front-end for LaTeX. Using a special OOo Writer template and a special configuration file for Writer2LaTeX, you can create well-structured LaTeX documents that resembles "hand-written" LaTeX documents. You can compare this to the way LyX works.

3.1 The LaTeX package `writer.sty`

The LaTeX documents generated by Writer2LaTeX requires the package `writer.sty`, which is contained in the Writer2LaTeX distribution. This package contains some useful macros, that are not available in the standard LaTeX packages (for example `\textsubscript` to use subscript in text mode, `\ddotsup` to insert diagonal raising dots etc.).

It is sufficient to place `writer.sty` in the same directory as the converted LaTeX document. It will however be more convenient if you install it in your TeX distribution. The proper place will usually be the "local texmf tree", please see the documentation of your TeX distribution. Below are specific instructions for teTeX and MikTeX:

Instructions for teTeX (unix)

If you use teTeX you can install `writer.sty` as follows:

Open a shell and type

```
texconfig conf
```

This will list the configuration details for teTeX. Under the heading "Kpathsea" you will see a list of directories searched by TeX. You can put `writer.sty` in the subdirectory `tex` of any of these directories. Usually the directory

```
/home/<user name>/texmf/tex
```

can be used (you can create it if it doesn't exist).

Next you should type

```
texconfig rehash
```

to make teTeX refresh it's filename database.

Instructions for MikTeX (Windows)

If you use MikTeX you can install `writer.sty` as follows:

Copy `writer.sty` to the `tex` subdirectory in the local texmf tree. With a standard installation this will be the directory

```
c:\localtexmf\tex
```

If this directory does not exist you should start "MikTeX Options" (you can find this in the Start Menu). On the tab page **Roots** you can see the location of the local texmf tree.

Next you should start “MikTeX Options”. On the tab page **General**, click the button **Refresh Now** to make MikTeX refresh it’s filename database.

3.2 Converting to LaTeX from the command line

To convert a file to LaTeX use the command line

```
w2l [-latex] [-config <configfile>] <writer document to convert>
```

The parts in square brackets are optional.

This will produce a LaTeX file with the same name as the original document, but the extension changed to `.tex`.

Examples:

```
w2l mydocument.sxw
```

or

```
w2l -config clean.xml mydocument.sxw
```

If you specify the `-config` option, Writer2LaTeX will load this configuration file before converting your document. You can read more about configuration in section 3.5.

The script `w2l` also provides a shorthand notation to use the sample configuration files included in `writer2latex032.zip`. The command line is

```
w2l [-clean|-pdfscreen|-pdfprint|-article] <writer document to convert>
```

For example to produce a clean LaTeX file (ie. ignoring most of the formatting from the source document):

```
w2l -clean mydocument.sxw
```

It is recommended that you extend `w2l` / `w2l.bat` to support your own configuration files.

3.3 Converting to BibTeX from the command line

Writer2BibTeX extracts bibliography data to a BibTeX file. To do this use the commandline

```
w2l -bibtex <writer document to convert>
```

You can also extract the data as part of the conversion to LaTeX, see section 3.5.

3.4 Using Writer2LaTeX and Writer2BibTeX as export filters

If you choose **File** – **Export** in Writer you should be able to choose **LaTeX 2e**, **BibTeX data file** as file type.

Note: You have to use the export menu because there is no import filter for LaTeX/BibTeX. You should always save in the native format of OOo as well!

Note: Currently embedded graphics are not converted when Writer2LaTeX is used as an export filter. Also using Writer2BibTeX in conjunction with Writer2LaTeX is currently only possible from the command line. This is because of an issue with xmerge. A fix for this is planned for Writer2LaTeX version 0.3.4.

3.5 Configuration

LaTeX export can be configured with a configuration file. The configuration is read from several sources:

- First Writer2LaTeX reads the file `writer2latex.xml` in the same directory as `writer2latex.jar`. This file is supposed to contain installation-wide configuration.
- Then it reads the file `writer2latex.xml` in your home directory (unix, eg. `/home/username`) or user profile (windows, eg. `c:\documents and settings\username`). This file is supposed to contain user-specific default confguration.
- Then the file `<documentname>-config.xml` in the same directory in the source document is read.
- If you specify a configuration file on the command line, this file will replace `<documentname>-config.xml`.

The configuration file is an xml file, here are the default contents:

```
<config>
  <option name="create_document_config" value="true" />
  <option name="backend" value="generic" />
  <option name="inputencoding" value="ascii" />
  <option name="greek_math" value="true" />
  <option name="use_pifont" value="false" />
  <option name="use_ifsym" value="false" />
  <option name="use_wasysym" value="false" />
  <option name="use_bbding" value="false" />
  <option name="use_hyperref" value="true" />
  <option name="use_endnotes" value="false" />
  <option name="use_ulem" value="false" />
  <option name="use_bibtex" value="false" />
  <option name="bibtex_style" value="plain" />
  <option name="main_paragraph_style" value="" />
  <option name="ignore_unknown_paragraph_styles" value="false" />
  <option name="hard_paragraph_formatting" value="hard" />
  <option name="ignore_list_label_styles" value="false" />
  <option name="ignore_heading_styles" value="false" />
  <option name="ignore_footnotes_configuration" value="false" />
  <option name="ignore_page_formatting" value="false" />
  <custom-preamble />
</config>
```

The meaning of each part is explained in the following sections. Writer2LaTeX comes with four sample configuration files:

- `clean.xml` to produce a *clean* LaTeX file, ie. most of the formatting is ignored.
- `pdfscreen.xml` to produce a LaTeX file which is optimized for screen viewing using the package `pdfscreen.sty`.
- `pdfprint.xml` to produce a LaTeX file which is optimized for printing with pdfTeX.
- `article.xml` to produce a LaTeX article, see below.

Basic options

- If the option `create_document_config` if set to `true`, the document specific configuration file mentioned above will be created if it does not exist.
- The option `backend` can have any of the values `generic` (default), `dvips` or `pdftex`. This will create LaTeX files suitable for any backend/dvi driver, dvips or pdfTeX respectively.
- The option `inputencoding` can have any of the values `ascii` (default), `latin1`, `iso-8859-7`, `cp1252`, `koi8-r` or `utf8`. The latter requires Dominique Unruh's `ucs.sty`.

Font and symbol options

- The option `greek_math` can have the values `true` (default) or `false`. This means that greek letters in latin or russian text are rendered in math mode. This behaviour assumes that greek letters are used as symbols in this context, and has the advantage that greek text fonts are not required. It is *not* used in greek text, where it would be awful.
- The option `use_pifont` can have the values `true` or `false` (default). This enables the use of *Zapf Dingbats* using the LaTeX package `pifont.sty`.
- The option `use_wasysym` can have the values `true` or `false` (default). This enables the use of the *wasy* symbol font using the LaTeX package `wasysym.sty`.
- The option `use_ifsym` can have the values `true` or `false` (default). This enables the use of the *ifsym* symbol font using the LaTeX package `ifsym.sty`.
- The option `use_bbding` can have the values `true` or `false` (default). This enables the use of the *bbding* symbol font (a clone of Zapf Dingbats) using the LaTeX package `bbding.sty`.

Options for other packages

- The option `use_hyperref` can have the values `true` (default) or `false`. This enables use of the package `hyperref.sty` to include hyperlinks in the LaTeX document.

- The option `use_endnotes` can have the values `true` or `false` (default). This enables use of the package `endnotes.sty` to include endnotes in the LaTeX document. If set to `false`, endnotes will be converted to footnotes.
- The option `use_ulem` can have the values `true` or `false` (default). This enables use of the package `ulem.sty` to support underlining and crossing out in the LaTeX document.

Options for BibTeX

- The option `use_bibtex` can have the values `true` or `false` (default). This enables the use of BibTeX for bibliography generation. If it is set to `false`, the bibliography is included as text.
- The option `bibtex_style` can have any BibTeX style as value (default is `plain`). This is the BibTeX style to be used in the LaTeX document.

Options to control export of formatting

In Writer, formatting is controlled by styles. Per default Writer2LaTeX will convert all styles to LaTeX while preserving as much formatting as possible. You can however change this behaviour in the configuration file.

- The option `main_paragraph_style` can be set to any Writer paragraph style. Writer2LaTeX usually wraps up each paragraph in a LaTeX environment, eg.

  ```
  \begin{styleTextbody}
  This is the text of the paragraph.
  \end{styleTextbody}
  ```

  but not this particular style. This will create a more normal looking LaTeX file, since ususally only special paragraphs (ie. quotations) are wrapped up in enviroments. Note that this will ignore any paragraph formatting in this style, but not character formatting.

- The option `hard_paragraph_formatting` can have the values `hard` (default), `soft` and `ignore`. This option controls how hard paragraph formatting is handled. The default places the formatting instructions directly in the text, making it somewhat hard to read.

  If you change the value to `soft`, Writer2LaTeX will not differentiate between hard and soft formatting; this means that several enviroments wil be created like for soft formatting. They wil usually be named something like `stylePi`, `stylePii` etc.

  The value `ignore` instructs Writer2LaTeX to ignore any hard paragraph formatting and only use soft formatting (this affects paragraph formatting only, not hard character formatting associated with the paragraph).

- The option `ignore_unknown_paragraph_styles` can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore any paragraph styles that does not have a style-map in the configuration file, see below.

- The option `ignore_list_label_styles` can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore the character formatting used in list labels (including outline numbering), so that labels will be formatted like the surrounding text.

- The option `ignore_heading_styles` can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore the formatting for headings, so that LaTeX's defaults will be used.

- The option `ignore_footnotes_configuration` can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore the formatting for footnotes and endnotes (this is what you define with **Tools** – **Footnotes**), so that LaTeX's defaults will be used.

- The option `ignore_page_formatting` can have the values `true` or `false` (default). Setting the option to `true` will instruct Writer2LaTeX to ignore page formatting. This is not recommended.

Style maps

In addition you can specify maps from styles in Writer to your own LaTeX styles in the configuration. Currently this is possible for text styles, paragraph styles and list styles. For example the following confguration files overrides the text style Emphasis and the paragraph style Quotations as well as the list style theorem.

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <style-map name="Emphasis" class="text" before="\emph{" after="}" />
  <style-map name="Quotations" class="paragraph"
before="\begin{quotation}" after="\end{quotation}" />
  <style-map name="theorem" class="list" before="" after="" />
  <style-map name="theorem" class="listitem" before="\begin{theorem}"
after="\end{theorem}" />
</config>
```

Note that there are two entries for a list style: The first one to specify the LaTeX code to put before and after the entire list. The second one to specify the LaTeX code to put before and after each list item. The example shows how to create theorem environments from a list in Writer.

When you override a style, all formatting specified in the original document will be igored.

Math symbols

In OOo Math you can add user-defined symbols. Writer2LaTeX already understands the predefined symbols such as `%alpha`. If you define your own symbols, you can add an entry in the confguration that specifies LaTeX code to use. The `math-symbol-map` element is used for this:

```
<math-symbol-map name=’’ddarrow’’ latex=’’\Downarrow’’ />
```

This example will map the symbol `%ddarrow` to the LaTeX code `\Downarrow`.

Custom preamble

The text you specify in the element `custom-preamble` will be copied verbatim into the LaTeX preamble. For example:

```
    <custom-preamble>\usepackage{palatino}</custom-preamble>
```

to typeset your document using the postscript font palatino.

3.6 Using OpenOffice.org as a frontend for LaTeX

Writer2LaTeX has some rudimentary support for using OOo as a frontend for LaTeX . The long term goal of this is to turn Writer into a near-wysiwyg LaTeX editor somewhat like LyX.

Here is a short description:

Create a new document based on the template LaTeX-article.stw.

This template contains a number of styles that corresponds to LaTeX styles:

Paragraph styles **Heading 1** through **Heading 5** should be used for headings (as usual). The styles **flushleft**, **flushright**, **center**, **quote**, **quotation**, **verse** corresponds to the environments of the same name in LaTeX.

The paragraph styles **itemize**, **enumerate** are used to start a new list.

The paragraph style **theorem** is used to insert a numbered theorem.

Text styles **Huge**, **huge**, **LARGE**, **Large**, **large**, **normalsize**, **small**, **footnotesize**, **scriptsize**, **tiny**, **textrm**, **textsf**, **texttt**, **textup**, **textit**, **textsl**, **textsc**, **textmd**, **textbf** all corresponds to the font changing commends of the same name in LaTeX.

If you use these styles (and no other style!) and uses the configuration file `article.xml` when you convert your document with Writer2LaTeX, you will get a result that resembles a handwritten LaTeX file.

Note: As Writer2LaTeX cannot import LaTeX files into Writer, you should save your document in the native format and convert to LaTeX when you are finished (or want to see the result).

4 Using Writer2xhtml

Writer2xhtml is producing standards compliant XHTML files, in particular it can be used to put math on the web using the XHTML + MathML combination. Thus Writer2xhtml can convert into any of these XHTML variants:

- HTML 1.0 strict, which follows the guidelines for HTML compatibility, so that the output should be viewable with any browser that supports HTML 4. The default file extension for this format is `.html`.

- XHTML 1.1 + MathML 2.0, which currently is viewable with Mozilla and Amaya only. The default file extension is `.xhtml`.

- XHTML 1.1 + MathML 2.0 using XSL transformations from the W3C Math Working Group to make the file viewable also in some browsers that needs a plugin to display MathML, eg. Internet Explorer with MathPlayer plugin. The default file extension is `.xml`.

  This is how W3C's Math Working Group recommends to put "math on the web".

Writer2xhtml is quite flexible; in particular with respect to the handling of formatting:

- You can let Writer2xhtml convert the style information in the source document and thus get an xhtml document that has the same look and feel as the original.

- You can use your own style sheet and let Writer2xhtml convert the content only. You can map styles in OOo to xhtml elements and css classes, see sections 4.3 and 4.4

4.1 Converting to XHTML from the command line

To convert a file to XHTML use the command line

```
 w2l -xhtml|-xhtml+mathml|-xhtml+mathml+xsl [-config <configfile>]
<writer document to convert>
```

The parts in square brackets are optional.

This will produce an XHTML file with the same name as the original document, but a different file extension.

The option `-xhtml+mathml` is used to produce XHTML 1.1 + MathML 2.0, the option `-xhtml+mathml+xsl` produces the variant using XSL transformations.

Examples:

```
w2l -xhtml+mathml+xsl mydocument.sxw
```

or

```
w2l -xhtml -config clean.xml mydocument.sxw
```

If you specify the `-config` option, Writer2xhtml will load this configuration file before converting your document. You can read more about configuration in section 4.3.

The script `w2l` also provides a shorthand notation to use the sample configuration file included in `writer2latex032.zip`. The command line is

```
w2l -cleanxhtml <writer document to convert>
```

This configuration file produces a "clean" xhtml file (see section 4.4), for example:

```
w2l -cleanxhtml mydocument.sxw
```

It is recommended that you extend `w2l` / `w2l.bat` to support your own configuration files.

4.2 Using Writer2xhtml as an export filter

If you choose **File – Export** in Writer you should be able to choose **XHTML 1.0 strict**, **XHTML 1.1 + MathML 2.0** or **XHTML 1.1 + MathML 2.0 (xsl)** as file type.

Note: You have to use the export menu because Writer2xhtml does not provide an import filter for XHTML. You should always save in the native format of OOo as well!

Note: Currently embedded graphics are not converted when Writer2xhtml is used as an export filter. Also splitting at headings only works from the command line. This is because of an issue with the xmerge framework. A fix for this is planned for Writer2xhtml version 0.3.4.

4.3 Configuration

XHTML export can be configured with a configuration file. The configuration is read from several sources:

- First Writer2xhtml reads the file `writer2latex.xml` in the same directory as `writer2latex.jar`. This file is supposed to contain installation-wide configuration.
- Then it reads the file `writer2latex.xml` in your home directory (unix, eg. `/home/username`) or user profile (windows, eg. `c:\documents and settings\username`). This file is supposed to contain user-specific default configuration.
- Then the file `<documentname>-config.xml` in the same directory in the source document is read.
- If you specify a configuration file on the command line, this file will replace `<documentname>-config.xml`.

The configuration file is an xml file, here are the default contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <option name="create_document_config" value="false" />
  <option name="xhtml_custom_stylesheet" value="" />
  <option name="xhtml_ignore_styles" value="false" />
  <option name="xhtml_use_dublin_core" value="true" />
  <option name="xhtml_scaling" value="100%" />
  <option name="xhtml_split_level" value="0" />
</config>
```

Options

- If the option `create_document_config` if set to `true`, the document specific configuration file mentioned above will be created if it does not exist.
- The option `xhtml_custom_stylesheet` is used to specify an URL to your own, external stylesheet. If the value is empty or the option is not specified, no external stylesheet will be used.
- The option `xhtml_ignore_styles` is used to specify if formatting should be exported. If the value is true, no style information will be exported (in this case you should specify

a custom style sheet!).

- The option `xhtml_use_dublin_core` is used to specify if Dublin Core Meta data should be exported (the format will be as specified in http://dublincore.org/documents/dcq-html/). If the value is `false`, it will not be exported.

- The option `xhtml_scaling` is used to specify a scaling of all formatting, ie. to get a different text size than the original document. The value must be a percentage.

- The option `xhtml_splitting` is used to specify that the documents should be split in several documents and the outline level at which the splitting should happen (the default 0 means no split). This is convenient for long documents. Each output document will get a simple navigation panel in the header and the footer.

Style maps

In addition to the options, you can specify that certain styles in Writer should be mapped to specific XHTML elements and CSS style classes. Here are some examples showing how to use some of the built-in Writer styles to create XHTML elements:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <!-- map OOo paragraph styles to xhtml elements -->
  <xhtml-style-map name="Text body" class="paragraph"
           element="p" css="(none)" />
  <xhtml-style-map name="Sender" class="paragraph"
           element="address" css="(none)" />
  <xhtml-style-map name="Quotations" class="paragraph"
           block-element="blockquote" block-css="(none)"
           element="p" css="(none)" />

  <!-- map OOo text styles to xhtml elements -->
  <xhtml-style-map name="Citation" class="text"
           element="cite" css="(none)" />
  <xhtml-style-map name="Emphasis" class="text"
           element="em" css="(none)" />


  <!-- map hard formatting attributes to xhtml elements -->
  <xhtml-style-map name="bold" class="attribute"
           element="b" css="(none)" />
  <xhtml-style-map name="italics" class="attribute"
           element="i" css="(none)" />
</config>
```

An extended version of this is distributed with Writer2LaTeX, please see the file `cleanxhtml.xml`.

The attributes of the `xhtml-style-map` element are used as follows:

- `name` specifies the name of the Writer style.
- `class` specifies the styles class in Writer; this can either be `text`, `paragraph`, `frame`, `list` or `attribute`. The last value does not specify a real style, but refers to hard formatting attributes. The possible names in this case are `bold`, `italics`, `fixed` (for fixed pitch fonts), `superscript` and `subscript`.
- `element` specifies the XHTML element to use when converting this style. This is not used for frame and list styles.
- `css` specifies the CSS style class to use when converting this style. If it is not specified or the value is ‘‘`(none)`’’, no CSS class will be used.
- `block-element` only has effect for paragraph styles. It is used to specify a block XHTML element, that should surround several exported paragraphs with this style.
- `block-css` specifies the CSS style class to be used for this block element. If it is not specified or the value is ‘‘`(none)`’’, no CSS class will be used.

For example the rules above produces code like this:

```
<p>This paragraph is Text body</p>
<address>This paragraph i Sender</address>
<blockquote>
  <p>This paragraph is Quotations</p>
  <p>This paragraph is also Quotations</p>
</blockquote>
<p>This paragraph is also Text body and has some <em>text with emphasis
style</em> and uses some <b>hard formatting</b>.</p>
```

You can use your own Writer styles together with your own CSS style sheet to create further style mappings, for example:

```
<xhtml-style-map name="Some OOo style" class="paragraph"
           block-element="div" block-css="block_style"
           element="p" css="par_style" />
```

to produce output like this:

```
<div class=’’block_style’’>
  <p class=’’par_style’’>Paragraph with Some OOo style</p>
  <p class=’’par_style’’>Yet another</p>
</div>
```

Note that the rules for hard formatting are only used when `xhtml_ignore_styles` is set to `true`. It is not recommended to rely on these rules, using real text styles is preferable. They are included because the use of hard character formatting is very common even in otherwise well-structured documents.

4.4 Using OpenOffice.org to create XHTML documents

The configuration file `cleanxhtml.xml` that is distributed with Writer2LaTeX, can be used to create semantically rich XHTML content, which can be formatted with your own stylesheet (you should edit the file to add the URL to the stylesheet you want to use).

A subset of the built-in styles in Writer are mapped to XHTML elements (note that the style names are localized, so this is for the english version of OpenOffice.org):

OOo Writer style	***OOo Writer class***	***XHTML element***
Text body	paragraph style	`p`
Sender	paragraph style	`address`
Quotations	paragraph style	`blockquote`
Preformatted Text	paragraph style	`pre`
List Heading	paragraph style	`dt` (in `dl`)
List Contents	paragraph style	`dd` (in `dl`)
Horizontal Rule	paragraph style	`hr`
Citation	text style	`cite`
Definition	text style	`dfn`
Emphasis	text style	`em`
Example	text style	`samp`
Source Text	text style	`code`
Strong Emphasis	text style	`strong`
Teletype	text style	`tt`
User entry	text style	`kbd`
Variable	text style	`var`
bold	hard formatting attribute	`b`
italics	hard formatting attribute	`i`
fixed pitch font	hard formatting attribute	`tt`
superscript	hard formatting attribute	`sup`
subscript	hard formatting attribute	`sub`

So by using these styles only, you will create well-structured XHTML documents. See the document `sample-xhtml.sxw` for an example of how to use this.

Warning: Some elements are not allowed inside `pre`, so this might in some cases lead to invalid documents. This will be fixed in Writer2LaTeX 0.3.4.

5 Using Writer2LaTeX from another Java application

Writer2LaTeX uses the *xmerge* framework. Please see the documentation at `xml.openoffice.org/xmerge` for explanation about how to use Writer2LaTeX from Java.