Collabora Productivity

Async dialog execution – What it is and why it's needed

By Jan Holešovský
Collabora Productivity

kendy@collabora.com @JHolesovsky +holesovsky skype: janholes

LibreOffice Online: Server part

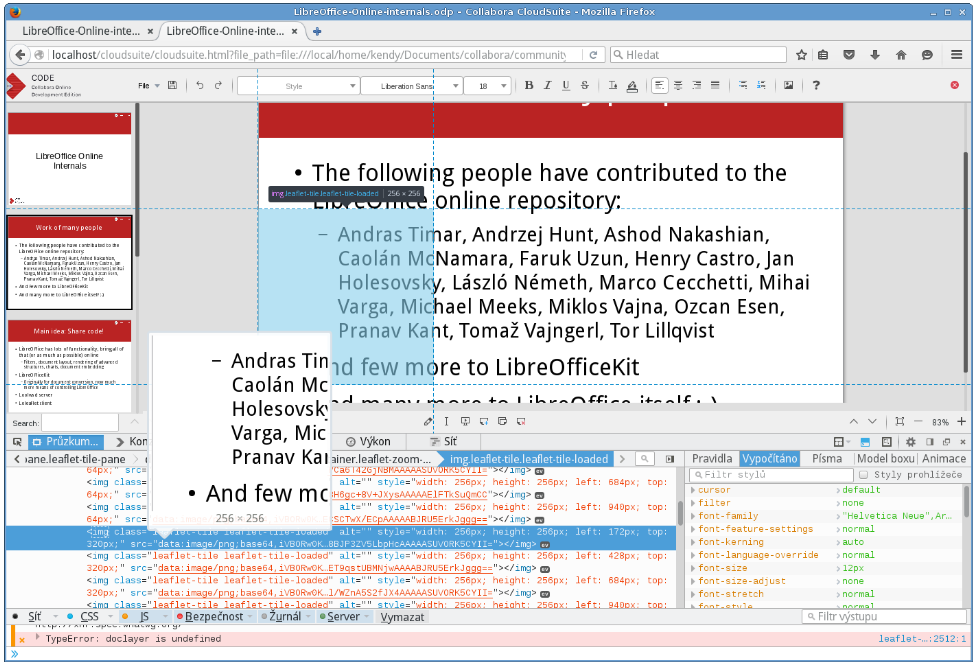
The Websocket Daemon - loolwsd

- Manages communication with file storage via WOPI protocol
- Spawns LibreOffice instances via LibreOfficeKit (LOK) and manages their lifecycle
  - These take care of rendering of the document
- Manages the user's interaction with the document
  - Passing commands to LOK
  - Passing callbacks back to the JavaScript clients
- All this is in C++

LibreOffice Online: Client part

Loleaflet

- Written in JavaScript, based on ‘leaflet’ - framework for map rendering
- Communicates with loolwsd
- The document itself consists of tiles:
- Menus, toolbars, status bar
    - All that is JS


LibreOffice-Online-internals.odp - Collabora CloudSuite - Mozilla Firefox
LibreOffice-Online-inte...
LibreOffice-Online-inte...
localhost/cloudsuite/cloudsuite.html?file_path=file:///local/home/kendy/Documents/collabora/community
Hledat
CODE
Collabora Online Development Edition
File
Style
Liberation Sans
18
LibreOffice Online Internals
Work of many people
Main idea: Share code!
Search:
img.leaflet-tile.leaflet-tile-loaded | 256 × 256
83%
Průzkum...
Kon
Výkon
Síť
ane.leaflet-tile-pane
ainer.leaflet-zoom-...
img.leaflet-tile.leaflet-tile-loaded
Pravidla
Vypočítáno
Písma
Model boxu
Animace
Filtr stylů
Styly prohlížeče
Síť
CSS
JS
Bezpečnost
Žurnál
Server
Vymazat
Filtr výstupu

- The following people have contributed to the LibreOffice online repository:
    - Andras Timar, Andrzej Hunt, Ashod Nakashian, Caolán McNamara, Faruk Uzun, Henry Castro, Jan Holesovsky, László Németh, Marco Cecchetti, Mihai Varga, Michael Meeks, Miklos Vajna, Ozcan Esen, Pranav Kant, Tomaž Vajngerl, Tor Lillqvist
- And few more to LibreOfficeKit
- And many more to LibreOffice itself :)

```
<img class="leaflet-tile leaflet-tile-loaded" alt="" style="width: 256px; height: 256px; left: 172px; top:
320px;" src="data:image/png;base64,iVBORw0K…8BJP3ZV5LbpHcAAAAASUVORK5CYII="></img>
<img class="leaflet-tile leaflet-tile-loaded" alt="" style="width: 256px; height: 256px; left: 428px; top:
320px;" src="data:image/png;base64,iVBORw0K…ET9qstUBMNjwAAAABJRU5ErkJggg=="></img>
<img class="leaflet-tile leaflet-tile-loaded" alt="" style="width: 256px; height: 256px; left: 684px; top:
320px;" src="data:image/png;base64,iVBORw0K…l/WZnA5S2fJX4AAAAASUVORK5CYII="></img>
```

```
cursor                     default
filter                     none
font-family                "Helvetica Neue",Ar…
font-feature-settings      normal
font-kerning               auto
font-language-override     normal
font-size                  12px
font-size-adjust           none
font-stretch               normal
```

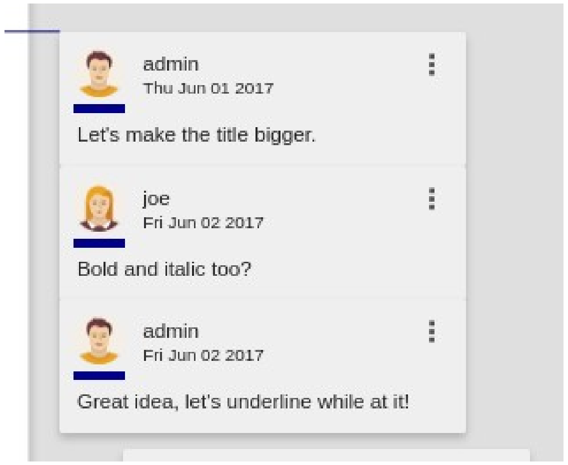
```
TypeError: doclayer is undefined                leaflet-…:2512:1
```

But: it’s very impractical to reimplement everything in JS...

Finding the Right Balance: JS vs. Core

Initially everything was rendered by LibreOffice

- In the early prototypes – no tiles, just gtk broadway
- Then we decided to use the tiled approach
- Cursors, selections – all that turned to be impractical in tiles, and we started rendering that separately, in an overlay
- Comments and redlining were next, those needed too much interaction when in tiles
  - Also they look better in JS (possibility to animate etc.)



admin
Thu Jun 01 2017
Let's make the title bigger.

joe
Fri Jun 02 2017
Bold and italic too?

admin
Fri Jun 02 2017
Great idea, let's underline while at it!

But what about dialogs?

We started adding JS ones

- Find / replace, special character, insert table, …
- Lengthy process! Needed something better…

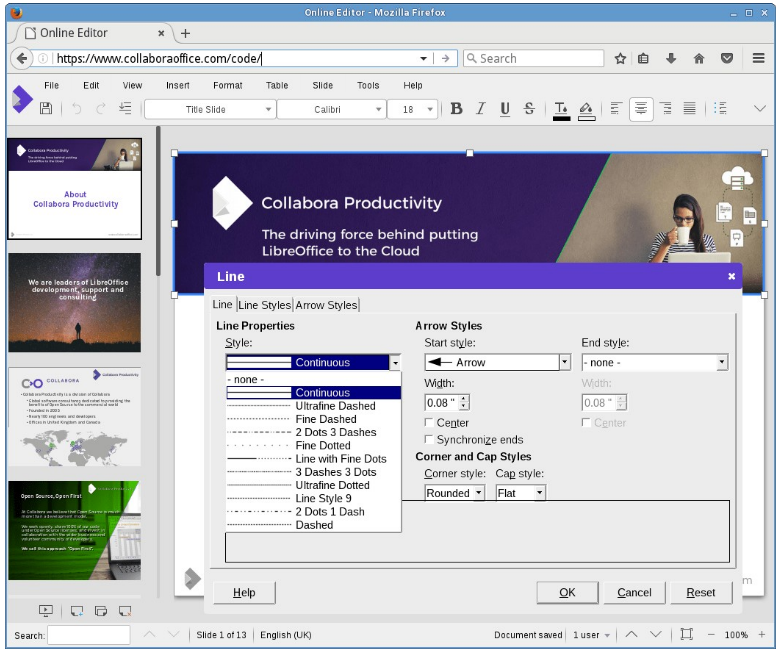
Dialog tunneling!

- Just reuse all the dialogs that are already there in LibreOffice
- The plan: Let the core render them, and pass them as bitmaps to Online
  - Nearly a year later: finally done ;-)
  - Most of the hard work done by Pranav Kant, big thanks!

Working!

The following features are now exposed

- Advanced character, paragraph and page properties
- Line, fill, cell properties, etc.
- All that collaboratively!



Online Editor - Mozilla Firefox

https://www.collaboraoffice.com/code/

Collabora Productivity

The driving force behind putting LibreOffice to the Cloud

Line

Line | Line Styles | Arrow Styles

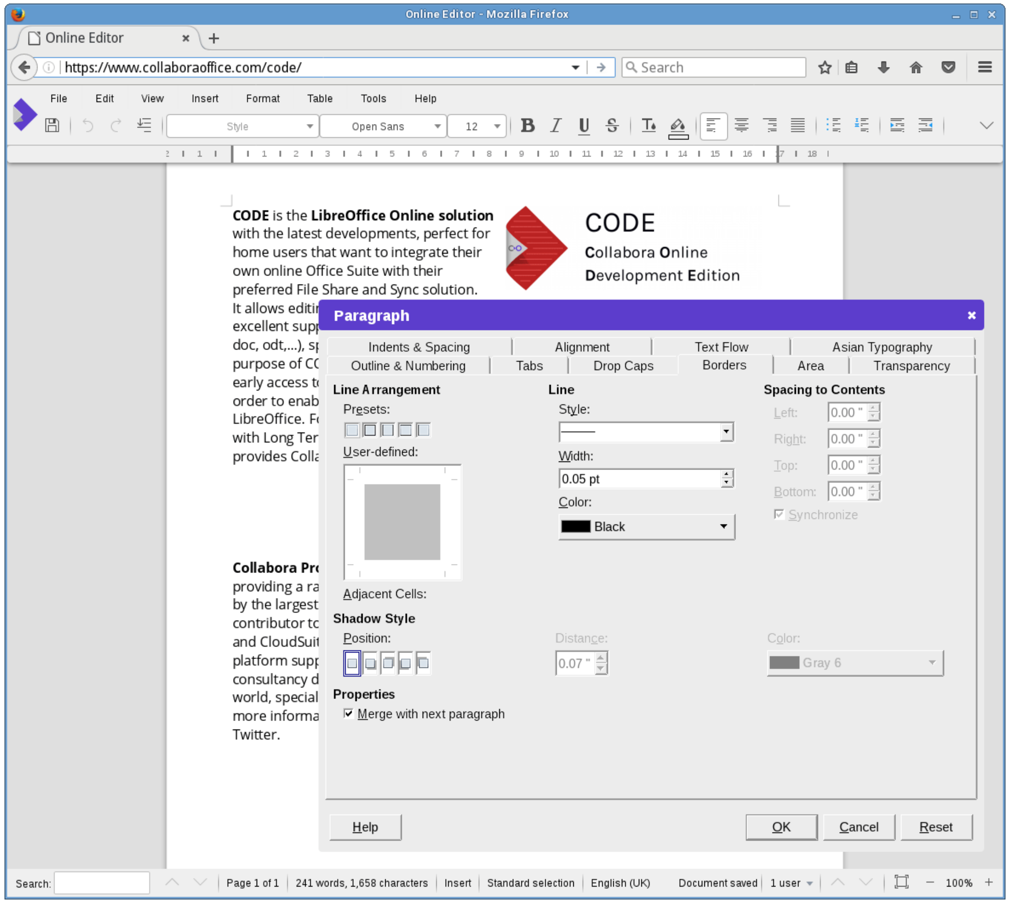
Line Properties
Style: Continuous
- none -
Continuous
Ultrafine Dashed
Fine Dashed
2 Dots 3 Dashes
Fine Dotted
Line with Fine Dots
3 Dashes 3 Dots
Ultrafine Dotted
Line Style 9
2 Dots 1 Dash
Dashed

Arrow Styles
Start style: Arrow
End style: - none -
Width: 0.08 "
Width: 0.08 "
Center
Synchronize ends
Center

Corner and Cap Styles
Corner style: Rounded
Cap style: Flat

Help | OK | Cancel | Reset

Slide 1 of 13 | English (UK) | Document saved | 1 user | 100%



Online Editor - Mozilla Firefox

https://www.collaboraoffice.com/code/

CODE is the **LibreOffice Online solution** with the latest developments, perfect for home users that want to integrate their own online Office Suite with their preferred File Share and Sync solution.

CODE
Collabora Online
Development Edition

Paragraph

Indents & Spacing | Alignment | Text Flow | Asian Typography | Outline & Numbering | Tabs | Drop Caps | Borders | Area | Transparency

Line Arrangement
Presets:
User-defined:
Adjacent Cells:

Line
Style:
Width: 0.05 pt
Color: Black

Spacing to Contents
Left: 0.00 "
Right: 0.00 "
Top: 0.00 "
Bottom: 0.00 "
Synchronize

Shadow Style
Position:
Distance: 0.07 "
Color: Gray 6

Properties
☑ Merge with next paragraph

Help | OK | Cancel | Reset

Page 1 of 1 | 241 words, 1,658 characters | Insert | Standard selection | English (UK) | Document saved | 1 user | 100%

Technical Details

How Does it Work?

Nearly everything is done down in VCL

- Added various callbacks – dialog created, invalidate, etc.
- Reusing the dialog screenshotting feature for rendering the content
- Added a concept of LOKNotifier
  - Most of the LOK notification is done in sfx2 – but that is a higher layer
  - LOKNotifier is an interface that is instantiated in sfx2, but can be used in VCL – for the notifications about dialog creation, what was invalidated, where to paint
- LibreOfficeKit extended accordingly

LibreOfficeKit Extensions for Dialog Tunneling

Methods

- void paintWindow(unsigned nWindowId, unsigned char* pBuffer, const int x, const int y, const int width, const int height)
- void postWindow(unsigned nWindowId, int nAction)
  - General events, so far only closing the window
- void postWindowKeyEvent(unsigned nWindowId, int nType, int nCharCode, int nKeyCode)
- void postWindowMouseEvent(unsigned nWindowId, int nType, int nX, int nY, int nCount, int nButtons, int nModifier)
- void postWindowMouseEvent(unsigned nWindowId, int nType, int nX, int nY, int nCount, int nButtons, int nModifier)

Callbacks

- LOK_CALLBACK_WINDOW, with a JSON payload
  - Indicating actions like “created”, “title_changed”, “size_changed”, “invalidate”, “cursor_invalidate”, “cursor_visible” and “close”

Language Support

One document can be co-edited by multiple users

- And each of them can have their UI in a different language
- LibreOffice used static objects for the text resources
- ~All the places had to be converted:

```
-    static std::locale loc(Translate::Create("cui"));
-    return Translate::get(pKey, loc);
+    return Translate::get(pKey, Translate::Create("cui"));
```

- Similarly SfxModule had to be adapted to be able to switch language when the view switches to a different user

Converting dialogs to async

Modal Dialogs

They call Execute() which blocks

- Not that events would stop flowing – Yield() called inside Execute()
  - Editing still works, AND two (or more) users can open the same dialog just fine from different views!
- The problem is when they are to be closed & the changes have to be applied
  - All the Execute()'s have to end first before the execution continues
  - Problem! - one of the users can go for lunch in the meantime, and the other never gets the changes applied

Modal → Modal Async Execution

The solution is to convert the modal dialogs to async

- They still stay modal, but do not block in Execute() any more
- LibreOffice already had StartExecuteModal which was working fine, but the converting code was leading to big amount to changes
- Introduced a new StartExecuteAsync() with a lambda – thanks Michael Meeks

```
-   ScopedVclPtr<SfxAbstractTabDialog> pDlg(pFact→CreateScAttrDlg(...));
+   VclPtr<SfxAbstractTabDialog> pDlg(pFact→CreateScAttrDlg(...));
[...]
-   short nResult = pDlg→Execute();
+   std::shared_ptr<SfxRequest> pRequest(new SfxRequest(rReq));
+   pDlg->StartExecuteAsync([=](sal_Int32 nResult){
    [... the code that was previously following after Execute ...]
+   });
```

Non-modal dialogs

Work out of the box

- No conversion to asynchronous is necessary
- Usually they are using the sfx infrastructure
  - Using the ChildWindow::RegisterChildWindow(SID_<name>)
- Show / hidden using ToggleChildWindow(SID_<name>)
  - In the main event loop, no Execute() => no problem

Usual Caveats

“I issued a dialog via .uno: command, but it does not appear in the Online”

- Most probably the dialog does not have a parent – uses nullptr
- Solution: Assign it a parent, ideally window of the view shell

“The dialog does not switch languages for users”

- Static variable holding the locale; but less of a problem these days after conversion to gettext – thanks Caolán McNamara
- Solution: Find it & de-static-ize

Anything else

- Happy to help on the dev mailing list or on the IRC!

Collabora Productivity

Thank You for Listening!

And the following people for working on this:

Pranav Kant (main author of the tunnelling), Henry Castro, Michael Meeks

By Jan Holešovský

kendy@collabora.com @JHolesovsky +holesovsky skype: janholes