ODF in action

Using OpenDocument format text in OpenOffice, KOffice and AbiWord

Tom Kuipers Alan Berg

Numerous office suites and word processors support the *OpenDocument format* (ODF). ODF is an open standard for saving and exchanging office documents. The standard has been developed to provide an open alternative to proprietary, for example Microsoft Office, document formats.

In this article we will take a visual tour through different free software editors and examine their core abilities in creating, editing and saving an ODF document. The exercise may on the surface seem trivial, but in reality it has important implications. We demonstrate on a small scale that which may happen in real world situations when a document is shared by different users in multiple environments: home, work, on the road.

Note: If you have a reasonable internet connection then the authors recommend downloading the live CD's mentioned in the resource section. This will allow you to preview all the software mentioned and experiment with the Gnome and KDE desktop environments.

The Ubuntu live OpenOffice combination

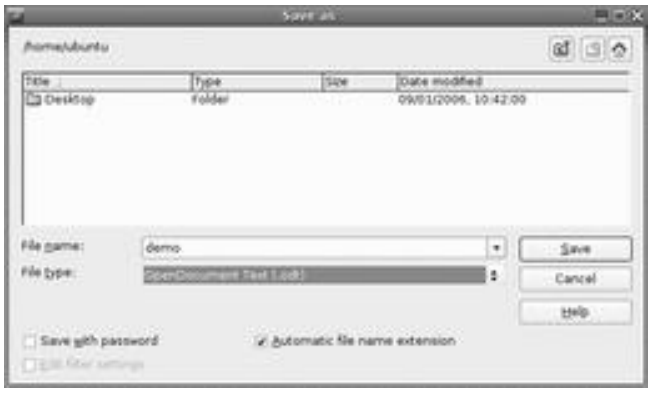
To start the tour let us focus on OpenOffice, a well respected professional office suite. Along with text documents the application also has the ability to save spreadsheets, charts and presentations in ODF. In this article, for the sake of simplicity, we will concentrate on manipulating text documents. Please note, all documents mentioned within this article are contained in a tar file referenced in the resources section.

The recipe: After booting from the *Ubuntu* live CD (version 5.10), we started up OpenOffice Writer from within the *Gnome* desktop environment and created a new word processor document. The document contained multiple paragraphs, different headings, various fonts, a table and a picture. We considered the document stereotypical of its type. After finishing writing the document, we saved via selecting `Save` from the `File` menu. By default OpenOffice.org loads and saves files in the OpenDocument file format. Notice that the file type has been set to `OpenDocument Text (.odt)` (shown in figure 1). Further, please note that OpenDocument Spreadsheets are saved with extension `.ods`, and OpenDocument Presentations as `.odp` files. We saved the edited document as `demo.odt`. We plan later to modify the document from within other applications. Due to the fact that we are running from a Live CD we cannot guarantee write access to the hard drive. Therefore we recommend storing file on a USB pen drive. However, on a positive note, the *Knoppix* live CD (version 4.02) had no troubles guessing type and mounting the windows partitions on my home computer.

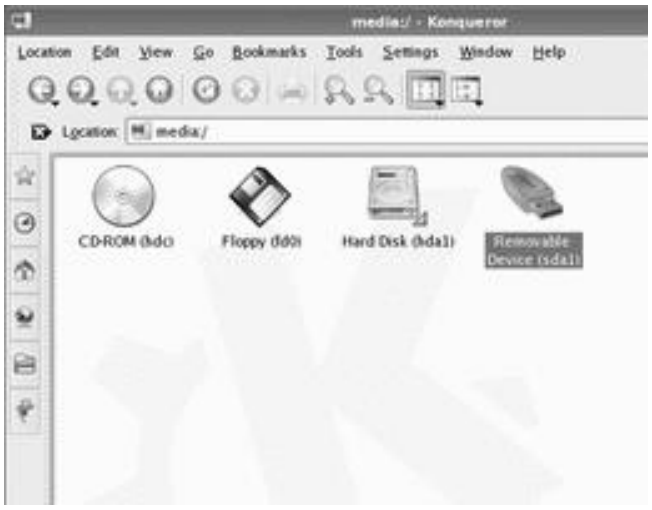
Slax 5.0.6 with KOffice

Slax, another excellent live CD distribution, is derived from **Slackware**. We deployed the Standard Edition which employs KDE 3.4 (the K Desktop Environment). The word processor we had available and tested was KWord version 1.4 part of the well regarded **KOffice suite**.

Figure 1: Saving a document in OpenDocument Text format with OpenOffice



Mounting a USB stick



The recipe: Let us first test if we can now open the previously created document `demo.odt` in this new environment. Go to the `File` menu and select `Open`. The dialog window displays a list of supported filters (file types), but does not display the file type extensions. To open our `demo.odt` file we point the operating system to the USB stick. First, double click on the `System` desktop icon, then go to `Storage Media`. The USB stick gets mounted when you click on the `Removable Device` icon.

In KWord `.odt` files are referred to as *OASIS* OpenDocument Text.

An aside: OASIS is the originating standards body. OASIS is a not-for-profit, global consortium that drives the development, convergence and adoption of e-business standards. OASIS has developed many interesting standards spread across a wide spectrum of problem domains. At the University of Amsterdam we provision our premier E-learning environment via XML based communications adhering to OASIS enterprise standards. This group really understands how to write high quality standards for mission critical systems.

When you open the document you will notice that the document looks different from how it was saved in OpenOffice. The picture which was centered is now aligned to the left. Additionally there is extra white space below the image. All these minor changes accumulate into generating a two page document instead of the original single page.

Although the document can be read without problems and all content is available, this is not what you would expect from an open standard for exchanging office files. However, that said, in the short term the software market is rapidly evolving. It would not surprise the authors if the compatibility issue is vastly improved or removed within months.

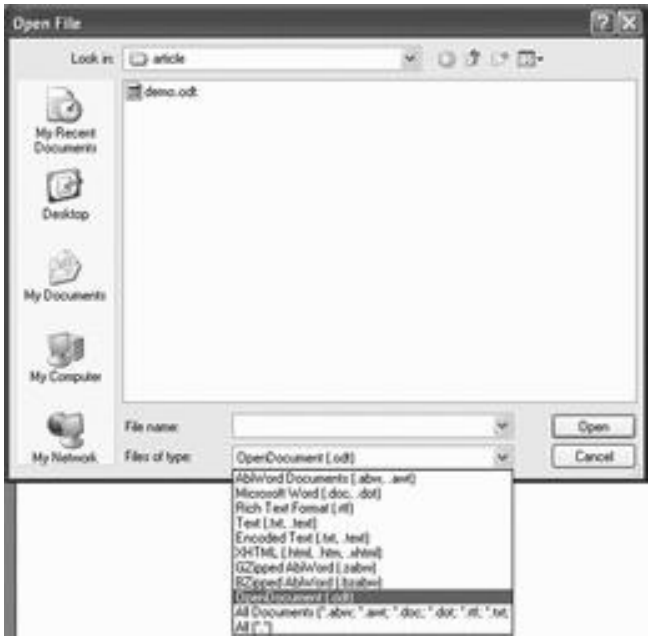
To continue the roundtrip it is now necessary that we make changes to the demo document. We are now highly motivated to see how other popular word processors handle the document. We now insert an extra paragraph above the picture and centre the picture. The modified document is then saved, by clicking `Save` from the `File` menu or pressing `Ctrl+S`. Now we do not have to specify the file format, because the word processor automatically saves the file to `.odt` format.

An important benefit of standards is the enabling of easy and transparent exchange of data

Windows XP with Abiword

An important benefit of standards is the enabling of easy and transparent exchange of data. You are not tied to a specific application or vendor. This interoperability does not just exist on the application level but also on platform level as well. To demonstrate this we open the document created and edited in the previous steps, but now on a Windows platform. For this purpose we use the standalone word processor *AbiWord* running under Windows XP.

Opening an OpenDocument Text with AbiWord on Windows XP



AbiWord has ambitions to become a fully cross-platform word processor. Currently AbiWord runs on most UNIX systems, Windows 95 and later, and MacOS X.

When exchanging documents, data loss is obviously not acceptable. We consider losing the layout as a form of data loss as well, and therefore the loss of layout is a serious matter

Recently (January 2006) the released version 2.4.2 has native support for opening OpenDocument (`.odt`) files. However, earlier versions had read support enabled by installing a specific OpenDocument import plugin.

When we visit the `Open` item within the `File` menu a dialog window appears. The dialog shows a list of file types containing the option *OpenDocument (.odt)*. Without problems we can point to the USB stick and open `demo.odt`.

Remarkably, the document doesn't look the same as it did when it was last saved with KWord (KOffice). So, just like when we moved from OpenOffice to KWord, our sample document suffers from non-trivial layout differences. Again, while we haven't lost content, parts of the formatting details have been lost. While the image is still centered as it should be, the paragraph wraps around the image. If rendered consistently, then the paragraph should begin underneath the picture.

When exchanging documents, data loss is obviously not acceptable. We consider losing the layout as a form of data loss as well, and therefore the loss of layout is a serious matter. Some may say that included images are not that important as long as they are shown. They may argue that the primary focus of a word processor is the processing of text and that dealing with pictures is of secondary concern. But we think format is an integral part of office documents that combine text, graphics, charts etc. The formatting is the key attribute of a word processor over a text editor.

Within AbiWord it is not possible to save a document in ODF. There is no plugin available that offers this functionality. The developers prefer to use the RTF (Rich Text Format) document file format as the best method to share documents with other word processing software. As becomes clear from a mailing list discussion (`http://www.abisource.com/mailinglists/abiword-dev/2003/Apr/0167.html`), the reason for this lies in technical issues in the underlying data structure. The internal data representation of a document in AbiWord matches RTF more closely. It is the author's personal views that market momentum for ODF will speak out and motivate developers into complying with a standard way of doing business.

Documenting an OpenDocument document

OpenDocument files are stored in JAR (Java Archive) format. JAR files which are really just compressed ZIP files. An ODF document is actually a collection of XML files and other data files such as images. To show the contents of our sample `demo.odt` file using a terminal, type `unzip -v demo.odt` and view the expanded content.

Market survey

Standards are only truly useful once applied in a wide context. This means there is a need for applications that fully adopt the ODF. On a positive note, current support is not

only limited to well known word processors. Numerous vendors offer standards based compatibility, including large companies such as IBM and its Workspace Collaboration Services and the commercial office suite StarOffice from Sun. Corel's WordPerfect lacks support at the present time but is expected to adopt ODF in the near future. Note: Corel was among the original members of the OASIS committee that developed the specification. Despite all of this we are not there yet. As we have seen from the experiments, consistent rendering of detailed formatting between word processors is a significant issue. This one issue will act as a drag on market penetration unless vigorously dealt with. Currently, most of the standalone word processors only offer the ability to import `.odt` formatted documents. Momentum is building based on user demand. So, you'd expect that when this official standard gains in popularity, software manufacturers will try harder to create consistent exporting features. This will in turn boost the use of open standards file formats.

Adoption not only depends on support from word processors, but also from a range of other applications. For instance the fact that the Google Desktop search application understands ODF is promising. *The market is converging.*

Conclusion

The importance of the OpenDocument Text format is best reflected in the differences between open and proprietary formats. Take for example the Microsoft `.doc` format. The proprietary format is not fully documented and other vendors are hindered in this subtle way from implementing the `.doc` format functionality in their products. This implies that users risk "vendor lock in". In this situation, customers become completely dependent upon a single software vendor. And worse still, they are dependent on that vendor for future support of the numerous versions of the format. In the proprietary world you are almost forced to upgrade every so often to newer versions. Users that store their data in an open format avoid this situation leaving them free to switch to other software while still being able to access their original data. Organizations and individuals are recognising that office documents should be accessible now and in the future. This can only be realistically accomplished by storing data in a vendor-neutral, open standard format. In the middle term, the real investment is not in the software to create and use office documents, but is the content of documents that are created.

As an open standard, ODF has the advantages of: freedom of choice between software vendors, durability, and exchangeability. The disadvantage in the short term is consistent rendering and initial issues in newly written software. However, ensuring middle term success will require sustained momentum in the market place and this in turn demands compatibility. We believe this is now happening and, thankfully, very rapidly.

Resources

- Ubuntu Linux (`http://www.ubuntulinux.org`)
- OpenOffice (`http://www.openoffice.org`)
- Slax (`http://slax.linux-live.org`)
- KOffice (`http://www.koffice.org`)
- AbiWord (`http://www.abisource.com`)

Copyright information

About the authors

Tom Kuipers, MA., is a developer at the University of Amsterdam. His fields of expertise include ColdFusion, Java, JSF, XML, XSLT, CMS, streaming media and electronic learning environments. He likes to play with his casemodded Linux box.

Alan Berg, BSc., MSc., PGCE, has been a lead developer at the Central Computer Services at the University of Amsterdam for the past seven years. You can contact him at reply.to.berg@chello.nl (`reply.to.berg@chello.nl`)