odt2braille Developer Guide

for version 0.0.1, 9 August 2010

by Bert Frees

Table of Contents

Introduction

1 Hacking

System Requirements

In order to hack odt2braille, you need the following software:

OpenOffice.org

The latest version of the OpenOffice.org office suite can be downloaded here.

OpenOffice.org Software Development Kit

The OpenOffice.org SDK is an add-on for OpenOffice.org. It provides the necessary tools and documentation for programming the OpenOffice.org APIs and creating own extensions. The latest version of OpenOffice.org SDK can be downloaded here. Make sure you install OpenOffice.org first. The version of OpenOffice.org SDK should be equal or lower than the version of OpenOffice.org.

NetBeans

Odt2braille was developed in NetBeans, an integrated development environment. The latest version of NetBeans can be downloaded here. Make sure the Java SE pack is included.

OpenOffice.org API Plugin for NetBeans

To install the OOo API Plugin for NetBeans, launch NetBeans, go to [*Tools* > *Plugins*], then select *OpenOffice.org API Plugin* in *Available Plugins* and click on *Install*. Now relaunch NetBeans and configure the plugin in [*Tools* > *Options* > *Miscellaneous* > *OOo API Plugin*] by selecting the appropriate OpenOffice.org installation and OpenOffice.org SDK folder.

[@TODO] NetBeans with OpenOffice.org API plugin may be replaced by Eclipse with OOEclipse plugin.

Odt2braille Source Code

The source code is available on Sourceforge and can be downloaded directly into NetBeans as follows:

- [*Team - Subversion - Checkout*]
- *Repository URL*: "https://odt2braille.svn.sourceforge.net/svnroot/odt2braille". Leave "*User*" and "*Password*" fields blank.
- *Next* >
- *Repository Folders*: "DaisyPipeline, Odt2Braille, Odt2BrailleAddOn"
- Check "*Scan for NetBeans Projects after checkout*".
- *Finish*

The source code can also be downloaded without NetBeans:

- `svn co https://odt2braille.svn.sourceforge.net/svnroot/odt2braille path/to/odt2braille/folder/`

Build

Build projects with NetBeans:

- Open "*Projects*" window, than right click a project and choose menu item "*Build*".

Build projects with Apache ANT:

- `cd` to project directory
- `ant`

Create .oxt file with NetBeans:

- Open "*Projects*" window, than right click "*Odt2BrailleAddOn*" and choose menu item "*Create OXT*" or "*Deploy and Run Extension in OpenOffice.org*".
- The .oxt file will be created in Odt2BrailleAddOn/dist.

Create .oxt file with Apache ANT:

- `cd` to Odt2BrailleAddOn directory
- `ant uno-package`
- The .oxt file will be created in Odt2BrailleAddOn/dist.

Documentation

Texinfo:

- `cd` to Odt2BrailleAddOn/doc

-
  ```
  texi2dvi --pdf
  odt2braille-user-doc.texi
  ```

-
  ```
  makeinfo --html
  [--no-split]
  [--no-headers]
  --output=odt2braille-user-doc.html
  odt2braille-user-doc.texi
  ```

Javadoc:

- `cd` to directory containing all projects

-
  ```
  javadoc [-private]
  [-author]
  [-version]
  [-breakiterator]
  -d Odt2BrailleAddOn\doc\javadoc
  -subpackages be.docarch:org_pef_text
  -sourcepath Odt2BrailleAddOn\src;Odt2Braille\src;DaisyPipeline\src
  ```

Downloads

- Apache Subversion: `http://subversion.apache.org/`
- Apache Ant (1.7.1 or higher): `http://ant.apache.org/`
- Javadoc: `http://java.sun.com/j2se/javadoc/downloads/index.html#findjavadoc`
- Texinfo: `http://www.gnu.org/software/texinfo/`

2 Projects

This is an overview of the projects that are created in NetBeans after odt2braille has been succesfully checked out.

Odt2BrailleAddOn

Odt2BrailleAddOn is the OpenOffice.org extension. It makes extensive use of the OpenOffice.org UNO API. It takes care of the graphical user interface (menu's, dialogs, progress bars, etc.) and allows for loading settings from and saving settings to OpenOffice.org or the OpenOffice.org Writer document. For the actual document processing, Odt2BrailleAddOn relies on the Odt2Braille library.

Packages in Odt2BrailleAddOn:

`be.docarch.odt2braille.addon`

Odt2Braille

The Odt2Braille library takes care of the actual document processing. It enables the conversion of a flat xml odt file to a pef (Portable Embosser Format) file. Furthermore, this pef file can be converted to a variety of other generic braille formats, or it can be converted to an embosser-specific braille file (and optionally sent to an embosser device). The braille transcription is powered by liblouisxml, and for the pef processing, Odt2Braille uses the DaisyPipeline library. The Java OpenDocument Library (JODL) is used for creating and cleaning the flat .odt file (see `http://odt2daisy.sourceforge.net/downloads/`).

Packages in Odt2Braille:

`be.docarch.odt2braille`

DaisyPipeline

The DaisyPipeline project contains the `org_pef_text` and `org_pef_text.pef2text` packages. These packages have been adopted from the DAISY Pipeline and have been slightly modified. Their purpose is to convert a PEF 2008-1 document into a plain text braille file (see also the `pef2text` documentation).

Packages in DaisyPipeline:

`org_pef_text` and `org_pef_text.pef2text`

3 Liblouisxml

Liblouisxml is the heart of the braille transcription. It is an open-source library intended to provide complete Braille transcription services for XML documents. Liblouisxml is built on top of liblouis, its translation engine. The translation is driven through text based translation tables which define the translation rules. The formatting of braille is defined in semantic mappings that define how a specific XML input tag is to be rendered in the Braille output. Liblouisxml is embedded in odt2braille as an executable. It can be found in the Odt2BrailleAddOn project under `liblouis/bin`. The translation tables and configuration files are kept under `liblouis/share`.

For more information read the `liblouisxml` and `libouis` manuals.

4 Tutorials

Adding an embosser

To add support for an embosser, the protocol for giving print instructions to that embosser has to be known. This includes e.g.

- the configuration of the header which initiates the print job and gives general printing information,
- the way Braille pages and Braille lines are represented,
- the way each Braille character is represented (the character set),
- the footer which ends the print job, etc.

In addition, you should know the dimensions of cell spacing and line spacing, whether the embosser can print interpoint (duplex), which paper sizes are supported, etc.

Once the protocol is known, it can be implemented. What follows is a more or less general way of adapting the code in order to add an embosser. But because each embosser is different, extra adjustments may have to be made.

- In `org_pef_text.pef2text.EmbosserFactory`, expand the `EmbosserType` enumeration with a new embosser type. Add a `case` for this new embosser type to the `switch` statement in the `newEmbosser` function.
- Possibly, a new character set may have to be defined in `org_pef_text.TableFactory` as well. Edit the `TableType` enumeration and the `newTable` function.
- Finally, in `be.docarch.odt2braille.Settings`, the functions `embosserIsSupported`, `changeEmbosser`, `tableIsSupported`, `paperSizeIsSupported`, `changePaperSize`, `getMaxPaperWidth`, `getMaxPaperHeight`, `getMinPaperWidth`, `getMinPaperHeight` and `duplexIsSupported` need adjustments.

5 Task list

- More flexibility by providing more settings!
  - Settings for footnotes, endnotes, TNs, bibliography, ...
  - Volume info and transcription info.
  - 'Continued' suffix.
  - Border lines of tables & textboxes ??? => linesAbove/linesBelow (integer) uitbreiden met borderAbove/borderBelow (string) -> Aantal karakters in string = aantal lijnen in border -> n-de karakter = braille symbool waarmee n-de border lijn gevuld wordt
  - Table Of Contents: - headings tot level ? weergeven - printpagina nummers weergeven - braillepagina nummers weergeven
  - ...
- Mac OS, Linux, ... !!!
- Add more "Braille formatting standards" (now: only BANA) (eg. also UEBC)
- Documentation!
- Support more embossers.
- Only Interpoint55 is tested.
- Tooltips in dialogs.
- Keyboard shortcut for Braille menu.
- Howto handle unknown characters? (-> now: dots 3456) (-> "undefined" opcode ?)
- Hidden paragraphs (or paragraphs in hidden sections) => transcriber's notes (braille-only material). TN => settings!
- 8 dot Braille.
- Tactile graphics.
- If the liblouisxml process takes to long (e.g. with Chinese), OpenOffice.org might think the program is not responding.
- OpenOffice.org accessibility for screen readers (on Windows)?
- OCR plugin for OpenOffice.org?
- http://www.thessalonica.org.ru/en/index.html ?
- Check if the number of cells per line (and the number of lines per page) is sufficient (if too small, this might cause liblouisxml to fail).
- Endnotes in preliminary pages: fix bug.
- makePEF(): should be made more robust.
- Odt2Daisy.preProcessing(): should be made faster. Replace DOM processing with XSL transformation where this is possible. This will result in better performance. (Vincent Spiewak is rewriting JODL in XSLT 2.0)
- When multi-volume feature of liblouisxml is finished: Odt2Braille code might be simplified.
- Jlouis ?
- UTD & liblouisutdml?

- Presentations & spreadsheets.
- Commandline tool (Odt2Braille.jar).
- Er kunnen nog lege paragrafen voorkomen (als er lege 'span's inzaten: bv door special typeface / languages) Door 'clear formatting' toe te passen op deze paragrafen is dit probleem weg.
- Uitleg van WinBraille over g0,g1,g2,... * g0 = one to one * g1 = literary one to one with capital and figure prefix * g2,g3,g4,... = contracted braille
- Wanneer de instelling "export/ emboss naar specifieke brailleprinter" is opgeslagen => OOo crasht (Windows XP) bij oproepen van dialoogvenster => is dit nog steeds zo?
- Voor Ubuntu: OOo 3.2.1 nodig ?
- Settings in Tools > Options... ?
- Zoveel mogelijk originele tabellen includen en sommige default bestanden (tables) overschrijven met eigen (files)
- ...