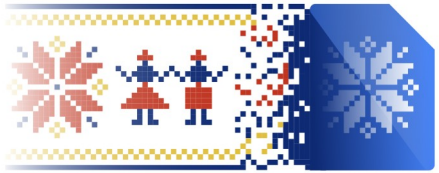
Introduction into Writer development

Miklos Vajna
Software Engineer
vmiklos@collabora.com
2023-09-23

Collabora Online



LibreOffice Conference
Bucharest 2023

LibreOffice Technology

About Miklos

- From Hungary
- More details:
  - https://www.collaboraoffice.com/about-us/
- Google Summer of Code 2010 / 2011
  - Rewrite of the Writer RTF import/export
- Then a full-time LibreOffice developer for SUSE
- Now a contractor at Collabora

Collabora Online

LibreOffice Technology

Tools helping development

- Git: log, blame, bisect
- Ctags / id-utils + http://docs.libreoffice.org
- Gdb, xray, tpconv
- Vim / emacs
- Pretty-printing:
  - SAL_DEBUG()
  - Edit zip file in-place
  - XML / RTF pretty-printer
  - Doc-dumper
- Specifications: ODF, DOCX, DOC, RTF, etc.

Where is the code?

- LibreOffice has many modules (238 ATM on master)
- Writer-related modules
  - sw (StarWriter): Writer itself
    - Document model, layout, UI, some filters
  - xmloff: (most of) ODF import/export
  - writerfilter: UNO-based DOCX/RTF import
  - oox: shared OOXML bits (between DOCX, XSLX, PPTX)
  - starmath: equation editor

LibreOffice Technology

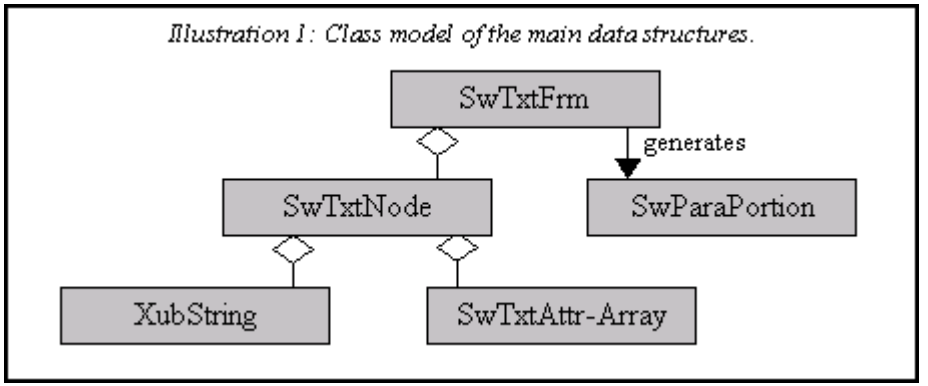
Document model

- The model from MVC
  - View is called layout; controller is called a shell
- One opened document is one SwDoc instance
  - SwDoc::GetNodes() → SwNode array (has pretty-printer in gdb)
- Inside that, building block: paragraphs
  - One paragraph: one SwNode
- Terminology:
  - Word has sections, paragraphs and runs
  - Writer has page styles, sections, paragraphs and text portions

Collabora Online

How properties are stored

- SwNode has the paragraph text as a single OUString
- Properties:
  - SfxPoolItem
  - Stored in an SfxItemSet
    - Think of it as a map<int, any>
- “int” is called a WhichId:
  - Writer specific ones are in sw/inc/hintids.hxx
- SfxPoolItem is has many subclasses, examples:
  - Bold: SvxWeightItem (Sv: StarView)
  - Paragraph adjust (centered e.g.): SvxAdjustItem



LibreOffice Technology

More on SfxItemSet

- Can contain ranges of WhichIds: _pWhichRanges
  - Array of pointers: value “n”: start of a range
  - Value “n+1”: end of a range
  - End of the list: 0
- Can have a parent: think of style inheritance
- While debugging: _nCount contains the size
- Items are pointers: _aItems
  - If a property is “set”, its pointer is non-zero

LibreOffice Technology

Character attributes

- Direct formatting is in SwTxtNode::m_pSwpHints
  - Each such formatting is a “hint”
  - Either just a character index
    - E.g. field
  - Or a start-end (e.g. bold)

Collabora Online

LibreOffice Technology

How to debug the doc. model

- Demo:
  - Gdb
  - Document model XML dump
  - Xray

Collabora Online

LibreOffice Technology

UNO API

- This is the public API, any change to it comes with some cost
  - Still, not set in stone
  - Extensions use this, UNO-supported languages (C++, Java, Python etc) can connect to a running soffice using URP
- If the document model is changed, the API has to be updated in most cases
  - We serialize everything to ODF, and that uses the UNO API as well
    - Bad: slower than necessary
    - Good: UNO API is kept up to date

UNO API (continued)

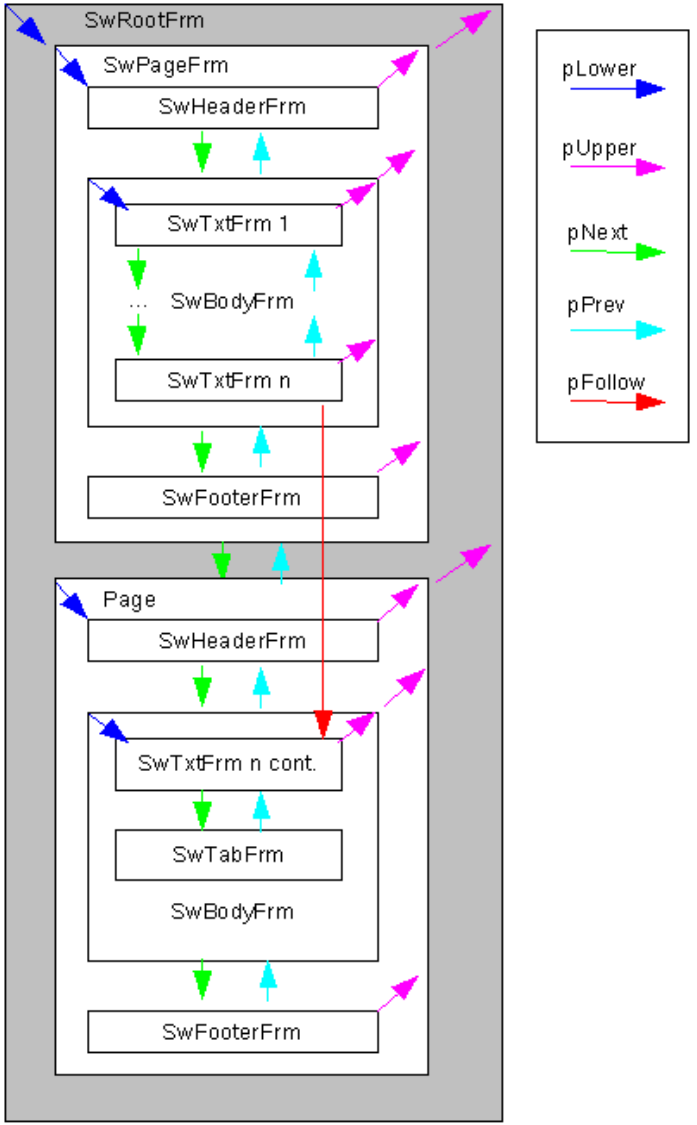
- When adding a new feature, if this is implemented, can read / write the document model
  - Other approach: implement the UI
- Properties themselves:
  - Most SfxPoolItem has two methods to load / save:
  - QueryValue() + PutValue()
- New frame, paragraph, character, list (etc.) property:
  - sw/source/core/unocore/
  - Maps between UNO's string + any key-value and WhichIds + SfxPoolItems
- Trick: InteropGrabBag

LibreOffice Technology

Layout

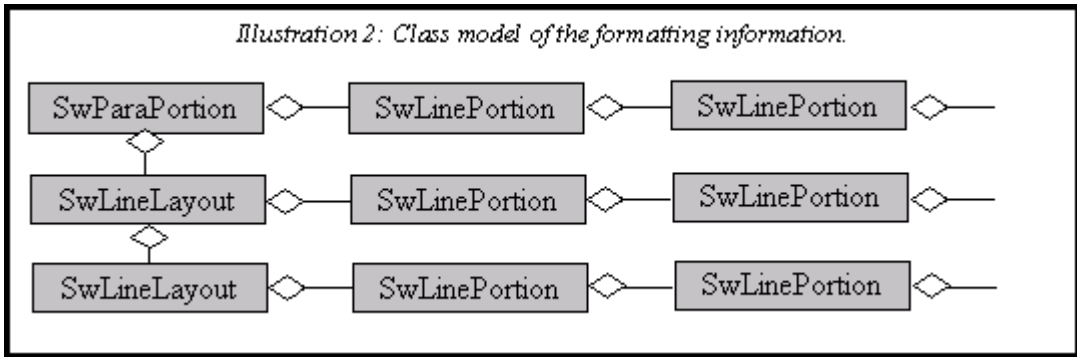
- Most complex part:
  - No easy way to test automatically
    - Think of missing fonts on test machines
  - Document model has only paragraphs, not pages
- An opened document can have multiple views
  - Try it: Window → new window
- Typically single layout: SwRootFrm (root frame)
  - Inside: pages – SwPageFrame
  - Paragraphs – SwTextFrame

Layout


SwRootFrm
SwPageFrm
SwHeaderFrm
SwTxtFrm 1
...
SwBodyFrm
SwTxtFrm n
SwFooterFrm
Page
SwHeaderFrm
SwTxtFrm n cont.
SwTabFrm
SwBodyFrm
SwFooterFrm
pLower
pUpper
pNext
pPrev
pFollow

Layout inside a paragraph

- No more frames:



LibreOffice Technology

Doc model → layout notifications

- SwModify: kind of a server, e.g. SwTextNode
- SwClient: the client, e.g. SwTextFrame
- SwModify ↔ SwClient is 1:N
- SwModify has Modify(SfxPoolItem* pOld, SfxPoolItem *pNew)
  - So layout can react without building from scratch
  - SwClient can only be registered in one SwModify
    - sw::MergedPara helps with this for redlining, which needs an N:M relation between text nodes and frames

Collabora Online

Related: TextFrames and drawings

- Writer has its own TextFrame on its UI
  - Can contain anything: tables, columns, fields, etc.
  - Does not support advanced drawing features
    - Like rounded corners
- Drawinglayer (shared) takes care of all other drawings
  - Also has a rectangle, with all features one can ever wish
    - Rounded edges, rotations, etc.
    - Except it doesn't know about Writer layout, so can't contain fields, etc.
- Problem for Word interop:
  - The UI / UNO API now can create a TextBox for a draw shape in Writer
  - This is just a draw shape + TextFrame pair in practice

Filters

- Every feature stored in the document model has to be serialized / loaded back to every file format
  - Or you loose data
  - In practice: ODF should not loose data, the rest should be good enough
- Important filters:
  - ODF (.odt)
  - OOXML (.docx)
  - WW8 (.doc)
  - RTF (.rtf)
  - Rest: HTML, plain text, etc.

ODF filter

- If you extend the document model, this has to be updated before the change hits a release
  - So users have at least one format which don't loose data for sure
- Mostly uses the UNO API:
  - Code under xmloff/
- Some Writer-specific bits are using the internal API:
  - sw/source/filter/xml/
- Is an open standard, proposals for new features can be submitted

LibreOffice Technology

OOXML: DOCX

- Import:
  - Uses the UNO API, code under writerfilter/
  - Tokenizer:
    - Shared XML parser, model.xml → tokens
  - Domain mapper:
    - Handles the incoming stream of tokens and maps them to UNO API
  - Tokenizer → dmapper traffic is XML logged:
    - SW_DEBUG_WRITERFILTER=1, then /tmp/test.docx*.xml after load
- Export:
  - Shared with RTF/WW8, uses internal API
  - sw/source/filter/ww8/docx*

LibreOffice Technology

OOXML: shared parts

- For drawing and other shared parts, writerfilter calls into oox
  - drawingML import: oox/source/drawingml/
  - drawingML export: oox/source/export/
  - Exporter also writes VML for Word 2007
  - Also: metadata parsing (author date, etc.)
- Math expressions: both import/export under starmath/
  - starmath/source/ooxml*

XFastParser

- DOCX import is a push parser
- Benefit: can implement feature incrementally
- Drawback: XML is text, would have to compare strings a lot → slow
- Solution: we know all the expected strings (namespaces, element names, attribute names, attribute values)
  - Register a string → id map before parsing
  - Exactly what XFastParser does

Collabora Online

XFastParser (continued)

- Other than being “fast”, how does it work?
- Problem: we don't want a single handler class (startElement, endElement, etc) for the whole document, it would be a God object
- Solution: XFastContextHandler interface
  - createFastChildContext() method to handle child contexts → can be a different class

DOCX import: model.xml

- DOCX tokenizer works by having all its configuration in the model.xml, then generated code does the real work
- Input: XML stream + mapping definitions (model.xml)
- Output: token stream
  - XML elements: SPRM tokens, contains Attribute tokens
  - XML attributes: Attribute tokens

LibreOffice Technology

DOCX import: model.xml syntax

- Parsed using a built-time Python script…
  - Used to be worse, in XSLT
- Concept:
  - Take the RNG schema (grammar / defines)
  - Add matching resource tags that define the token maps
- Example: framePr

Collabora Online

WW8 (.doc)

- Oldest Writer filter:
  - binfilter was even older, but it's removed
- Import/export somewhat shared
- Uses the internal API
- Code under sw/source/filter/ww8/
- Shared (doc, xls, ppt) parts:
  - filter/source/msfilter/
- Using doc-dumper may help

RTF (.rtf)

- Export is shared with DOC/DOCX:
  - Code under sw/source/filters/ww8/rtf*
- Import is shared with DOCX:
  - Code under writerfilter/source/rtftok/
  - Domain mapper is the same for RTF and DOCX
- Math:
  - Import generates OOXML tokens (RTF-specific part is inside the normal RTF tokenizer)
  - Export is shared with DOCX:
    - Code under starmath/source/rtf*

Tests

- What's easy: filter tests
  - Both import / export
  - Poke around with xray, then assert the UNO document model
- The rest is more challenging
  - We have uwriter, which has access to private sw symbols
  - Layout test: can assert the layout dump
  - UI tests: uiwriter, it tests the shell

LibreOffice Technology

UI

- Again, shared with other modules where makes sense
- Doesn't use the UNO API
- Input/output for the dialog is an SfxItemSet
- Own toolkit: VCL
  - Dialogs use the Gtk3 .ui format now
    - Glade is a GUI to edit those
  - If have to touch an older dialog: save + commit first
    - Only then perform your functional changes

LibreOffice Technology

Help

- Lots of help buttons on UI
- Typically every existing dialog has a related help page
- If you add a new UI element, makes sense to spend 5 minutes on updating the related help
  - Requires a --with-help build
- XML based, also stored in git, just different repo
- Offline help is generated from that

Collabora Online

LibreOffice Technology

Extending ODF

- ODF is really close to the UNO API what we offer
  - Typically 1 UNO property is 1 XML attribute in ODF
- If you extend the UNO API
  - Go ahead with updating the ODF filter
  - After implementation is ready:
  - See https://wiki.documentfoundation.org/Development/ODF_Implementer_Notes#LibreOffice_ODF_extensions
  - Submit a proposal to OASIS, so it can be part of the next version of the standard

Bookmarks

- Wiki: https://wiki.documentfoundation.org/Development/Writer
  - New feature checklist, ODF implementer notes, etc.
- sw README: http://opengrok.libreoffice.org/xref/core/sw/README
- Older Writer notes:
  - http://cgit.freedesktop.org/libreoffice/build/tree/doc/sw-flr.otl?h=master-backup
  - http://cgit.freedesktop.org/libreoffice/build/tree/doc/sw.txt?h=master-backup