CSSToXSLFO Manual

Version 1.0rc2
Werner Donné
Re BVBA
11th January 2005

Introduction

CSSToXSLFO is a tool which converts an XML document, combined with a CSS2 style sheet, into an XSLFO file. It has some special provisions for XHTML, which is also an XML vocabulary. The tool implements a reasonable subset of CSS2. It also adds a few extensions for handling page-related issues properly. Note that the tool is not a general-purpose printing tool for any kind of HTML pages you can find on the Internet.

The goal of CSSToXSLFO is to provide a rather easy interface to fine printing environments that use XSLFO as their input. It is a compromise between the simplicity of style sheet expression and the quality of the result. XSLFO is quite difficult. Writing style sheets that produce it are mostly written in XSLT, which is not straightforward to everyone either. CSS on the other hand is rather simple and yet it is powerful. In fact it combines element selection and formatting specification in one easy-to-learn syntax. The cost is that a lot of interesting XSLFO features are not available.

An area where the tool can be a plus is the programmatic generation of reports within applications. The variety in style for reports is not that great. The offered feature set of CSSToXSLFO can be sufficient. Having report programmers learn XSLFO and XSLT is not always an option, while many know CSS and XHTML well enough to be productive with it.

Another use-case for CSSToXSLFO is writing documents in XML. One can put work in a style sheet once and reuse that through the mark-up paradigm, in which content and formatting are separated. The formatting features should be sufficient to produce day-to-day documents in a typical business environment. Such documents don't tend to be very complicated, with respect to layout that is.

Specifying Style Sheets

The most general way of specifying a style sheet for a document with CSSToXSLFO is the proposal in section 2.2 of [CSS2]. It consists of a processing instruction, which precedes the document, looking like this:

```
<?xml:stylesheet type="text/css" href="style.css"?>
```

For XHTML there are a few additional options. You can use the `link` element to link a style sheet (only persistent style sheets) to the document or you can embed it with the `style` element. The `style` attribute is also available as specified in [XHTML].

Running It

There are four packages you can run from the command-line: one that produces plain XSLFO, one that returns the output of XEP, a product from RenderX (http://www.renderx.com), another that returns the output of XSLFormatter, a product from Antenna House (http://www.antennahouse.com) and finally, one runs FOP (http://xml.apache.org/fop/). The last one comes with a filter that removes a few properties, which are not supported by FOP. This makes FOP complain less. You need JDK1.4 or higher to run them. The command-lines look as follows:

For plain CSSToXSLFO:

```
> java -jar css2xslfo.jar url_or_filename <options>
```

For XEP3:

```
> java -Dcom.renderx.xep.ROOT=<XEP location> -jar css2xep.jar
    url_or_filename <options>
```

For XEP4:

```
> java -Dcom.renderx.xep.CONFIG=<XEP location>/xep.xml
    -jar <XEP location>/lib/css2xep.jar url_or_filename <options>
```

For XSLFormatter:

```
> set dynamic library path to <XSLFormatter location>/lib
> set environment variable AH_FONT_CONFIGFILE to
    <XSLFormatter location>/etc/font-config.xml
> java -jar <XSLFormatter location>/lib/css2xsl.jar
    url_or_filename <options>
```

For FOP:

```
> java -jar css2fop.jar url_or_filename <options>
```

Additional system properties and/or enviroment variables can be set. Please consult the product-specific documentation for this.

In order for css2xep.jar to work, you should place it in the <XEP location>/lib directory and create a link to or a copy of your XEP JAR file with the name “xep.jar”. Since XEP4 the link or copy are no longer needed, because the XEP JAR file has the expected name.

For css2xsl.jar to work, you should place it in <XSLFormatter location>/lib.

The css2fop.jar file needs to be next to fop.jar, which should be next to the packages is uses. This normally is in the FOP lib directory.

CSSToXSLFO uses the XSLT-processor that comes with the JDK, which is Xalan from Apache. For better performance you can prepend Xalan 2.6.0 to your boot classpath as follows (assuming /usr/local as the installation directory):

```
> java -Xbootclasspath/p:/usr/local/xalan-j_2_6_0/bin/xalan.jar
    -jar css2xslfo.jar url_or_filename <options>
```

Common Options

The following options are common to all three variants. The document to be processed can be specified with a URL or filename. If it is omitted, stdin will be read.

-baseurl <URL>
: Change the base URL of the input document. By default it is the URL of the document itself.

-c <URL or filename>
: Specify a catalog in the format defined by SGML Open Technical Resolution TR9401:1997. Only the “PUBLIC” and “SYSTEM” keywords are supported.

-h
: Display the command-line syntax.

-p <comma-separated list of URLs or filenames>
: A list of pre-processing XSLT style sheets that are executed on the input document, in the specified order, before anything else. The result should be valid XHTML.

-uacss <URL or filename>
: Use another User Agent style sheet than the one built-in for XHTML. Note that the latter will only be active for documents which are in the XHTML namespace (http://www.w3.org/1999/xhtml).

-v
: Turn on XML validation of the input document.

parameter=value
: Specify User Agent parameters. Equivalent CSS constructs precede these.

Options Specific To css2xslfo.jar

-debug
: Produces the file CSSToXSLFO.xml, which is the input document having all the applied CSS properties on the elements in another namespace. This is useful for checking your CSS style sheet as well as the tool itself.

-fo <filename>
: The XSLFO output file. If it is omitted stdout will be written instead.

Options Specific To css2xep.jar

One the following options should be specified.

-pdf <filename>
: The PDF output file.

-ps <filename>
: The PostScript output file.

Options Specific To css2xsl.jar

-pdf <filename>
: The PDF output file. This option is mandatory.

Options Specific To css2fop.jar

-fc <filename>
: An user configuration file.

-pdf <filename>
: The PDF output file.

-ps <filename>
: The PostScript output file.

-q
: Makes FOP silent.

-svg <filename>
: The SVG output file.

User Agent Parameters

The User Agent parameters are common to all three packages. They have no effect if there are @page rules in the style sheet, except for the “rule-thickness” parameter. Furthermore, equivalent CSS constructs, when present in the style sheet, always preceed.

column-count
: The number of columns on a page. The default is “1”.

country
: The country code. The default is “GB”.

font-size
: The point size of the font. The default for paper sizes “a5” and “b5” is “10pt”. For all other paper sizes the default is “11pt”. See also the “paper-size” parameter.

html-header-mark
: An HTML element can be passed here. Its contents will be used as the running header. By default there is no mark.

language
: The language code. The default is “en”.

odd-even-hift
: The amount by which the page contents is shifted in the inline progression direction when the paper mode is “twosided”. The default is “10mm”. See also the “paper-mode” parameter.

orientation
: The allowed values are “portrait”, which is the default, and “landscape”.

paper-margin-bottom
: The bottom margin of a page. The default is “0mm”.

paper-margin-left
: The left margin of a page. The default is “25mm”.

paper-margin-right
: The right margin of a page. The default is “25mm”.

paper-margin-top
: The top margin of a page. The default is “10mm”.

paper-mode
: The allowed values are “onesided”, which is the default, and “twosided”.

paper-size
: The allowed values are “a4”, which is the default, “a5”, “b5”, “executive”, “letter” and “legal”.

rule-thickness
: The default thickness for rules when there was no CSS specification for it. The default is “0.2pt”.

writing-mode
: The XSLFO writing mode. The default is “lr-tb”. Other possible values are “rl-tb”, “tb-rl”, “lr”, “rl” and “tb”. See also [XSLFO].

Compliance With CSS2

Specifications

Section	Implemented	Remarks and restrictions
4.1 Syntax	yes	Thanks to Flute.
4.2 Rules for handling parsing errors	partial	Unknown properties will end up in the XSL-FO file and cause errors in a XSL-FO processor.
4.3 Values	yes	Thanks to Flute.
4.4 CSS document representation	yes	Thanks to Flute.
5 Selectors	partial	All sections but 5.11.2, 5.11.3 and 5.12.2.
6 Assigning property values, Cascading and Inheritance	yes	
7 Media types	yes	By design, only types `all` and `print` are supported.
8 Box model	yes	
9.1.1 The viewport	no	
9.1.2 Containing blocks	yes	
9.2.1 Block-level elements and block boxes	partial	Compact and run-in boxes and markers are not supported.
9.2.2 Inline-level elements and inline boxes	partial	Compact and run-in boxes and inline tables are not supported.
9.2.3 Compact boxes	no	

Section	Implemented	Remarks and restrictions
9.2.4 Run-in boxes	no	
9.2.5 The 'display' property	partial	See property table.
9.3 Positioning schemes	yes	
9.4 Normal flow	yes	
9.5 Floats	yes	
9.6 Absolute positioning	yes	
9.7 Relationships between 'display', 'position', and 'float'	yes	
9.9 Layered presentation	yes	
9.10 Text direction: the 'direction' and 'unicode-bibi' properties	yes	
10 Visual formatting model details	yes	
11 Visual effects	yes	
12.1 The :before and :after pseudo-elements	yes	
12.2 The 'content' property	yes	
12.3 Interaction of :before and :after with 'compact' and 'run-in' elements	no	
12.4 Quotation marks	yes	
12.5 Automatic counters and numbering	yes	
12.6.1 Markers: the 'marker-offset' property	no	
12.6.2 Lists: the 'list-style-type', 'list-style-image', 'list-style-position', and 'list-style' properties	partial	The list-style-image and list-style-position properties are not supported. See also the property table.
13.2.1 Page margins	yes	
13.2.2 Page size: the 'size' property	yes	
13.2.3 Crop marks: the 'marks' property	no	
13.2.4 Left, right, and first pages	yes	
13.2.5 Content outside the page box	yes	
13.3 Page breaks	partial	Named pages are only supported for direct sibling children in the `body` element of XHTML or the document element in other XML vocabularies.

Section	Implemented	Remarks and restrictions
13.4 Cascading in the page context	yes	
14 Colors and Backgrounds	yes	
15 Fonts	partial	@font-face and descriptors are not supported.
16 Text	yes	
17 Tables	partial	Inline tables are not supported. Anonymous table objects are only supported for missing table groups and missing table cells in a row, on the condition that there are table column elements. Audio rendering is not supported.
18 User interface	no	
19 Aural style sheets	no	

Properties

Property	Implemented	Remarks and restrictions
azimuth	no	
background	yes	
background-attachment	yes	
background-color	yes	
background-image	yes	
background-position	yes	
background-repeat	yes	
border	yes	
border-bottom	yes	
border-bottom-color	yes	
border-bottom-style	yes	
border-bottom-width	yes	
border-collapse	partial	Not for `inline-table`.
border-color	yes	
border-left	yes	
border-left-color	yes	
border-left-style	yes	
border-left-width	yes	
border-right	yes	

Property	Implemented	Remarks and restrictions
border-right-color	yes	
border-right-style	yes	
border-right-width	yes	
border-spacing	partial	Not for `inline-table`.
border-style	yes	
border-top	yes	
border-top-color	yes	
border-top-style	yes	
border-top-width	yes	
borded-width	yes	
bottom	yes	
caption-side	yes	
clear	yes	
clip	yes	
color	yes	
content	yes	
counter-increment	yes	
counter-reset	yes	
cue	no	
cue-after	no	
cue-before	no	
cursor	no	
direction	yes	
display	partial	The values `run-in`, `compact`, `marker` and `inline-table` are not supported.
elevation	no	
empty-cells	yes	
float	yes	
fonts	yes	
font-family	yes	
font-size	yes	
font-size-adjust	yes	
font-stretch	yes	
font-style	yes	

Property	Implemented	Remarks and restrictions
font-variant	yes	
font-weight	yes	
height	yes	
left	yes	
letter-spacing	yes	
line-height	yes	
list-style	partial	See individual properties.
list-style-image	no	
list-style-position	no	
list-style-type	partial	Only the values `disc`, `circle`, `square`, `decimal`, `lower-roman`, `upper-roman`, `lower-alpha`, `lower-latin`, `upper-alpha`, `upper-latin` and `none` are supported. The additional glyphs defined in [CSS3L] are also supported. Those are `box`, `check`, `diamond` and `hyphen`.
margin	yes	
margin-bottom	yes	
margin-left	yes	
margin-right	yes	
margin-top	yes	
marker-offset	no	
marks	no	
max-height	yes	
max-width	yes	
min-height	yes	
min-width	yes	
orphans	yes	
outline	no	
outline-color	no	
outline-style	no	
outline-width	no	
overflow	yes	
padding	yes	

Property	Implemented	Remarks and restrictions
padding-bottom	yes	
padding-left	yes	
padding-right	yes	
padding-top	yes	
page	partial	Only for direct sibling children of the `body` element in XHTML or for direct sibling children of the document element in other XML vocabularies.
page-break-after	yes	
page-break-before	yes	
page-break-inside	yes	
pause	no	
pause-after	no	
pause-before	no	
pitch	no	
play-during	no	
play-range	no	
position	yes	
quotes	yes	
richness	no	
right	yes	
size	yes	
speak	no	
speak-header	no	
speak-numeral	no	
speak-punctuation	no	
speech-rate	no	
stress	no	
table-layout	yes	
text-align	yes	
text-decoration	yes	
text-indent	yes	
text-transform	yes	
top	yes	

Property	Implemented	Remarks and restrictions
unicode-bibi	yes	
vertical-align	yes	
visibility	yes	
voice-family	no	
volume	no	
white-space	yes	
widows	yes	
width	yes	
word-spacing	yes	
z-index	yes	

Extensions

The extension features of the tool mostly pertain to page-oriented aspects. Care has been taken to not introduce new syntax. There are, however, a number of new properties. Those are normally safely ignored by browsers. In the case where there would be an impact on the layout produced by browsers, the properties can be confined to the “print” medium through @media rules.

Page Regions

This extension introduces XSLFO-compatible page regions. Regions can be defined by placing a `region` property on an element. Such an element may be anywhere in the document. The allowed values are `bottom`, `left`, `right` and `top`.

On top of that, either the `width` property, for left and right regions, or the `height` property, for top and bottom regions, should be defined for them. They will determine the dimensions of the page regions. The default value for `width` is “20mm”. For `height` it is “10mm”.

The regions work together with the @page rules, i.e. it is possible to specify different regions, which correspond to the different page types in the style sheet. This can be achieved by also specifying the `page` property, which is a standard CSS2 property. The cascading mechanism for @page rules also apply to the regions. The style sheet for this manual for example is as follows:

```
div.bottom-left, div.bottom-right { display: none; }

@media print
{
  div.bottom-left
  {
    height: 15mm;
    page: left;
    region: bottom;
    text-align: left;
  }

  div.bottom-right
  {
    height: 15mm;
    page: right;
    region: bottom;
    text-align: right;
  }

  span.page:before { content: counter(page); }
}
```

This says that on left pages the bottom region is left-aligned, while on right pages it is right-aligned. The `span` element is used in the following region definitions:

```
<div class="bottom-left">
  <p> </p>
  <div><span class="page"/></div>
</div>

<div class="bottom-right">
  <p> </p>
  <div><span class="page"/></div>
</div>
```

In order for the region elements not to interfere with the normal flow it is best to set their display type to `none`.

Page Numbering

The two special counters `page` and `pages` in this tool are taken over from the CSS3 Paged Media Module (see also [CSS3P]). The `page` can be used just like any other counter, except that it is confined to the regions. The following example shows a document with a preface and a body. Each reset the page count. The preface has a lower Roman numbering style, while the body uses the decimal style. If the bottom region didn't reset the counter, numbering would continue from the preface, but with a change of style.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title></title>
    <style type="text/css">
@page preface { margin: 10% }
@page body { margin: 10% }

div.bottom-preface
{
  display: none;
  page: preface;
  region: bottom;
}

div.bottom-body
{
  display: none;
  page: body;
  region: bottom;
}

div.bottom-preface > span.page:before
{
  counter-reset: page;
  content: counter(page, lower-roman);
}

div.bottom-body > span.page:before
{
  counter-reset: page;
  content: counter(page, decimal);
}

div.preface { page: preface; }
div.body { page: body; }
    </style>
  </head>
  <body>
    <div class="bottom-preface"><span class="page"/></div>
    <div class="bottom-body"><span class="page"/></div>
    <div class="preface">
      <p>Text.</p>
    </div>
    <div class="body">
      <p>Text.</p>
    </div>
  </body>
</html>
```

Page References

You sometimes want to write phrases like “The diagram on page 19 ...”. The CSSToXSLFO tool provides this functionality through the `page-ref` function, which can be used in the `content` property. Its only parameter is the name of an attribute that contains the ID of another element. The function call will be replaced with the number of the page that element is on.

In XHTML it is a bit more complicated to achieve the desired result, because there aren't many extension attributes available used for it. The following fragment shows how it can be done:

```
<img id="img1" src="file:///t.png"/>
...
<span class="page-ref"><span class="img1"/></span>
```

The accompanying style sheet rule would then be:

```
span.page-ref > span:before { content: page-ref(class); }
```

Leaders

It is possible to use XSLFO leaders through the display type `leader`. The properties defined in section 7.21 of [XSLFO] (“Leader and Rule Properties”) can be used in a CSS style sheet. If you want to create table of contents lines or something similar, you also need the XSLFO property `text-align-last`, described in section 7.15.10 of [XSLFO]. The following example shows how a table of contents line could be made in XHTML.

```
<div class="toc">
  <a href="#chapter1">Title of Chapter 1</a><span class="leader"/>
  <span class="page-ref"><span class="chapter1"/></span>
</div>
```

The piece of style sheet that goes with it is:

```
div.toc
{
  text-align-last: justify;
}

span.leader
{
  display: leader;
  leader-pattern: dots;
}

span.page-ref > span:before
{
  content: page-ref(class);
}
```

Named Strings

Named strings, as described in [CSS3G], are supported in CSSToXSLFO. This consists of the `string-set` property, with which contents can be captured, and the `string()` function. The latter can occur in the value of the `content` property. The `string-set` property accepts values which are similar to those of the `content` property. There is an additional keyword `contents`, which is replaced with the string value of the element carrying the `string-set` property.

The following is a simple XHTML example of how you can create a running header that refers to the current chapter.

```
<body>
  <div class="top">
    <span class="mark"/>
  </div>
  ...
  <h1>Chapter Title</h1>
  ...
</body>
```

Here is the bit of style sheet that does it:

```
div.top
{
  region: top;
  display: none;
}

div.top > span.mark:before { content: string(mark); }

h1 { string-set: mark contents; }
```

Hyphenation

Text can be hyphenated through the `hyphenate` property, which in inherited. The possible values are `true` and `false`. Hyphenation is turned on by default.

Footnotes

It is possible to produce numbered footnotes using the `footnote` property. The footnote counter can be reset by giving it the value `reset`. A footnote can be generated with the value `create`. Note that the display type of the element should be `inline`. The following example resets the counter per chapter. It also shows how a footnote could be handled on the screen and on paper respectively.

```
h1 { footnote: reset; }

@media print
{
  span.footnote
  {
    display: inline;
    footnote: create;
  }
}

@media screen
{
  span.footnote:after { content: ")"; }

  span.footnote:before { content: " (Note: "; }
}
```

Orientation

You can rotate text with the `orientation`. This only works for block-level elements. The possible values are 0, 90, 180, 270, -90, -180, -270. They represent the degrees in the counter-clockwise direction. The initial value is 0.

The CSS3 List Glyphs

The glyphs for the `list-style-type` property, as defined in [CSS3L], are implemented.

Multicolumn

The properties `column-count`, which must be strictly positive, and `column-gap`, which is a length, a multicolumn layout can be specified for a page. Both properties are allowed in an @page rule. As a consequence, if you want to switch between column modes, you have to switch pages as well.

Embedding In An Application

CSSToXSLFO has an API. It consists of one Java class: `be.re.css.CSSToXSLFOFIlter`. This is a subclass of `org.xml.sax.helpers.XMLFilterImpl`. The following is the Javadoc.

be.re.css

Class CSSToXSLFOFilter

```
java.lang.Object
extended by org.xml.sax.helpers.XMLFilterImpl
extended by be.re.css.CSSToXSLFOFilter
```

All Implemented Interfaces:

org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler, org.xml.sax.XMLFilter, org.xml.sax.XMLReader

---

public class **CSSToXSLFOFilter**
extends org.xml.sax.helpers.XMLFilterImpl

A filter that accepts an XML document and produces an XSLFO document.

Constructor Summary
`CSSToXSLFOFilter()`
`CSSToXSLFOFilter(java.net.URL baseUrl)`
`CSSToXSLFOFilter(java.net.URL baseUrl, java.net.URL userAgentStyleSheet)`
`CSSToXSLFOFilter(java.net.URL baseUrl, java.net.URL userAgentStyleSheet, java.util.Map userAgentParameters)`
`CSSToXSLFOFilter(java.net.URL baseUrl, java.net.URL userAgentStyleSheet, java.util.Map userAgentParameters, boolean debug)`
`CSSToXSLFOFilter(java.net.URL baseUrl, java.net.URL userAgentStyleSheet, java.util.Map userAgentParameters, org.xml.sax.XMLReader parent)`
`CSSToXSLFOFilter(java.net.URL baseUrl, java.net.URL userAgentStyleSheet, java.util.Map userAgentParameters, org.xml.sax.XMLReader parent, boolean debug)`
`CSSToXSLFOFilter(java.net.URL baseUrl, java.net.URL userAgentStyleSheet, org.xml.sax.XMLReader parent)`

Method Summary	
java.net.URL	**getBaseUrl**()
java.util.Map	**getParameters**()
java.net.URL	**getUserAgentStyleSheet**()
void	**parse**(org.xml.sax.InputSource input)
void	**parse**(java.lang.String systemId)
void	**setBaseUrl**(java.net.URL baseUrl)
void	**setParameters**(java.util.Map userAgentParameters)
void	**setParent**(org.xml.sax.XMLReader parent)
void	**setUserAgentStyleSheet**(java.net.URL userAgentStyleSheet)

Methods inherited from class org.xml.sax.helpers.XMLFilterImpl
characters, endDocument, endElement, endPrefixMapping, error, fatalError, getContentHandler, getDTDHandler, getEntityResolver, getErrorHandler, getFeature, getParent, getProperty, ignorableWhitespace, notationDecl, processingInstruction, resolveEntity, setContentHandler, setDocumentLocator, setDTDHandler, setEntityResolver, setErrorHandler, setFeature, setProperty, skippedEntity, startDocument, startElement, startPrefixMapping, unparsedEntityDecl, warning

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CSSToXSLFOFilter

```
public CSSToXSLFOFilter()
                 throws CSSToXSLFOException
```

CSSToXSLFOFilter

```
public CSSToXSLFOFilter(java.net.URL baseUrl)
                 throws CSSToXSLFOException
```

CSSToXSLFOFilter

```
public CSSToXSLFOFilter(java.net.URL baseUrl,
                        java.net.URL userAgentStyleSheet)
                 throws CSSToXSLFOException
```

CSSToXSLFOFilter

```
public CSSToXSLFOFilter(java.net.URL baseUrl,
                        java.net.URL userAgentStyleSheet,
                        java.util.Map userAgentParameters)
                 throws CSSToXSLFOException
```

CSSToXSLFOFilter

```
public CSSToXSLFOFilter(java.net.URL baseUrl,
                        java.net.URL userAgentStyleSheet,
                        java.util.Map userAgentParameters,
                        boolean debug)
                 throws CSSToXSLFOException
```

CSSToXSLFOFilter

```
public CSSToXSLFOFilter(java.net.URL baseUrl,
                        java.net.URL userAgentStyleSheet,
                        org.xml.sax.XMLReader parent)
                 throws CSSToXSLFOException
```

CSSToXSLFOFilter

```
public CSSToXSLFOFilter(java.net.URL baseUrl,
                        java.net.URL userAgentStyleSheet,
                        java.util.Map userAgentParameters,
                        org.xml.sax.XMLReader parent)
                 throws CSSToXSLFOException
```

CSSToXSLFOFilter

```
public CSSToXSLFOFilter(java.net.URL baseUrl,
                        java.net.URL userAgentStyleSheet,
                        java.util.Map userAgentParameters,
                        org.xml.sax.XMLReader parent,
                        boolean debug)
                 throws CSSToXSLFOException
```

Method Detail

getBaseUrl

```
public java.net.URL getBaseUrl()
```

getParameters

```
public java.util.Map getParameters()
```

getUserAgentStyleSheet

```
public java.net.URL getUserAgentStyleSheet()
```

parse

```
public void parse(org.xml.sax.InputSource input)
           throws java.io.IOException,
                  org.xml.sax.SAXException
```

Throws:

`java.io.IOException`

`org.xml.sax.SAXException`

---

parse

```
public void parse(java.lang.String systemId)
           throws java.io.IOException,
                  org.xml.sax.SAXException
```

Throws:

`java.io.IOException`

```
org.xml.sax.SAXException
```

setBaseUrl

```
public void setBaseUrl(java.net.URL baseUrl)
```

setParameters

```
public void setParameters(java.util.Map userAgentParameters)
```

setParent

```
public void setParent(org.xml.sax.XMLReader parent)
```

setUserAgentStyleSheet

```
public void setUserAgentStyleSheet(java.net.URL userAgentStyleSheet)
```

Building CSSToXSLFO

The tool comes with an ANT file. The default target only builds the css2xslfo.jar file. Then there are also the `xep`, `xsl` and `fop` targets, which produce css2xep.jar, css2xsl.jar and css2fop.jar respectively.

Special Provisions for XHTML

While the tool works for any XML vocabulary it does a number of things for XHTML specifically. Other vocabularies may be supported in the same way at some later stage. The items are the following:

- Non-CSS presentational hints are translated to the corresponding CSS rules, as prescribed in section 6.4.4 of [CSS2];
- The `lang` attribute is honored;
- Hyperlinks are recognized and translated in XSLFO links;
- The `link` element can be used to specify external style sheets;
- Style sheets can be embedded with the `style` element;
- The `style` attribute is honored;
- The `img` element is interpreted and processed;
- The `html-header-mark` user agent parameter is available;
- Named pages work for direct child elements of the `body` element;

- There is user agent style sheet for XHTML that cascades against the one in appendix A of [CSS2].

The User Agent Style Sheet For XHTML

```
@import "xhtml.css";

@media print
{
  a[href]
  {
    color: blue;
    text-decoration: none;
  }

  blockquote, dl, ol, p, ul
  {
    margin: 0.83em 0;
  }

  blockquote
  {
    margin-left: 3em;
    margin-right: 3em;
  }

  body
  {
    font-family: serif;
    padding: 0;
  }

  body:lang(da)
  {
    quotes: "\00BB" "\00AB";
  }

  body:lang(de-DE), body:lang(de-AT)
  {
    quotes: "\201E" "\201C" "\201A" "\2018"
  }

  body, body:lang(en), body:lang(es)
  {
    quotes: "\201C" "\201D" "\2018" "\2019";
  }

  body:lang(fr)
  {
    quotes: "\00AB " " \00BB" "\2039 " " \203A";
  }
```

```
body:lang(it)
{
  quotes: "\00AB " " \00BB";
}

body:lang(nl)
{
  quotes: "\201D" "\201D" "\2019" "\2019";
}

body:lang(no), bodylang:(pt), body:lang(de-CH)
{
  quotes: "\00AB" "\00BB" "\2039" "\203A"
}

body:lang(sv)
{
  quotes: "\00BB" "\00BB";
}

caption
{
  margin: 0.5em 0;
}

dt
{
  page-break-after: avoid;
}

h1
{
  font-size: 1.6em;
  margin-bottom: 0.7em;
  margin-top: 1.4em;
}

h2
{
  font-size: 1.3em;
  margin-bottom: 0.6em;
  margin-top: 1.2em;
}

h3
{
  font-size: 1.1em;
}

h3, h4
{
  margin-bottom: 0.5em;
  margin-top: 1em;
}
```

```
h1, h2, h3, h4, h5, h6
{
  hyphenate: false;
}

hr
{
  border: 0.1pt solid;
}

li
{
  margin-bottom: 0.8em;
  margin-top: 0.8em;
}

li p, li blockquote, li dl, li ol, li ul
{
  margin-bottom: 0.5em;
  margin-top: 0.5em;
}

li li
{
  margin-bottom: 0.5em;
  margin-top: 0.5em;
}

li li p, li li blockquote, li li dl, li li ol, li li ul
{
  margin-bottom: 0.3em;
  margin-top: 0.3em;
}

li li li
{
  margin-bottom: 0.4em;
  margin-top: 0.4em;
}

li li li p, li li li blockquote, li li li dl, li li li ol, li li li ul
{
  margin-bottom: 0.3em;
  margin-top: 0.3em;
}

li, p
{
  text-align: justify;
}

pre
{
```

```
    font-size: 0.85em;
  }

  ul
  {
    list-style-type: disc;
  }

  ol li ul, ul li ul
  {
    list-style-type: circle;
  }

  ol li ol li ul, ol li ul li ul, ul li ol li ul, ul li ul li ul
  {
    list-style-type: square;
  }

  q:after
  {
    content: close-quote;
  }

  q:before
  {
    content: open-quote;
  }

  span.section-number
  {
    padding-right: 1em;
  }

  sub
  {
    font-size: 0.72em;
    vertical-align: -25%;
  }

  sup
  {
    font-size: 0.72em;
    vertical-align: 25%;
  }
}
```

References

[CSS2]
: “Cascading Style Sheets, level 2, CSS2 Specification”, W3C Recommendation 12 May 1998, Bert Bos, Håkon Wium Lie, Chris Lilley, Ian Jacobs, http://www.w3.org/-TR/1998/REC-CSS2-19980512.

[CSS3G]
“CSS3 Generated and Replaced Content Module”, W3C Working Draft 14 May 2003, Ian Hickson, http://www.w3.org/TR/2003/WD-css3-content-20030514.
[CSS3L]
“CSS3 module: Lists”, W3C Working Draft 7 November 2002, Ian Hickson, Tantek Çelı k, http://www.w3.org/TR/2004/WD-css3-lists-20021107.
[CSS3P]
“CSS3 Paged Media Module”, W3C Candidate Recommendation 25 February 2004, Håkon Wium Lie, Jim Bigelow, http://www.w3.org/TR/2004/CR-css3-page-20040225.
[XHTML]
“XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)”, W3C Recommendation 26 January 2000, revised 1 August 2002, http://www.w3.org/TR/-2002/REC-xhtml1-20020801.
[XSLFO]
“Extensible Stylesheet Language (XSL), Version 1.0”, W3C Recommendation 15 October 2001, Sharon Adler, Anders Berglund, Jeff Caruso, Stephen Deach, Tony Graham, Paul Grosso, Eduardo Gutentag, Alex Milowski, Scott Parnell, Jeremy Richman, Steve Zilles, http://www.w3.org/TR/2001/REC-xsl-20011015/.

Copyright