



The Perl OpenDocument Connector

Jean-Marie Gouarné
jmgdoc@cpan.org

The `OpenOffice::OODoc` module allows Perl programs to create, retrieve, update or delete any data in any OpenDocument-compliant file. It has a growing set of read/write accessors to flat text containers, such as paragraphs; structured or hierarchical containers, such as tables and lists; as well as variable fields, style descriptors, graphics, and global meta-data. These data may appear in text, spreadsheets, presentations, or drawings. I created `OpenOffice::OODoc` in 2003, two years before the official release of the OASIS OpenDocument format (ODF), so it was initially compliant with the `OpenOffice.org 1.0` file format. It became a CPAN distribution in the beginning of 2004, and switched to OpenDocument in 2005, although I still support the older `OpenOffice.org` format.

Forget the good old, office automation logic! This is an interface to the OpenDocument files which deals with the XML representation of a document, not with the features of an interactive editing software. To an end user, a three click operation looks to be straightforward. To the programmer, it may become a nightmare. On the other hand, `OODoc` provides some automatic processing features which don't make sense in an interactive context.

`OpenOffice::OODoc` shows up in an eclectic set of areas, including health care, food, legal, real estate, and petroleum exploration. Some use OpenDocument as a pivot format for mission-critical automatic document processing while the end-users' desktops use commercial office software; the appropriate format conversions happen in the background. `OpenOffice::OODoc` is useful even without `OpenOffice.org` or any other OpenDocument-compliant software. Beware though: this isn't a format conversion tool. It's designed only to get and set data in ODF files.

Opening and saving documents

A document is a set of XML members stored in a compressed, OpenDocument-compliant file. Programs process uncompressed XML documents,

whether previously loaded in memory or available on flat files, but I consider the "regular" ODF storage only for this article.

To processing a document, I need to get a "connector" (as called a "handler") for each workspace I want to process. The easiest way is through the `ooDocument()` constructor:

```
my $doc = ooDocument(
    file    => $filename,
    member => $workspace
);
```

The `member` option is a workspace selector, usually `content` (the default) or `styles`. The `content` workspace hosts the whole document body. Some text elements may be in other areas, such as page headers and footers, or the named, reusable style definitions. Another frequently used member is `meta`, which contains all of the metadata, such as author and creation date, for the document.

`ooDocument()` extracts from the ODF file, parses it, and maps it to a tree data structure in memory, which I store in `$doc`. This is an `OpenOffice::OODoc::Document` object.

To create a new file, I use the `create` option, which loads a set `OpenOffice.org` or `OpenDocument XML` templates, embedded in the `OpenOffice::OODoc` package, in order to create the structure of a new document in memory. I can use one from the four classes presently supported by `OpenOffice::OODoc`: `text`, `spreadsheet`, `presentation`, or `drawing`.

```
my $doc = ooDocument(
    file          => $filename,
    member        => $workspace,
    create        => $class,
    opendocument => 'true'
);
```

The last option, `opendocument`, selects the file format; `true` or `1`, selects the OpenDocument format (ODF), and `false` or `0` a format compatible with the