OpenDocument Text Exporter for Emacs' Org Mode

Jambunathan K

1 Project Summary

Description	The Authoritative fork of Org mode's ODT exporter
Version	
Depends on	
Suggests	
Enhances	
Published	
Author	Jambunathan K <kjambuanthan at gmail.com>
Maintainer	Jambunathan K <kjambuanthan at gmail.com>
BugReports	`https://github.com/kjambunathan/org-mode-ox-odt/issues`
License	
URL	`https://github.com/kjambunathan/org-mode-ox-odt`
Downloads	

ELPA URL
: `https://kjambunathan.github.io/elpa/`

User Manual (online HTML)
: `https://kjambunathan.github.io/org-mode-ox-odt/`

User Manual (PDF)
: `https://kjambunathan.github.io/org-mode-ox-odt/org-odt.pdf`

2 Getting Started with ODT export

2.1 Pre-requisites for ODT export

The ODT backend depends on the following programs

Program	**Purpose**
‘`zip`’[1]	To produce OpenDocument files
‘`unzip`’[2]	To unzip custom styles
‘`identify`’[3]	To identify the size of an inline image
‘`latex`’[4]	To compile LaTeX fragments to ‘`dvi`’ images
‘`dvisgm`’[5]	To convert ‘`dvi`’ images to ‘`svg`’
‘`dvipng`’[6]	To convert ‘`dvi`’ images to ‘`png`’
‘`convert`’[7]	To convert inline ‘`pdf`’ to ‘`png`’
‘`latexmlmath`’[8]	To convert LaTeX math snippets to MathML
‘`mathtoweb`’[9]	
‘`jabref`’[10]	To handle bibliography and citations

Of these, ‘`zip`’ is essential. Rest are optional.

2.2 Installation

You can install the OpenDocument Text export backend using the Emacs package manager. The archive URL for the package is `https://kjambunathan.github.io/elpa/`.

A typical configuration look like this

```
(custom-set-variables
 '(package-archives
   (quote
    (("gnu" . "https://elpa.gnu.org/packages/")
     ("ox-odt" . "https://kjambunathan.github.io/elpa/")))))
```

In the ‘`*Packages*`’ buffer, packages from this archive show up as below

```
JabrefExportChicagoODF 1.2.2      ... Jabref Plugin for export to Chicago Manual of St
ox-odt                 9.2.6.263  ... OpenDocument Text Exporter for Org Mode
```

[1] Info-ZIP

[2] Info-ZIP

[3] ImageMagick

[4] TeX Live

[5] TeX Live

[6] dvipng

[7] ImageMagick

[8] LaTeXML

[9] MathToWeb

[10] JabRef

2.3 Configuration

Here is a sample configuration.

```
(custom-set-variables
 '(org-odt-convert-process "LibreOffice")
 '(org-odt-preferred-output-format "docx")
 '(org-odt-transform-processes
   '(("Optimize Column Width of all Tables"
      "soffice" "--norestore" "--invisible" "--headless"
      "macro:///OrgMode.Utilities.OptimizeColumnWidth(%I)")
     ("Update All"
      "soffice" "--norestore" "--invisible" "--headless"
      "macro:///OrgMode.Utilities.UpdateAll(%I)")
     ("Reload"
      "soffice" "--norestore" "--invisible" "--headless"
      "macro:///OrgMode.Utilities.Reload(%I)")))
 '(org-jabref-command '("jabref" "-n" "true"))
 '(org-latex-to-mathml-convert-command "java -jar %j -unicode -force -df %o %I")
 '(org-latex-to-mathml-jar-file
   "/home/kjambunathan/src/org-mode-ox-odt/contrib/odt/mathtoweb/mathtoweb.jar"))

(setcdr (assq 'system org-file-apps-defaults-gnu) "xdg-open %s")

(require 'ox-jabref)
```

Above configuration sets up the ODT backend as follows:

1. Use ‘`"LibreOffice"`’ (i.e., ‘`soffice`’ executable) as the document converter
2. Generate a ‘`docx`’ document for every ‘`odt`’ document
3. Process the ‘`odt`’ document with a set of LibreOffice Basic Macros to
   - Optimize the column width of all tables
   - Update cross-references, table of contents etc.
   - (if you are already viewing a past version of a ‘`odt`’ file), re-load the new file in the same application window.
4. Tell where your ‘`JabRef`’ and ‘`mathtoweb`’ executables are located, and how they are invoked.
5. (if you are using a GNU system) open the ‘`odt`’ document with your preferred OpenDocument viewer, presumably ‘`LibreOffice`’.
6. Load ‘`ox-jabref`’ so as to produce documents with bibliography and citations .

3 ODT export commands

3.1 Exporting to ODT

C-c C-e o o (‘`org-odt-export-to-odt`’)

Export as OpenDocument Text file.

If ‘`org-odt-preferred-output-format`’ is specified, automatically convert the exported file to that format. See Section 4.1 [Automatically exporting to other formats], page 5.

C-c C-e o O

Export as OpenDocument Text file and open the resulting file.

If ‘`org-odt-preferred-output-format`’ is specified, open the converted file instead. See Section 4.1 [Automatically exporting to other formats], page 5.

4 Extending ODT export

The ODT exporter can interface with a variety of document converters and supports popular converters out of the box. As a result, you can use it to export to formats like ‘`doc`’ or convert a document from one format (say ‘`csv`’) to another format (say ‘`ods`’ or ‘`xls`’).

If you have a working installation of LibreOffice, a document converter is pre-configured for you and you can use it right away. If you would like to use ‘`unoconv`’ as your preferred converter, customize the variable ‘`org-odt-convert-process`’ to point to ‘`unoconv`’. You can also use your own favorite converter or tweak the default settings of the LibreOffice and ‘`unoconv`’ converters. See Section 12.1 [Configuring a document converter], page 15.

4.1 Automatically exporting to other formats

Very often, you will find yourself exporting to ODT format, only to immediately save the exported document to other formats like ‘`doc`’, ‘`docx`’, ‘`rtf`’, ‘`pdf`’ etc. In such cases, you can specify your preferred output format by customizing the variable ‘`org-odt-preferred-output-format`’. This way, the export commands (see Section 3.1 [Exporting to ODT], page 4) can be extended to export to a format that is of immediate interest to you.

4.2 Converting between document formats

There are many document converters in the wild which support conversion to and from various file formats, including, but not limited to the ODT format. LibreOffice converter, mentioned above, is one such converter. Once a converter is configured, you can interact with it using the following command.

`M-x org-odt-convert`

Convert an existing document from one format to another. With a prefix argument, also open the newly produced file.

5 Applying custom styles

The ODT exporter ships with a set of OpenDocument styles (see Section 12.2 [Working with OpenDocument style files], page 15) that ensure a well-formatted output. These factory styles, however, may not cater to your specific tastes. To customize the output, you can either modify the above styles files directly, or generate the required styles using an application like LibreOffice. The latter method is suitable for expert and non-expert users alike, and is described here.

5.1 Applying custom styles - the easy way

1. Create a sample '`example.org`' file with the below settings and export it to ODT format.

   ```
   #+OPTIONS: H:10 num:t
   ```

2. Open the above '`example.odt`' using LibreOffice. Use the Stylist to locate the target styles - these typically have the '`Org`' prefix - and modify those to your taste. Save the modified file either as an OpenDocument Text ('`.odt`') or OpenDocument Template ('`.ott`') file.
3. Customize the variable '`org-odt-styles-file`' and point it to the newly created file. For additional configuration options see Section 12.2.2 [Overriding factory styles], page 16.

   If you would like to choose a style on a per-file basis, you can use the '`#+ODT_STYLES_FILE`' option. A typical setting will look like

   ```
   #+ODT_STYLES_FILE: "/path/to/example.ott"
   ```

   or

   ```
   #+ODT_STYLES_FILE: ("/path/to/file.ott" ("styles.xml" "image/hdr.png"))
   ```

5.2 Using third-party styles and templates

You can use third-party styles and templates for customizing your output. This will produce the desired output only if the template provides all style names that the '`ODT`' exporter relies on. Unless this condition is met, the output is going to be less than satisfactory. So it is highly recommended that you only work with templates that are directly derived from the factory settings.

6 Links in ODT export

ODT exporter creates native cross-references for internal links. It creates Internet-style links for all other links.

A link with no description and destined to a regular (un-itemized) outline heading is replaced with a cross-reference and section number of the heading.

A '`\ref{label}`'-style reference to an image, table etc. is replaced with a cross-reference and sequence number of the labeled entity. See Chapter 10 [Labels and captions in ODT export], page 13.

7 Tables in ODT export

Export of native Org mode tables (See Section “Tables” in `org`) and simple ‘`table.el`’ tables is supported. However, export of complex ‘`table.el`’ tables - tables that have column or row spans - is not supported. Such tables are stripped from the exported document.

By default, a table is exported with top and bottom frames and with rules separating row and column groups (See Section “Column Groups” in `org`). Furthermore, all tables are typeset to occupy the same width. If the table specifies alignment and relative width for its columns (See Section “Column Width and Alignment” in `org`) then these are honored on export.[1]

You can control the width of the table by specifying ‘`:rel-width`’ property using an ‘`#+ATTR_ODT`’ line.

For example, consider the following table which makes use of all the rules mentioned above.

```
#+ATTR_ODT: :rel-width 50
| Area/Month    |   Jan |   Feb |   Mar |   Sum |
|---------------+-------+-------+-------+-------|
| /             |     < |       |       |     < |
| <l13>         |  <r5> |  <r5> |  <r5> |  <r6> |
| North America |     1 |    21 |   926 |   948 |
| Middle East   |     6 |    75 |   844 |   925 |
| Asia Pacific  |     9 |    27 |   790 |   826 |
|---------------+-------+-------+-------+-------|
| Sum           |    16 |   123 |  2560 |  2699 |
```

On export, the table will occupy 50% of text area. The columns will be sized (roughly) in the ratio of 13:5:5:5:6. The first column will be left-aligned and rest of the columns will be right-aligned. There will be vertical rules after separating the header and last columns from other columns. There will be horizontal rules separating the header and last rows from other rows.

If you are not satisfied with the above formatting options, you can create custom table styles and associate them with a table using the ‘`#+ATTR_ODT`’ line. See Section 12.4 [Customizing tables in ODT export], page 17.

[1] See MathToWeb

8 Images in ODT export

8.1 Embedding images

You can embed images within the exported document by providing a link to the desired image file with no link description. For example, to embed '`img.png`' do either of the following:

```
[[file:img.png]]
[[./img.png]]
```

8.2 Embedding clickable images

You can create clickable images by providing a link whose description is a link to an image file. For example, to embed a image org-mode-unicorn.png which when clicked jumps to `http://Orgmode.org` website, do the following

```
[[http://orgmode.org][./org-mode-unicorn.png]]
```

8.3 Sizing and scaling of embedded images

You can control the size and scale of the embedded images using the '`#+ATTR_ODT`' attribute.

The exporter specifies the desired size of the image in the final document in units of centimeters. In order to scale the embedded images, the exporter queries for pixel dimensions of the images using one of a) ImageMagick's identify program or b) Emacs '`create-image`' and '`image-size`' APIs.[1] The pixel dimensions are subsequently converted in to units of centimeters using '`org-odt-pixels-per-inch`'. The default value of this variable is set to '`display-pixels-per-inch`'. You can tweak this variable to achieve the best results.

The examples below illustrate the various possibilities.

Explicitly size the image

To embed '`img.png`' as a 10 cm x 10 cm image, do the following:

```
#+ATTR_ODT: :width 10 :height 10
[[./img.png]]
```

Scale the image

To embed '`img.png`' at half its size, do the following:

```
#+ATTR_ODT: :scale 0.5
[[./img.png]]
```

Scale the image to a specific width

To embed '`img.png`' with a width of 10 cm while retaining the original height:width ratio, do the following:

```
#+ATTR_ODT: :width 10
[[./img.png]]
```

[1] Use of ImageMagick is only desirable. However, if you routinely produce documents that have large images or you export your Org files that has images using a Emacs batch script, then the use of ImageMagick is mandatory.

Scale the image to a specific height

To embed ‘`img.png`’ with a height of 10 cm while retaining the original height:width ratio, do the following

```
#+ATTR_ODT: :height 10
[[./img.png]]
```

8.4 Anchoring of images

You can control the manner in which an image is anchored by setting the ‘`:anchor`’ property of it’s ‘`#+ATTR_ODT`’ line. You can specify one of the the following three values for the ‘`:anchor`’ property - ‘`"as-char"`’, ‘`"paragraph"`’ and ‘`"page"`’.

To create an image that is anchored to a page, do the following:

```
#+ATTR_ODT: :anchor "page"
[[./img.png]]
```

9 Math formatting in ODT export

The ODT exporter has special support for handling math.

9.1 Working with LaTeX math snippets

LaTeX math snippets (See Section "LaTeX fragments" in `org`) can be embedded in the ODT document in one of the following ways:

1. MathML

   This option is activated on a per-file basis with

   ```
   #+OPTIONS: LaTeX:t
   ```

   With this option, LaTeX fragments are first converted into MathML fragments using an external LaTeX-to-MathML converter program. The resulting MathML fragments are then embedded as an OpenDocument Formula in the exported document.

   You can specify the LaTeX-to-MathML converter by customizing the variables '`org-latex-to-mathml-convert-command`' and '`org-latex-to-mathml-jar-file`'.

   If you prefer to use MathToWeb[1] as your converter, you can configure the above variables as shown below.

   ```
   (setq org-latex-to-mathml-convert-command
         "java -jar %j -unicode -force -df %o %I"
         org-latex-to-mathml-jar-file
         "/path/to/mathtoweb.jar")
   ```

   You can use the following commands to quickly verify the reliability of the LaTeX-to-MathML converter.

   `M-x org-export-as-odf`

   Convert a LaTeX math snippet to OpenDocument formula ('`.odf`') file.

   `M-x org-export-as-odf-and-open`

   Convert a LaTeX math snippet to OpenDocument formula ('`.odf`') file and open the formula file with the system-registered application.

2. PNG images

   This option is activated on a per-file basis with

   ```
   #+OPTIONS: LaTeX:dvipng
   ```

   With this option, LaTeX fragments are processed into PNG images and the resulting images are embedded in the exported document. This method requires that the dvipng program be available on your system.

9.2 Working with MathML or OpenDocument formula files

For various reasons, you may find embedding LaTeX math snippets in an ODT document less than reliable. In that case, you can embed a math equation by linking to its MathML ('`.mml`') source or its OpenDocument formula ('`.odf`') file as shown below:

[1] See MathToWeb

```
[[./equation.mml]]
```

or

```
[[./equation.odf]]
```

10 Labels and captions in ODT export

You can label and caption various category of objects - an inline image, a table, a LaTeX fragment or a Math formula - using '#+LABEL' and '#+CAPTION' lines. See Section "File Archives" in emacs. ODT exporter enumerates each labeled or captioned object of a given category separately. As a result, each such object is assigned a sequence number based on order of it's appearance in the Org file.

In the exported document, a user-provided caption is augmented with the category and sequence number. Consider the following inline image in an Org file.

```
#+CAPTION: Bell curve
#+LABEL:   fig:SED-HR4049
[[./img/a.png]]
```

It could be rendered as shown below in the exported document.

```
Figure 2: Bell curve
```

You can modify the category component of the caption by customizing the variable 'org-odt-category-strings'. For example, to tag all embedded images with the string 'Illustration' (instead of the default 'Figure') use the following setting.

```
(setq org-odt-category-strings
      '(("en" "Table" "Illustration" "Equation" "Equation")))
```

With this, previous image will be captioned as below in the exported document.

```
Illustration 2: Bell curve
```

11 Literal examples in ODT export

Export of literal examples (See Section “Literal examples” in `org`) with full fontification is supported. Internally, the exporter relies on ‘`htmlfontify.el`’ to generate all style definitions needed for a fancy listing.[1] The auto-generated styles have ‘`OrgSrc`’ as prefix and inherit their color from the faces used by Emacs ‘`font-lock`’ library for the source language.

If you prefer to use your own custom styles for fontification, you can do so by customizing the variable ‘`org-odt-create-custom-styles-for-srcblocks`’.

You can turn off fontification of literal examples by customizing the variable ‘`org-odt-fontify-srcblocks`’.

[1] Your ‘`htmlfontify.el`’ library must at least be at Emacs 24.1 levels for fontification to be turned on.

12 Advanced topics in ODT export

If you rely heavily on ODT export, you may want to exploit the full set of features that the exporter offers. This section describes features that would be of interest to power users.

12.1 Configuring a document converter

The ODT exporter can work with popular converters with little or no extra configuration from your side. See Chapter 4 [Extending ODT export], page 5. If you are using a converter that is not supported by default or if you would like to tweak the default converter settings, proceed as below.

1. Register the converter

   Name your converter and add it to the list of known converters by customizing the variable '`org-odt-convert-processes`'. Also specify how the converter can be invoked via command-line to effect the conversion.
2. Configure its capabilities

   Specify the set of formats the converter can handle by customizing the variable '`org-odt-convert-capabilities`'. Use the default value for this variable as a guide for configuring your converter. As suggested by the default setting, you can specify the full set of formats supported by the converter and not limit yourself to specifying formats that are related to just the OpenDocument Text format.
3. Choose the converter

   Select the newly added converter as the preferred one by customizing the variable '`org-odt-convert-process`'.

12.2 Working with OpenDocument style files

This section explores the internals of the ODT exporter and the means by which it produces styled documents. Read this section if you are interested in exploring the automatic and custom OpenDocument styles used by the exporter.

12.2.1 Factory styles

The ODT exporter relies on two files for generating its output. These files are bundled with the distribution under the directory pointed to by the variable '`org-odt-styles-dir`'. The two files are:

'`OrgOdtStyles.xml`'

  This file contributes to the '`styles.xml`' file of the final '`ODT`' document. This file gets modified for the following purposes:

  1. To control outline numbering based on user settings.
  2. To add styles generated by '`htmlfontify.el`' for fontification of code blocks.

'`OrgOdtContentTemplate.xml`'

  This file contributes to the '`content.xml`' file of the final '`ODT`' document. The contents of the Org outline are inserted between the '`<office:text>`' . . . '`</office:text>`' elements of this file.

Apart from serving as a template file for the final ‘`content.xml`’, the file serves the following purposes:

1. It contains automatic styles for formatting of tables which are referenced by the exporter.
2. It contains ‘`<text:sequence-decl>`’ ... ‘`</text:sequence-decl>`’ elements that control how various entities - tables, images, equations etc - are numbered.

12.2.2 Overriding factory styles

The following two variables control the location from which the ODT exporter picks up the custom styles and content template files. You can customize these variables to override the factory styles used by the exporter.

‘`org-odt-styles-file`’

Use this variable to specify the ‘`styles.xml`’ that will be used in the final output. You can specify one of the following values:

1. A ‘`styles.xml`’ file

   Use this file instead of the default ‘`styles.xml`’
2. A ‘`.odt`’ or ‘`.ott`’ file

   Use the ‘`styles.xml`’ contained in the specified OpenDocument Text or Template file
3. A ‘`.odt`’ or ‘`.ott`’ file and a subset of files contained within them

   Use the ‘`styles.xml`’ contained in the specified OpenDocument Text or Template file. Additionally extract the specified member files and embed those within the final ‘`ODT`’ document.

   Use this option if the ‘`styles.xml`’ file references additional files like header and footer images.
4. ‘`nil`’

   Use the default ‘`styles.xml`’

‘`org-odt-content-template-file`’

Use this variable to specify the blank ‘`content.xml`’ that will be used in the final output.

12.3 Creating one-off styles

There are times when you would want one-off formatting in the exported document. You can achieve this by embedding raw OpenDocument XML in the Org file. The use of this feature is better illustrated with couple of examples.

1. Embedding ODT tags as part of regular text

   You can include simple OpenDocument tags by prefixing them with ‘`@`’. For example, to highlight a region of text do the following:

   ```
   @<text:span text:style-name="Highlight">This is a
   highlighted text@</text:span>.  But this is a
   regular text.
   ```

Hint: To see the above example in action, edit your '`styles.xml`' (see Section 12.2.1 [Factory styles], page 15) and add a custom '`Highlight`' style as shown below.

```
<style:style style:name="Highlight" style:family="text">
  <style:text-properties fo:background-color="#ff0000"/>
</style:style>
```

2. Embedding a one-line OpenDocument XML

   You can add a simple OpenDocument one-liner using the '`#+ODT:`' directive. For example, to force a page break do the following:

   ```
   #+ODT: <text:p text:style-name="PageBreak"/>
   ```

   Hint: To see the above example in action, edit your '`styles.xml`' (see Section 12.2.1 [Factory styles], page 15) and add a custom '`PageBreak`' style as shown below.

   ```
   <style:style style:name="PageBreak" style:family="paragraph"
            style:parent-style-name="Text_20_body">
     <style:paragraph-properties fo:break-before="page"/>
   </style:style>
   ```

3. Embedding a block of OpenDocument XML

   You can add a large block of OpenDocument XML using the '`#+BEGIN_ODT`' ... '`#+END_ODT`' construct.

   For example, to create a one-off paragraph that uses bold text, do the following:

   ```
   #++BEGIN_EXPORT ODT
   <text:p text:style-name="Text_20_body_20_bold">
   This paragraph is specially formatted and uses bold text.
   </text:p>
   #++END_EXPORT ODT
   ```

12.4 Customizing tables in ODT export

You can override the default formatting of the table by specifying a custom table style with the '`#+ATTR_ODT`' line. For a discussion on default formatting of tables see Chapter 7 [Tables in ODT export], page 8.

This feature closely mimics the way table templates are defined in the OpenDocument-v1.2 specification.[1]

12.4.1 Custom table styles - an illustration

To have a quick preview of this feature, install the below setting and export the table that follows.

```
(setq org-odt-table-styles
      (append org-odt-table-styles
              '(("TableWithHeaderRowAndColumn" "Custom"
                 ((use-first-row-styles . t)
                  (use-first-column-styles . t)))
                ("TableWithFirstRowandLastRow" "Custom"
                 ((use-first-row-styles . t)
```

[1] OpenDocument-v1.2 Specification

```
                          (use-last-row-styles . t))))))
     #+ATTR_ODT: :style "TableWithHeaderRowAndColumn"
     | Name  | Phone | Age |
     | Peter |  1234 |  17 |
     | Anna  |  4321 |  25 |
```

In the above example, you used a template named '`Custom`' and installed two table styles with the names '`TableWithHeaderRowAndColumn`' and '`TableWithFirstRowandLastRow`'. (**Important:** The OpenDocument styles needed for producing the above template have been pre-defined for you. These styles are available under the section marked 'Custom Table Template' in OrgOdtContentTemplate.xml (see Section 12.2.1 [Factory styles], page 15). If you need additional templates you have to define these styles yourselves.

12.4.2 Custom table styles - the nitty-gritty

To use this feature proceed as follows:

1. Create a table template[2]

   A table template is nothing but a set of '`table-cell`' and '`paragraph`' styles for each of the following table cell categories:

   - Body
   - First column
   - Last column
   - First row
   - Last row
   - Even row
   - Odd row
   - Even column
   - Odd Column

   The names for the above styles must be chosen based on the name of the table template using a well-defined convention.

   The naming convention is better illustrated with an example. For a table template with the name '`Custom`', the needed style names are listed in the following table.

Table cell type	**'`table-cell`' style**	**'`paragraph`' style**
Body	'`CustomTableCell`'	'`CustomTableParagraph`'
First column	'`CustomFirstColumnTableCell`'	'`CustomFirstColumnTableParagraph`'
Last column	'`CustomLastColumnTableCell`'	'`CustomLastColumnTableParagraph`'
First row	'`CustomFirstRowTableCell`'	'`CustomFirstRowTableParagraph`'
Last row	'`CustomLastRowTableCell`'	'`CustomLastRowTableParagraph`'
Even row	'`CustomEvenRowTableCell`'	'`CustomEvenRowTableParagraph`'
Odd row	'`CustomOddRowTableCell`'	'`CustomOddRowTableParagraph`'
Even column	'`CustomEvenColumnTableCell`'	'`CustomEvenColumnTableParagraph`'

[2] See the '`<table:table-template>`' element of the OpenDocument-v1.2 specification

Odd column	'CustomOddColumnTableCell'	'CustomOddColumnTableParagraph'

To create a table template with the name 'Custom', define the above styles in the '<office:automatic-styles>' ... '</office:automatic-styles>' element of the content template file (see Section 12.2.1 [Factory styles], page 15).

2. Define a table style[3]

   To define a table style, create an entry for the style in the variable 'org-odt-table-styles' and specify the following:

   - the name of the table template created in step (1)
   - the set of cell styles in that template that are to be activated

   For example, the entry below defines two different table styles 'TableWithHeaderRowAndColumn' and 'TableWithFirstRowandLastRow' based on the same template 'Custom'. The styles achieve their intended effect by selectively activating the individual cell styles in that template.

```
(setq org-odt-table-styles
      (append org-odt-table-styles
              '(("TableWithHeaderRowAndColumn" "Custom"
                 ((use-first-row-styles . t)
                  (use-first-column-styles . t)))
                ("TableWithFirstRowandLastRow" "Custom"
                 ((use-first-row-styles . t)
                  (use-last-row-styles . t))))))
```

3. Associate a table with the table style

   To do this, specify the table style created in step (2) as part of the 'ATTR_ODT' line as shown below.

```
#+ATTR_ODT: :style "TableWithHeaderRowAndColumn"
| Name  | Phone | Age |
| Peter |  1234 |  17 |
| Anna  |  4321 |  25 |
```

12.5 Validating OpenDocument XML

Occasionally, you will discover that the document created by the ODT exporter cannot be opened by your favorite application. One of the common reasons for this is that the '.odt' file is corrupt. In such cases, you may want to validate the document against the OpenDocument RELAX NG Compact Syntax (RNC) schema.

For de-compressing the '.odt' file[4]: See Section "File Archives" in emacs. For general help with validation (and schema-sensitive editing) of XML files: See Section "Introduction" in nxml-mode.

If you have ready access to OpenDocument '.rnc' files and the needed schema-locating rules in a single folder, you can customize the variable 'org-odt-schema-dir'

[3] See the attributes 'table:template-name', 'table:use-first-row-styles', 'table:use-last-row-styles', 'table:use-first-column-styles', 'table:use-last-column-styles', 'table:use-banding-rows-styles', and 'table:use-banding-column-styles' of the '<table:table>' element in the OpenDocument-v1.2 specification

[4] '.odt' files are nothing but 'zip' archives

to point to that directory. The ODT exporter will take care of updating the '`rng-schema-locating-files`' for you.

13 Main Index

14 Key Index

15 Command and Function Index

16 Variable Index

17 What is New