AurigaDoc: User Guide

Author : Khurshidali Shaikh <kshaikh at aurigalogic dot com>
Company : Auriga Logic
Revision : $Revision: 30 $
Last Modified : $Date: 2005-10-11 15:50:39 +0530 (Tue, 11 Oct 2005) $

Table Of Contents

1. What Is AurigaDoc

AurigaDoc is a java-xml-xsl based documentation tool for writing xml documents and converting them to other formats like HTML(single and multi-page), DHTML(frame based html with a toc tree), RTF(Microsoft Word 97 and above), PDF, Postscript, Java Help and HTML Help.

The idea is to keep a single document source(as xml) and convert it to various formats formats using XSLT.

While converting the xml to html you have the option of either creating a single html file for the entire document or multiple html files where each file contains a single section in the document with links to other sections.

Since xml is becoming a standard documents written in xml are portable to various formats.

AurigaDoc is built using open source software like

- Xerces - A java based XML(Extensible Markup Language) parser from Apache.
- Xalan - A java based XSLT(Extensible Stylesheet Transformation) engine from Apache.
- FOP - A java based Formatting Object Implementation from Apache.
- Batik - A java based package for handling SVG(Scalable Vector Graphic) from Apache.
- JFOR - A java package for writing rtf from xsl-fo.
- Java Help 2.0 - For generating output in Java Help 2.0 format.
- Java Mail API - For generating MIME multipart/related message.
- Steady State Software's CSS2 Parser: For parsing css stylesheets and applying them to pdf/postscript output.
- XOM API: For resolving xi:include tags.

2. Output Formats

AurigaDoc supports the following output formats.

- **HTML**: Single HTML file.
- **Multi Page HTML**: Single HTML file for each section in the source document. Also has table of content file with links to each section.
- **DHTML**: Frame based HTML output with the left frame containing an expandable/collapsible tree of the sections hierarchy and the right frame containing the topic selected from the left.
- **MIME multipart/related Message**: Format meant mainly for emails. Consists a single file with all the html, images, css, etc composed into a MIME multipart/related format. You can even specify options to mail this message to one or more email address(es).
- **FO**: Formatting Object.
- **PDF**: Adobe's Portable Document Format.
- **Postscript**: Postscript format.
- **RTF**: Rich Text Format for viewing in Microsoft Word.
- **Java Help**: Multi-page html output with additional files required by Java Help like the helpset file, map file, toc file and a full text search database.
- **HTML Help**: Multi-page html with other files required by HTML Help Compiler like the html help content(.hhc) file, html help project(.hhp) file and html help index(.hhk) file. On windows where HTML Help Workshop is installed the compiled (.chm) file is also generated.
- **Oracle Help For Java (OHJ)**: Multi-page html output with additional files required by OHJ. These include the helpset file, map file and toc file. Requires OHJ jars for indexing and viewing the output. OHJ can be downloaded from http://otn.oracle.com/software/tech/java/help/index.html.

- **MAN**: Unix man page.

3. Installation

To install aurigadoc follow the install instructions for the platform you are using:-

3.1. Installation On Windows

To install aurigadoc on windows follow the given steps:-

a. Untar the aurigadoc distribution to a directory.

   This will create a folder *aurigadoc*

b. Set the *AURIGADOC_HOME* environment variable to the absolute path of the *aurigadoc* directory created above.

c. Add *aurigadoc/bin* directory to the *PATH* environment variable.

d. Open *aurigadoc/bin/aurigadoc.properties* in a text editor and set the *aurigadoc.home* property to the absolute path of the *aurigadoc* directory created above.

> **Note:** Use double slashes in the path like this
>
> aurigadoc.home=c:\\software\\aurigadoc

e. If you need to compile HTML Help Files automatically, install HTML Help Workshop from Microsoft's site and set the value of *chm_compiler.path* to the absolute path of the HTML Help Compiler(hhc.exe).

> **Note:** Use double slashes in the path like this
>
> chm_compiler.path=c:\\Program Files\\HTML Help Workshop\\hhc.exe

3.2. Installation On Unix

To install aurigadoc on windows follow the given steps:-

a. Untar the aurigadoc distribution to a directory.

   tar -xvzf aurigadoc.tar.gz

   This will create a folder *aurigadoc*

b. Set the *AURIGADOC_HOME* environment variable to the absolute path of the *aurigadoc* directory created above.

c. Add *aurigadoc/bin* directory to the *PATH* environment variable.

d. Open *aurigadoc/bin/aurigadoc.properties* in a text editor and set the *aurigadoc.home* property to the absolute path of the *aurigadoc* directory created above.

4. Usage

AurigaDoc has been modified to provide multiple interfaces to the converter. The AurigaDoc converter can be called both from the command line as well as from a java class depending on your needs.

4.1. Using From Command Line

AurigaDoc converter utility can be used from the command line by executing the script *aurigadoc.sh*(for unix) or *aurigadoc.bat*(for windows) with the appropriate arguments.

Usage: aurigadoc.sh(or aurigadoc.bat) COMMAND OPTIONS PARAMETERS

COMMAND:

- -h: Display this help and exit successfully.
- -v: Display version information and exit successfully.
- -pdf: Convert input file to pdf.
- -ps: Convert input file to ps(postscript).
- -fo: Convert input file to fo.
- -awt: View pdf output of input file in awt viewer.
- -html: Convert input file to output html file.
- -mht: Convert input file to single html file and compile it into a MIME multipart/related message format.
- -mhtml: Convert input file to multiple html files in specified output directory.
- -dhtml: Convert input file to multiple html files with a toc tree in specified output directory.
- -chm: Convert input file to multiple html files with the html help content file(.hhc) and html help project file (.hhp).

  If HTML Help Compiler is installed on the system and the compiler path is specified in aurigadoc.properties a compiled chm file is also generated.
- -jhelp: Convert input file to multiple html files with supporting files to make a java help.

  If indexing option is set to y an index is created.

  If view options is set to yes then the output is launched in a java help viewer.

- -ohj: Convert input file to multiple html files with supporting files needed by Oracle Help For Java (OHJ).

  If indexing option is set to y an index is created.

  If view options is set to yes then the output is launched in a java help viewer.

  This options requires OHJ jars to be installed in AurigaDoc lib directory. OHJ can be downloaded from http://otn.oracle.com/software/tech/java/help/index.html.
- -jhview: View the helpset specified by helpset name and the helpset dir in a java help viewer.
- -man: Convert input file to unix man source file.

OPTIONS:

- -XML <file-name>: The input xml file.
- -OUT <path>: The path of output file or directory.
- -HSNAME helpset-name: The helpset name without the extension. Required for -jhview option.
- -HSDIR <path>: The dir path where the helpset is located. Required for -jhview option.

PARAMETERS:

-PARAM name=value: Additional parameter to be passed to the converter.

The foll parameters are supported

- xsl=<xsl-path>: the xsl for generating html, pdf, fo, rtf, awt, etc.
- frameset_xsl=<xsl-path>: the xsl for generating the frameset page in dhtml output.
- tree_xsl=<xsl-path>: the xsl for generating the toc tree in dhtml output.
- hhc_xsl=<xsl-path>: the xsl for generating the html help content file when converting to chm.

- hhp_xsl=<xsl-path>: the xsl for generating the html help project file when converting to chm.
- helpset_xsl=<xsl-path>: the xsl for generating the help set file while converting to Java Help.
- toc_xsl=<xsl-path>: the xsl for generating the toc file while converting to Java Help.
- map_xsl=<xsl-path>: the xsl for generating the map file while converting to Java Help.
- index_files=[y|n]: whether to create an index of the html files while converting to Java Help.
- launch_viewer=[y|n]: whether to launch the output in a java help viewer while converting to Java Help.
- send_mail=[y|n]: param indicating whether to also send the composed MHT as mail. If this is set to y, the following parameters may also be specified.
- subject: the subject for the mail.
- to: the email address of the recipient. multiple emails should be separated by comma.
- from: the email address of the sender.
- cc: the email address for the Cc list. multiple emails should be separated by comma.
- bcc: the email address for the Bcc list. multiple emails should be separated by comma.
- Profiling Parameters

- profile.os=something : operating system
- profile.lang=something : language
- profile.arch=something : computer architecture
- profile.revision=something : revision
- profile.revisionflag=something : revision status

- profile.role=something : user role
- profile.security=something : security level like high, medium, low, etc
- profile.userlevel=something : user level like beginner, intermediate, expert, etc
- profile.vendor=something : product vendor

EXAMPLES:

- html conversion:

  ```
  aurigadoc.sh -html -XML foo.xml -OUT foo.html
  ```

- html conversion using a custom xsl:

  ```
  aurigadoc.sh -html -XML foo.xml -OUT foo.html -PARAM
  xsl=path-to-xsl
  ```

- java help conversion with indexing and view option:

  ```
  aurigadoc.sh -jhelp -XML foo.xml -OUT foo-jhelp-files -PARAM
  index_files=y -PARAM launch_viewer=y
  ```

- chm conversion with a custom xsl for generating the html help content file.

  ```
  aurigadoc.sh -chm -XML foo.xml -OUT foo-chm-files -PARAM
  hhc_xsl=path-to-xsl
  ```

4.2. Using From Java

AurigaDoc converter utility can be used from the a java class be following the steps given below:-

- Add all jars in AURIGADOC_HOME/lib to your classpath.
- Add the AURIGADOC_HOME/bin directory to your classpath.
- Invoke the AurigaDoc converter using the following code snippet

  A simple example

```
import com.aurigalogic.aurigadoc.core.Driver;
import com.aurigalogic.aurigadoc.logger.DefaultLogger;
.
.
Driver driver = new Driver(); // instantiate driver
driver.setInputFile(inputFile); // set input file path
driver.setOutputFile(outputFile); // set output file/dir path
driver.setFormat(Driver.FORMAT_HTML); // set output format
driver.setLogger(new DefaultLogger()); // set logger (optional)
driver.run(); // run the driver
```

  Example of invoking AurigaDoc with a custom xsl

```
import com.aurigalogic.aurigadoc.core.Driver;
import com.aurigalogic.aurigadoc.logger.DefaultLogger;
import com.aurigalogic.aurigadoc.code.HTMLConverter;
.
.
Driver driver = new Driver(); // instantiate driver
driver.setInputFile(inputFile); // set input file path
driver.setOutputFile(outputFile); // set output file/dir path
```

```
driver.setFormat(Driver.FORMAT_HTML); // set output format
driver.setLogger(new DefaultLogger()); // set logger (optional)

Properties params = new Properties(); // converter parameters
params.setProperty(HTMLConverter.XSL, "path-to-my-custom-xsl");
driver.setParameters(params);

driver.run(); // run the driver
```

4.3. Using From Ant Target

AurigaDoc conversion can be invoked from an Ant target by using the following: -

```
<java dir="output-dir"
        classname="com.aurigalogic.aurigadoc.cmdline.Converter"
        fork="true">
        <classpath>
                <dirset dir="aurigadochome/bin" />
                <fileset dir="aurigadochome/lib">
                        <include name="**/*.jar" />
                </fileset>
        </classpath>
        <arg line="-html -XML user-guide.xml -OUT user-guide.html" />
</java>
```

In the above code

output-dir is the absolute path of output directory

aurigadochome is the absolute path of AurigaDoc home directory.

4.4. Using AurigaDoc Ant Task

Starting from version 1.3b, AurigaDoc is distributed with an ant task which can be used from an ant target. The following steps should be followed to install and run the aurigadoc task.

Installation:

- First install AurigaDoc by following the steps above
- Put the following code in your project's build.xml file.

```
<taskdef resource="aurigadoctask.properties">
        <classpath>
                <dirset dir="AURIGADOC_HOME/bin" />
                <fileset dir="AURIGADOC_HOME/lib">
                        <include name="**/*.jar" />
                </fileset>
        </classpath>
</taskdef>
```

where `AURIGADOC_HOME` is the absolute path to the AurigaDoc installation directory.

Task Description:

Parameters

Attribute	Description	Required
format	The output format required. Valid values are: html, dhtml, mhtml, mht,fo, pdf, ps, awt, rtf, man, chm, jhelp.	Yes
input	The path of the input file.	Yes
output	The path of the output file/directory.	Yes, unless the format is awt.
force	Force conversion even if the output	No; defaults to false.

	file(s) are up-to date.	

Parameters specified as nested elements:

param: aurigadoc task supports the nested param element(s) with the following attributes.

Attribute	Description	Required
name	The parameter name.	Yes
value	The parameter value.	Yes

Examples:

- Converting a document to html

```
<aurigadoc format="html" input="src.xml"
        output="out.html" />
```

- Converting a document to pdf with a custom xsl option

```
<aurigadoc format="pdf" input="src.xml" output="out.pdf">
        <param name="xsl" value="custom.xsl" />
</aurigadoc>
```

- Converting a document to jhelp with index option

```
<aurigadoc format="jhelp" input="src.xml"
                output="out-dir">
        <param name="index_files" value="y" />
</aurigadoc>
```

Note: The aurigadoc ant task will do the conversion only if the output files generated by a conversion are older then the source xml file or the output file(s) are deleted. The output files that are monitored include only text, xml and html and not any images/javascript files generated while building the dhtml output.

5. AurigaDoc Documentation Conventions

Documents to be converted using AurigaDoc should have the following structure:-

- The root element of the document is *<document>*
- The meta info about the document is contained in *<document-meta-info>* element inside the document root. This element has elements like title, attribute, etc.
- The formatting info about the document is contained within *<document-formatting-info>* under the document root.

  The following elements are supported in this element:-
- *<stylesheet url="url" />* - Used to associate a css to the generated html. (**html only**)
- *<stylesheet-fo url="url" />* - Used to associate a css to the output (**pdf/postscript**)
- *<left-margin>Xpt</left-margin>* - Used to specify left margin for pdf document. (**pdf/postscript/rtf**)
- *<right-margin>Xpt</right-margin>* - Used to specify right margin for pdf document. (**pdf/postscript/rtf**)
- *<top-margin>Xpt</top-margin>* - Used to specify top margin for pdf document. (**pdf/postscript/rtf**)
- *<header-height>Xpt</header-height>* - Used to specify bottom height for the page header in pdf. (**pdf/postscript/rtf**)
- *<footer-height>Xpt</footer-height>* - Used to specify bottom height for the page footer in pdf. (**pdf/postscript/rtf**)
- *<bottom-margin>Xpt</bottom-margin>* - Used to specify bottom margin for pdf document. (**pdf/postscript/rtf**)
- *<body-font-family></body-font-family>* - Used to specify the font

name for the pdf document body. (*Removed. use css for this*)

- *<body-font-size></body-font-size>* - Used to specify the font size for the pdf document body. (*Removed. use css for this*)
- *<document-title-font-family></document-title-font-family>* - Used to specify the font family for document title. (*Removed. use css for this*)
- *<document-title-font-size></document-title-font-size>* - Used to specify the font size for document title. (*Removed. use css for this*)
- *<document-attributes-font-family></document-attributes-font-family>* - Used to specify the font family for document attributes. (*Removed. use css for this*)
- *<document-attributes-font-size></document-attributes-font-size>* - Used to specify the font size for document attributes. (*Removed. use css for this*)
- *<generate-section-numbers>yes or no</generate-section-numbers>* - Used to specify whether sections numbers should be generated for each section. Default is yes. (**pdf/postscript/rtf**)
- *<generate-toc-page>yes or no</generate-toc-page>* - Used to specify whether a toc page should be generated. Default is yes. (**pdf/postscript/rtf**)

- The data to be displayed as a document header can be specified in the *<document-header>* element.
- The *<document-body>* element contains the actual content of the document.
- The *<table-of-content>* element has a hierarchical tree of *<links>* elements defining the sections hierarchy in the document.
- The *<link>* element has a *href* attribute which specifies the *<section>* element to link to. The value of the *href* attribute should start with a # and should not contain white spaces.

The name/label of the section to link to is described as text of <*link*> element.

In order to define sections in a nested manner just define the sections hierarchy in the this section. The actual <section> should not be nested i.e. while writing the actual *section* tag there should not be nesting of *section* tag within it. For details see the nested-sections sample.

- The <*section*> element is used to define the contents for each section of the document. This element has two attributes.
- *name* - the name of the section. This should be same as the the href attribute (without the #) of the <*link*> element for this section under <*table-of-content*>.
- *label* - the label to be displayed as the section's name.

  The <*section*> can have text and any xml safe html elements.

- The <*index*> element is used to define the document index. An index can have a label attribute which is used to specify the label for the index page. The <index> tag has one or more <indexitem> elements.

  An <indexitem> must have a label attribute to specify the text to display and may have a href attribute containing the section name to link to. In absence of the href attribute the index item is displayed but it is not linked.

  An indexitem may have nested indexitem elements.

  The index is rendered smartly across different output formats.

  - for Java Help and HTML a index tab is created

  - for html, dhtml, mhtml an html index tree is created.

  - for pdf a two-column index page is created at the end of the document.

- The <*section-link*> element has a *href* attribute which specifies the <*section*> element to link to. This tag is basically used for giving links to a section from the document body. Using this tag to link to sections makes sure that the link is maintained irrespective of whether the

document is converted to pdf, single html or multi html format.

- The <*document-footer*> element contain the information to be displayed as page footer.
- The are a set of special elements which are used while converting to pdf format. These are:-
- <*page-break*> - This element forces a page break in the pdf document.
- <*page-number*> - Inserts the page number of the pdf document.
- <*total-pages*> - Inserts the total number of pages in the document. Useful for rendering something like this `Page 2 of 40`.

Any html tag can be used in the document but while converting to pdf/postscript/rtf only the following tags are supported.

1. *a*
2. *img*
3. *span*
4. *font*
5. *u*
6. *s*
7. *b/strong*
8. *i/em*
9. *h1-h6*
10. *br*
11. *hr*
12. *div*
13. *p*
14. *pre*
15. *code*
16. *blockquote*
17. *address*

18. *center*
19. *ol*
20. *ul*
21. *li*
22. *dl/dt/dd*
23. *table*
24. *tr*
25. *td*
26. *sub*
27. *sup*

6. CSS (Cascading Style Sheet) Support

Starting from version 1.3b AurigaDoc has been modified to depend heavily on css for formatting document. The appearance for all parts of the document can be controlled using css.

A css can be associated to a document by using the `<stylesheet url=".." />` element inside the `<document-formatting-info>` element.

Even pdf/postscript output can be formatted using a css file. The css file for these outputs can be specified in the `<stylesheet-fo url=".." />` element inside the `<document-formatting-info>` element. See the src (xml) document of this user guide for an example. Although support for css in pdf/postscript output is limited, using a css can greatly improve the look of the output document.

The css url may either be a:

- http:// url
- an absolute file system path
- or a file system path relative to the output directory.

In addition to specifying a css file, formatting for supported html elements can also be controlled by using the `style` attribute.

CSS stylesheet is not applied while creating the rtf due to limitations in jfor.

7. Profiling

Many times it is required to generate a slightly different version of the same document for different audiences. Previously the only way to achieve this using AurigaDoc was to make a copy of the original document for each variation and make the necessary changes. This was a maintenance headache as common changes in one had to be manually copied into others.

Starting from version v1.4.0-b AurigaDoc supports profiling of documents. With profiling, elements(sections, paragraphs, etc) can be marked as conditional. While actual output generation one or more conditions can be specified to generate the output with elements matching those conditions.

e.g. some sections can be given a os attribute like this `<section os="win"..` to indicate that it is intended for only windows operating systems. While generating output a parameter can be passed to specify a value for os like this `-PARAM profile.os=win`

With the above parameter only elements which have a os attribute value of win will be included in the output. Other supported attributes can be specified similarly.

Even multiple values can be specified for a attribute like this `<section os="win,windows"..`

With the above setting the element will be selected if the value of profile.os is win or windows

AurigaDoc supports a predefined set of profiling attributes. These are similar to those supported by DocBook. Listed below are the attributes and the name of corresponding command line parameters.

Attribute	Description	Command Line Param
os	the operating system	profile.os
lang	language	profile.lang
arch	computer architecture	profile.arch
revision	revision	profile.revision

revisionflag	revision status	profile.revisionflag
role	can be used for indicating a user role.	profile.role
security	security level like high, medium, low, etc	profile.security
userlevel	user level like beginner, intermediate, expert, etc	profile.userlevel
vendor	product vendor	profile.vendor
format	output format like html, mhtml, dhtml, pdf, ps, fo, jhelp, etc. The profile.format parameter value is set implicitly depending on the output format specified. A sample usage of this would be to have different page header, etc for pdf and html related outputs.	NONE. The parameter is implicitly passed.

Some of the situations where profiling is useful are:-

- Creating conditional sections for different platforms, user levels, etc.
- Utilize features provided in one output format which are not there in other.

  - e.g. Display custom links, buttons, etc in the html based format but not in pdf format.

  - e.g. Display secondary windows and pop-up windows in the Java Help output, using the Java Help features. For other html based output the same could be done using javascript/css.

  - e.g. Embed flash in html output but display a static image in pdf output.

8. Modularity

Its is often desired to break up a single big documents into parts for ease of maintenance and for code/markup reuse. Also sometimes it is required to include external files(text, source code, etc) into the document without having to format/copy it in the including document.

Prior to this this was not possible with AurigaDoc. Now this can be achieved using XInclude. AurigaDoc now handles inclusion of external documents using XOM (http://www.cafeconleche.org/XOM/).

In order to use XInclude rewrite the document tag like this.

```
<document xmlns:xi="http://www.w3.org/2001/XInclude">
```

An external xml document can be included in the source xml document like this

```
<xi:include href="path-to-external-file" />
```

An external non-xml file(like text or source file) can be included like this. Note the parse="text" attribute.

```
<xi:include href="url-to-external-file" parse="text" />
```

9. Change History

The following changes have been incorporated in the below mentioned versions:-

9.1. Version 1.4.0-b

The following changes have been incorporated in Version 1.4.0-b:-

- Modified the html related xsl templates to avoid using table for generating TOC.
- Fixed bug in xml2mhtml.xsl that causes exception on some systems. See tracker 922672
- Fixed bug in pdf and chm conversion on JDK 1.3.
- Added jaxp.jar to fix java.lang.NoClassDefFoundError: javax/xml/transform/Source on JDK 1.3.x. See tracker 868276
- Made aurigadoc.jar executable. It can be executed using `java -jar aurigadoc.jar` command.
- Added support for span tag in pdf.
- Modified pdf output to leave proper space before and after ul, ol and li. Also modified pdf output to apply the css for these elements properly.
- Improved pdf output to handle the list-style-image css attribute unordered list(ul).
- Fixed bug in the distribution. *icons* directory was missing from the distribution due to which Java Help and OHJ conversions were failing.
- Added support for including external file using XInclude. See Modularity for details.
- Added support for profiling. See Profiling for details.
- Passing `-param name=value` from the command line does not work on windows. This is because = cannot be used as an argument on the windows command prompt. This has been fixed.

  Now for running on windows ~ can be used instead of = to separate the name and value of a parameter like this `-param name~value`

  The = separator is still supported and works on *nix systems.

- Added support to define a document index. For pdf and html output the index is rendered at the end of the documentt. For Java Help, CHM and OHJ the index is rendered as a index file supported by the corresponding help system.
- Included javadoc in PDF format in the distribution using AurigaDoclet.
- Added support to read FOP configuration from a config file. A blank config file is located in conf/fop/ directory. The config file is useful for embedding custom fonts in pdf and also to control some FOP behaviour. More details about the FOP configuration is available here http://xmlgraphics.apache.org/fop/configuration.html

9.2. Version 1.3

The following changes have been incorporated in Version 1.3:-

- Modified Java Help Output to produce output in Java Help 2.0 format.
- Improved Java Help Output to use fancy icons.
- Extended MHT converter to even send the generated output as email to the specified email addresses. This requires an SMTP server for sending mail. The name of the SMTP server can be configured in *aurigadoc.properties*.
- Added beta support for new output format: Oracle Help for Java(OHJ).

  In order to use this option fully

- Download OHJ from http://otn.oracle.com/software/tech/java/help/index.html
- Install OHJ on your system.
- Copy the OHJ jars from the OHJ installation directory to AurigaDoc lib directory.

  In order to view a sample of the output go to AurigaDoc *docs/ohj/bin/* folder and execute view.sh or view.bat.

- Fixed bugs in the RTF converter when the output file without directory name is specified.
- Made the command line options case insensitive for convenience. Instead of specifying -XML, -OUT and -PARAM you can now even specify -xml, -out and -param respectively.

9.3. Version 1.3b

The following changes have been incorporated in Version 1.3b:-

- An Ant task has been built to call AurigaDoc from an ant script.
- Support for specifying fonts for various sections while converting to pdf/postscript format has been removed. AurigaDoc now uses css for even pdf/postscript output in addition to the html outputs. See css support for more details.
- The class names used in the css rules have been modified to confirm to the css naming conventions. People upgrading from older version of AurigaDoc will have to patch the css style sheet. Use the css style sheet shipped with AurigaDoc for example.
- A new output format(mht) has been added which converts the source document to a single html file and compiles it to a single file which can be viewed in IE 5 or above. This format is a MIME multipart/related message format which is meant for emails.
- Included the latest jfor version (V_0_7_2rc1). This seems to produce better rtf output. Also fixed the fo xsl to avoid outputting unnecessary characters which were causing extra lines to appear in the rtf output.
- Fixed problems in the aurigadoc.sh script encountered under cygwin.
- All the xsls have been rewritten to use standard XSLT 1.0 functions. An exception is the xml2mhtml.xsl which uses the xalan redirect extension for generating multiple output files.

9.4. Version 1.2

The following changes have been incorporated in Version 1.2:-

- Fixed some bugs. One of the important ones was file output stream that were left unclosed. This used to create problems on windows when the document is given to the HTML Help Compiler while converting to chm. The HTML Help Compiler was not being able to open files with unclosed streams. This used to happen for small documents only.
- Did some code cleanup.
- Modified xsls to disable the unnecessary namespace that were outputted in the generated html.
- Fixed documentation bugs.
- Incorporated new JFOR version(V0.7.1) in the distribution.

9.5. Version 1.2b

The following changes have been incorporated in version(1.2b):-

- The pdf/postscript/rtf outputs have been modified to include a toc page after the cover page. This toc page has the list of sections/sub-sections along with their page numbers and are hyper-linked to the respective sections.
- The multi-page generation process has been modified to generate multiple pages using the xalan redirect extension. Due to this the xml document is not broken into sub documents using jdom. This has improved the conversion speed for multi-page html and also removed the dependency of AurigaDoc on JDOM.
- Java API have been provided to be able to use AurigaDoc from a java class.
- A command line interface has been written which uses AurigaDoc Java API for conversion to various output formats. The shell scripts(for unix and windows) have been modified to call the command line interface.
- Options have been provided in the command line and java api to specify a custom xsl while converting to any output format.
- Separate shell script for windows 9x(aurigadoc_win9x.bat) and windows 2000(aurigadoc_win2k.bat) have been replaced with a single batch script aurigadoc.bat which is intended to run both on windows 9x and windows 2000.
- The build scripts to build AurigaDoc from source has been removed and replaced with an Ant build file(build.xml). In Order to build this version from source Apache Ant(http://jakarta.apache.org/ant/) will be required.
- A new option has been added to convert AurigaDoc document to fo.
- A new option has been added to convert AurigaDoc document to Postscript.

- A new option has been added to convert AurigaDoc document to Java Help. A Java Help Viewer has also been provided to view the Java Help output.
- A new option has been provided to convert AurigaDoc document to HTML Help format.

  On unix only the HTML Help Project(.hhp) file and HTML Help Content(.hhc) file are created.

  On windows systems if HTML Help Workshop is installed and the chm compiler path is set in aurigadoc.properties file, the compiled(.chm) file is also generated.
- A new option has been added to convert AurigaDoc document to man source file for generating unix man pages. This output format does not support tables, fonts, links and other advanced elements which cannot be rendered in a unix man page.

9.6. Version 1.1

The following changes have been incorporated in version(1.1):-

- A new option has been added in which AurigaDoc converts the given xml document to html files and also generates a dhtml toc tree for navigating though the sections.
- Added support to specify if the generated output should have section numbers generated. By default section numbers will be generated for sections. This can be turned of for a document by using the following <generate-section-numbers>no</generate-section-numbers> in the document-formatting-info element.
- Modified xsls remove hard coding of fonts in pdf. Now fonts for document-title, document-attributes, section-header, etc can be specified in the xml document itself.
- Made provision to use different font size/color for each section level. e.g. section 1 , 2, 3, etc can be use a common font size/color and sections section 1.1, 1.2, 1.3, 2.1, 2.2, 2.3, etc can use a font size/color different than the above sections. For html this can be define in the stylesheet file and for pdf/rtf this can be modified by modifying the xsl(xml2pdf.xsl). See the css file and the xsl file for details.
- Added more samples to the samples directory.
- Removed jimi.jar from the distribution due to licensing restrictions. Due to this embedding png in pdf will not work. In order to overcome this download the jimi library from java.sun.com and copy the jimi jar it in the aurigadoc lib directory as jimi.jar.
- Made provision for specifying the bullet size for unordered lists while converting to pdf. The bullet size can be specified as attribute *bullet-size* of the <ul> element.

  e.g. <ul bullet-size="15pt" >.

10. Limitations

The current version of AurigaDoc has the following limitations:-

- Advanced table attributes like cellspacing are not supported while converting to pdf.
- The RTF output is limited in functionality and does not have support for nested list, tables, etc. Also the RTF output is targeted to be viewed using Microsoft Word(97 or above).

  The RTF output is generated using jfor which has its own limitations. Due to this the RTF output generated may not be as good as the pdf equivalent.

11. Troubleshooting

The following things should be checked in case there is a problem running AurigaDoc.

- Make sure you have set the *AURIGADOC_HOME* environment variable and the aurigadoc.home property in aurigadoc.properties to the aurigadoc installation directory. This will mostly solve your problems. If your problem is still not solved read ahead.
- Make sure you have appended *AURIGADOC_HOME/bin* to your *PATH* environment variable.
- Older versions of AurigaDoc (v1.0b to v1.0.2b) used to append the existing CLASSPATH to its own dynamically generated classpath while launching AurigaDoc. In case you are using these versions make sure that your CLASSPATH is not broken because of some characters like space, etc.
- Make sure you do not have another version of xerces, xalan, fop in your jre/lib/ext dir.
- If you encounter an error while invoking AurigaDoc with a custom xsl PARAM specify the PARAM like this

  aurigadoc.bat format -XML xml-file -OUT out-file -PARAM xsl~my-xsl

  Note: use ~ as a separator.

12. License

AurigaDoc is available under the GPL License recognized by The Open Source Initiative.

For alternative licensing, contact sales at aurigalogic dot com

For more info on AurigaDoc usage, see the docs directory in the distribution or mail *kshaikh at aurigalogic dot com*

INDEX