How to write a manual for a GNOME application with DocBook

COLLABORATORS

	TITLE :	*REFERENCE* :
	How to write a manual for a GNOME application with DocBook	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Manuel Rego Casasnovas	December 31, 2008	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Abstract

This article is a guide about how to write help manuals for GNOME applications using DocBook as standard tool. This manual explains the right way to add a tutorial on a GNOME project, as well as the process to translate it. The integration process to add a new option in application menu that shows the help manual, is also explained in this article.

1 Introduction

A manual explains how to use a particular program. It is a very good method of transferring knowledge. Moreover, a tutorial is usually more interactive and specific than a book, and could be used as a part of learning. Besides, a manual is usually updated to the last versions of an application, while books are quickly old-fashioned.

So if you are a user of a GNOME application that do not have a manual right know, you can easily write it and integrate in the application following this guide.

In order to write the manual you will need to use DocBook, a markup language to write technical documentation. If you are a newbie on this kind of tools you could need more time to write the tutorial, anyway with this how-to you will be able to do it for yourself.

The last step will be integrate the written manual, as the application help, adding an option into program menu to show that help. For this process is needed to do some changes on the source code, but the whole process is documented with this guide.

Finally, this article gives an overview about how to translate a manual of a GNOME application written with DocBook, this translation it will be also integrated with the application.

2 Writing the manual

First of all, you should know that exists a GNOME Documentation Project managed by a team and with the support of the community. In its web page you can find a lot of links and information related with how to write documentation for any GNOME project, from developer documents to user manuals.

As any free software community you could be part of that team and contribute with your time documenting some GNOME applications, they will be glad about your collaboration.

2.1 Style guide

Before start to write the manual you should read the style guidelines, in order to keep a common style appearance in all the documents related with the GNOME project. You can find this guide and many others at GNOME Documentation Library web page.

Anyway, just in case you can not read it, the next list tries to summarize the main advices extracted from the guide:

- A good technical documentation has to be comprehensive, conformant, clear, consistent and concise.
- There are four golden rules:
  - Limit each sentence to less than 25 words.
  - Limit each paragraph to one topic.
  - Provide instructions rather than descriptions.
  - Write in a neutral tone.
- Use screenshots only when they are really needed.
- Review the list of agreed terminology for use in GNOME documentation.

If you like that your work, the application help manual, to be included on future releases of that program, you should take care to follow the standards defined in the style guide. Otherwise, the application developers could not accept your collaboration, so this guide is very important and is highly recommended reading it.

2.2 DocBook

In order to write a standard manual for a GNOME application you have to use DocBook. DocBook is a language for writing structured documents using XML, currently maintained by OASIS (Organization for the Advancement of Structured Information Standards) .

Note
If you already know this system you can skip this section, anyway it includes only a few references about how to write documentation with DocBook.

Documents in DocBook are just simple XML files with some special tags, these tags define the structure of the document. The writer do not have to take care about the final appearance of the document, and he can be centered only on the content. This is the main advantage of this kind of systems to write documentation, like DocBook or LaTeX. Moreover the final result could be published in a lot of different formats (HTML, PDF, RTF, etc.)

You do not need any special editor to write a document in DocBook, just use your favorite text or XML editor. However, in order to generate the final document you need some tools that will be explained at Section 3.

There are a lot of references about DocBook on the web, but the most important are at the official website.

In the next list you can see the most common tags that you need to write an application manual for a GNOME application, from more to less used:

para Defines a paragraph, almost every text in DocBook is enclosed in a paragraph.

title Title text of a section or other block elements.

sect1, **sect2**, **sect3** These tags are used to define the document sections. The top level sections are tagged with `sect1`, subsections with `sect2` and subsubsections with `sect3`.

itemizedlist, **orderedlist**, **variablelist** Define different kind of lists: a simple list, an ordered list or a definition list.

xref Internal cross reference in the document.

ulink External reference to any link outside the document.

caution, **important**, **note**, **tip**, **warning** Some types of messages set off from the text.

application To mark a program name.

command, **option**, **parameter**, **replaceable** These tags define the name of an executable program with the different possible options and arguments.

screenshot, **mediaobject**, **imageobject**, **textobject** These set of tags are used to add screenshots in the tutorial.

menuchoice Define the selections needed to be done from a menu.

guibutton, **guiicon**, **guilabel**, **guimenu**, **guimenuitem**, **guisubmenu** Tags to reference different things related with the Graphical User Interface (GUI) of the application.

keycap, **keysym**, **keycombo** Different tags to talk about keys on a keyboard.

Finally, DocBook allows to define entities in order to assign a name to some chunk of data. For example, these entities give the possibility to divide a document in different files.

You can define entities with something like the following line:

```
<!ENTITY app "<application>MY-GNOME-APPLICATION</application>">
```

GNOME manuals always define some entities that you should use in your document, for example you could refer to the application with `&app;`.

Other entities are already defined to use some special characters, for example `<` for < or `&` for &.

2.3 Templates

The GDP (GNOME Documentation Project) provides some DocBook templates to write GNOME tutorials, in order to avoid start a manual from an empty document. You can download these templates at GDP Document Templates or GNOME Doc Utils repository.

There are three available templates:

- Applet Template: For GNOME applets (smaller applications).
- Application Template: For GNOME applications.
- Legal Info Template: A section with legal notes that is included by both templates above.

The next list enumerate some things that all the manuals should have, all of them are included in the standard templates:

- Sections identifiers: All the sections of the document should have an unique identifier to allow cross-reference.
- Authors names: The manual should contain information about all the authors that have worked on the document.
- Copyright information: Documents should have a copyright notice and a license allowing free redistribution.
- Revision history: Every manual should keep the history of all versions.
- Application version: The tutorial should identify the application version for which the document is written.
- Publisher name: All GNOME documents must include as publisher the GNOME Documentation Project.
- Software license: The manual should include information about the software license.
- Bug reporting: All documents should give a way for reporting bugs or comments about the documentation.

To sum up, the best way to write a good manual for a GNOME application is to use a template, as base document, and make all the needed modifications on the template.

3 Compiling the manual

At this moment, you could have already written an application manual following a template provided by GDP. However, you have not seen anything related with the final result, you only have seen a XML file with tags and texts. You will need to install some tools in order to can see the output of your tutorial, checking that everything is right.

3.1 Install packages

The required packages for Debian users are:

`docbook` Standard SGML representation system for technical documents.

`docbook-xsl` Stylesheets for processing DocBook XML files to various output formats.

`libxslt1.1` XSLT processing library - runtime library.

`yelp` Help browser for GNOME 2.

Optionally, you could install the next packages to generate the output in different formats:

`xmlto` XML-to-any converter.

`dblatex` Produces DVI, PostScript, PDF documents from DocBook sources.

Warning
For those who are not Debian users you can find more information at GNOME Handbook of Writing Software Documentation.

3.2 Checking the result

Firstly, you should check that your document is valid. You can use xmllint to do it:

xmllint --noout --noent --valid filename.xml

If it does not display any error then the document is valid. Otherwise you should modify your XML file to follow the DocBook reference.

Once you are sure that you have a proper document, you could see the final output using yelp:

yelp ghelp:/path/to/filename.xml

This program will open a GNOME help browser with your manual, you can navigate and see the different sections, checking that everything is right.

This will be the same result as when the manual will be integrated as application help.

3.3 Convert into different formats

On the other hand, you can generate different output formats from your document with different applications.

In order to get an HTML you could use xmlto:

xmlto html filename.xml or **xmlto html-nochunks filename.xml**

If you like to generate a PDF it is easier to use dblatex.

dblatex filename.xml

Moreover you could define your own stylesheet to generate a customized output.

Tip
Check the man pages of the different commands from this section, in order to know the possible options and arguments of each one.

4 Integrate as application help

This section explains how to integrate the written document as help of any GNOME application. You will need to place the DocBook file in an specific folder and make several changes on source code (mainly C and Autotools files).

Warning
If you are not a developer and you do not have any knowledge about programming, you should skip this section and send your manual directly to the developers of the program, usually via the mailing list or bug tracker of the project. They will take care of integrating your tutorial in their application.

4.1 Preparing the source code

First of all, you need to be sure is that `gnome-common` package is installed in your computer. After that, you could start the process defined below.

Usually DocBook manuals are placed inside a folder called `help`, moreover the name of the tutorial file is like the application name. Actually, the XML document in English should be inside the directory `help/C/`. As in the case of `po` directory, inside the folder `help` there is a `ChangeLog` file to keep a track about the changes related with the tutorial.

- Inside the `help` folder you need a file called `Makefile.am` with the next content:

```
include $(top_srcdir)/gnome-doc-utils.make
dist-hook: doc-dist-hook

DOC_MODULE = <document-name>
DOC_ENTITIES = legal.xml
DOC_INCLUDES =
DOC_FIGURES =
DOC_LINGUAS =
```

  Where:

  DOC_MODULE The name of the document, the file name without extension.

  DOC_ENTITIES List of files included in the manual as entities. `legal.xml` is always included.

  DOC_INCLUDES List of files included in the tutorial.

  DOC_FIGURES List of figure paths of the images included in the document. Figures are usually inside a folder called `help/C/figures/`.

  DOC_LINGUAS List of language codes for which the document is translated.

- Furthermore, a template OMF file is required inside the directory `help` and it must be called `<document-name>.omf.in`. The content of this template will be like:

```
<?xml version="1.0" standalone="no"?>
<omf>
  <resource>
    <subject category="<categories>"/>
    <type>manual</type>
    <relation seriesid="<generated-series-id>"/>
    <rights type="GNU FDL" license.version="1.1" holder="<holder-name>"/>
  </resource>
</omf>
```

  Where:

  <categories> List of categories separated by "`|`". These categories have to belong to a concrete list.

  <generated-series-id> The first time you can generate this number with the command **uuidgen**. You should not change this identifier anymore.

- The next step will be modify the `configure.ac` file, just to add the next lines:

```
GNOME_DOC_INIT

...

AC_CONFIG_FILES([
...
help/Makefile
...
])
```

- It is also needed to modify the top-level `Makefile.am` in order to:
  1. Add `gnome-doc-utils.make` to `EXTRA_DIST`.
  2. Add `gnome-doc-utils.make` to `DISTCLEANFILES`.
  3. Add `--disable-scrollkeeper` to `DISTCHECK_CONFIGURE_FLAGS`.
- Add a dependency on `gnome-doc-utils` module.

Once you have done all the steps defined above, the DocBook manual will be part of the program. It does not appear in application menu, but it will be installed on the system when the program is installed. You can check it opening the GNOME Help and looking for the name of your application.

4.2 Adding menu option

If you like that your manual appears in the application menu you should modify the source code of that application, so you will need to respect the guidelines to write source code in that project.

Usually the application uses `GtkActionEntry` to define the menu, so you will need to add the next lines in order to have a new entry called "Help", with an option "Contents" that will show your tutorial:

```
static const GtkActionEntry entries[] = {
  ...
  { "Help", NULL, N_("_Help") },
  ...
  { "HelpContents", GTK_STOCK_HELP, N_("_Contents"), "F1", NULL,
  G_CALLBACK (<application-function-help>) },
  ...
}
```

Here you can see how to define a callback to execute the function `<application-function-help>`, so here you need to use a function where you will open the help manual.

You can see some code snippets about how to do this in a lot of GNOME projects, for example:

Eye of GNOME At line 3345 of file `eog-window.c` (version 2.25.3).

Evince At line 4486 of file `ev-window.c` (version 2.25.2).

4.3 Opening help document

When the new option appears in the menu, you should implement some functionality inside the callback defined. The callback function has to open yelp and show the application help manual.

In order to open the tutorial you should use the function gtk_show_uri, passing as second argument the application name preceded by the prefix `ghelp:`:

```
gtk_show_uri (NULL,
  "ghelp:<application-name>",
  gtk_get_current_event_time (),
  &error);
```

As in the previous point you can see some code snippets in a lot of GNOME projects, for example:

Eye of GNOME At line 47 of file `eog-util.c` (version 2.25.3).

Evince At line 3412 of file `ev-window.c` (version 2.25.2).

5 Translation

Translations of GNOME manuals are made in the same way than translation of applications, using `.po` files based on gettext system.

In order to translate a manual to a new language you have to follow the next steps:

1. Create a folder inside `help` directory called as the language code.

   mkdir help/<langcode>

2. Create a `figures` folder (this is only needed if it exists a `figures` folder inside `help/C`).

   mkdir help/<langcode>/figures

3. Add the new language code to `DOC_LINGUAS` in `Makefile.am`.

   ```
   DOC_LINGUAS = es, fr, it, <langcode>
   ```

4. Use `make` inside the folder `help`, this will generate a file `<langcode>/<langcode>.po`.

   make

5. Now, you should translate the new `.po` file using some translation tool, or manually editing the file. However, it is strongly recommended to use some translation tool, because of this kind of applications will help you a lot during the translation process.

6. Once the translation is done, the last step should be prepare a patch to be sent to the translation team of this language, or directly to the developers, depending on the situation. You should not forget to update the `ChangeLog` file inside the `help` folder.

This way to do the translations, using gettext as translation system, has a big advantage, because of people used to translate applications can translate tutorials without any extra effort, just using the same tools and translating a new `.po` file. The translation process of a manual is transparent for them.

6 References

[1] *GNOME Documentation Project*, The GNOME Project.

[2] *GNOME Documentation Library*, The GNOME Project.

[3] Shaun McCance, *GNOME Documentation Build Utilities*, GNOME Documentation Project, 2004.

[4] John Fleck and Alexander Kirillov, *GNOME Documentation Style Guide*, GNOME Documentation Project, 2005.

[5] Shaun McCance, *GNOME Documentation XSLT Manual*, GNOME Documentation Project, 2004.

[6] David Mason, Daniel Mueth, Alexander Kirillov, Eric Baudais, Eugene O'Connor, and John Fleck, *GNOME Handbook of Writing Software Documentation*, GNOME Documentation Project, 2003.

[7] *GDP Document Templates*, The GNOME Project.

[8] *DocBook.org*

[9] Norman Walsh and Leonard Muellner, *DocBook*, The Definitive Guide, O'Reilly & Associates, Inc, 2006.