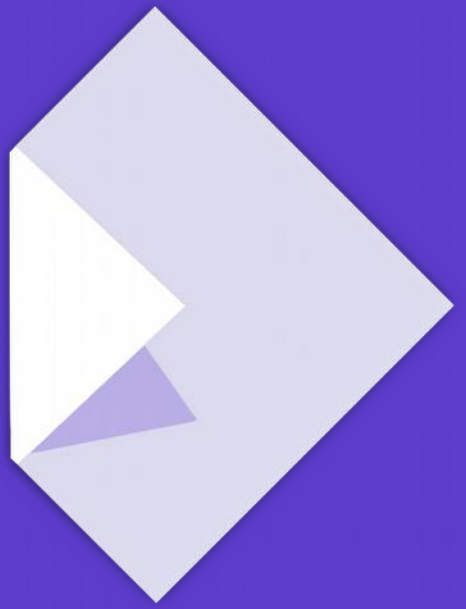
FOSDEM 2020

Integrate Collabora Online with Web Applications



Collabora Productivity

By Andras Timar

Collabora Productivity

andras.timar@collabora.com @timar +timar74

Collabora Online

- Uses LibreOffice/Collabora Office under the hood for document rendering
- Not a standalone application…
  - Does not implement user authentication
  - Does not implement document storage
  - Designed to be a building block of a web application that needs
    - Document viewer
    - Document editor
    - Collaborative editing
  - ODF, OOXML, plus many more file formats are supported
- **Therefore integration is the key to success!**

How to experience a success story?

- Install CODE on a dedicated server
  - https://www.collaboraoffice.com/code/
- Setup the SSL for the https access
- Install the integration between your document storage and the Online
  - Or if your are integrating yourself, implement the REST endpoints
  - And modify your webapp to add iframe for the Online
- Enjoy the success! :-)

  Or maybe you have an additional need before you are completely happy with Online in your webapp?

Alfresco

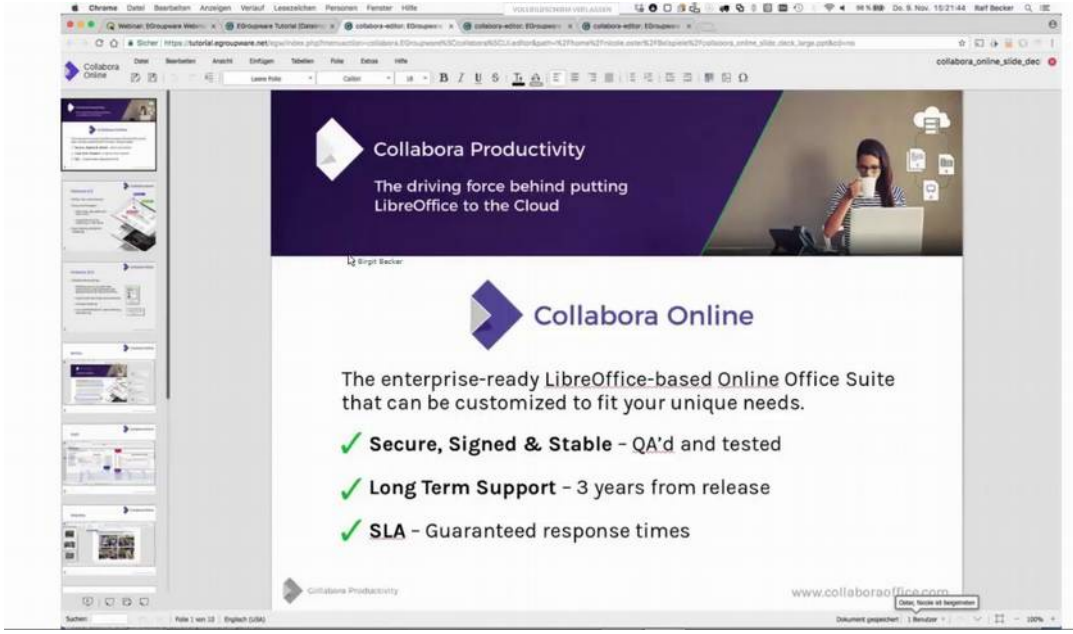
Alfresco connector available thanks to Arawa

- https://github.com/ArawaFr/libreoffice-online-repo
- Implemented in Java
- To match their integration needs, Collabora has implemented support for monitoring the loolwsd health
    - Several Online nodes can connect to one central place that collects the state

EGroupware

Connector available thanks to eGroupware from their repository

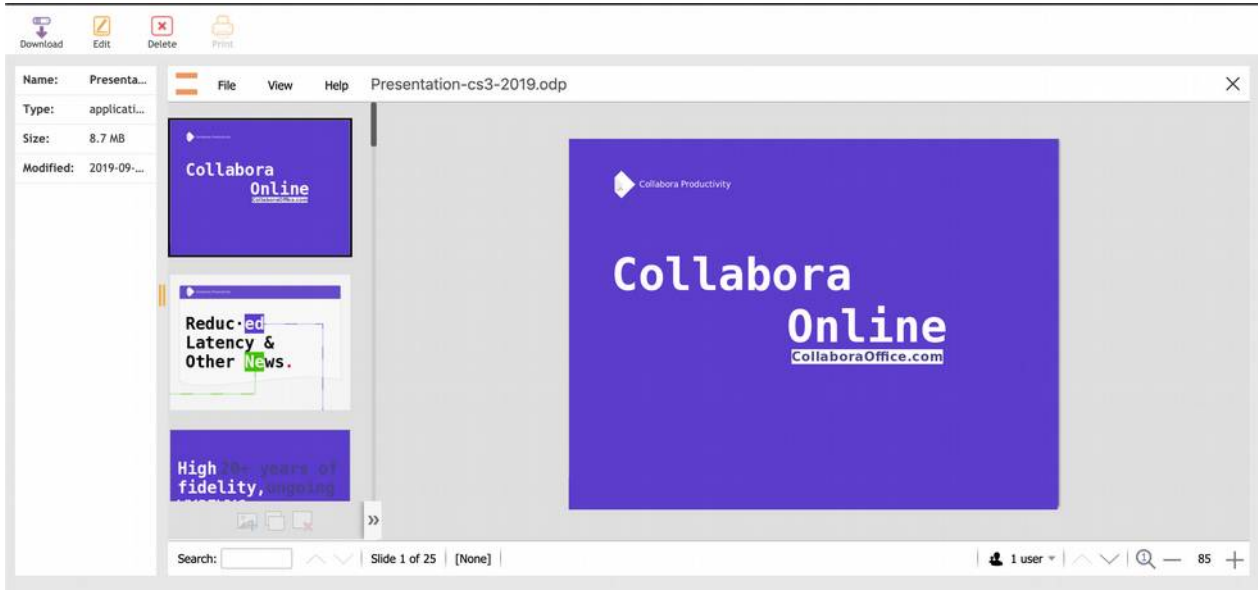
- https://github.com/EGroupware/collabora
- Implemented in PHP + JavaScript
- To match their integration needs, we have
  - Improved the printing in Firefox
  - Updated dockerfiles



Kolab

Connector available thanks to Kolab themselves.

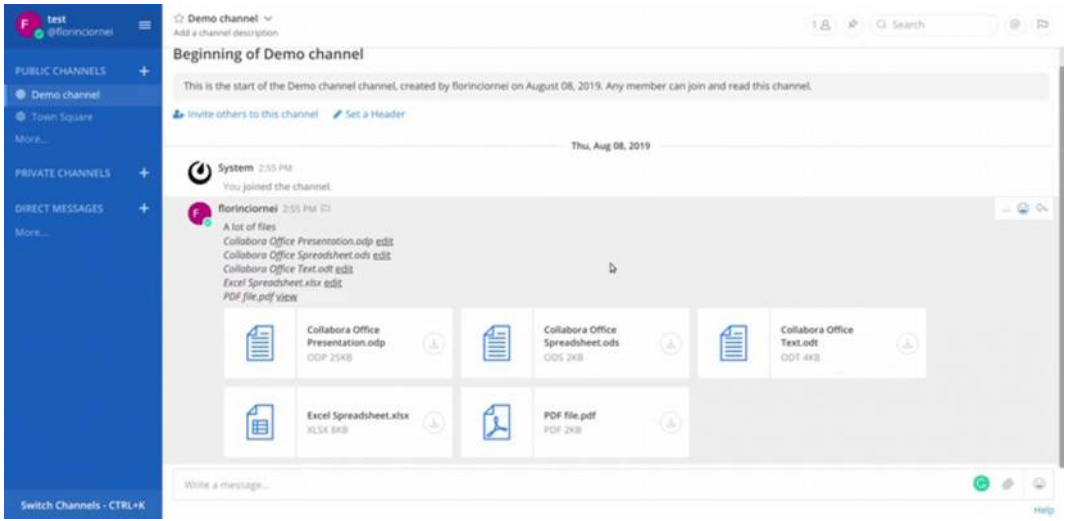
- https://git.kolab.org/source/wopi/
- To match their integration, we had to expose various functionality via postMessage
  - Like closing all the views from the master view, downloading or printing



Mattermost

Connector written by Florin Ciornei (Collabora)

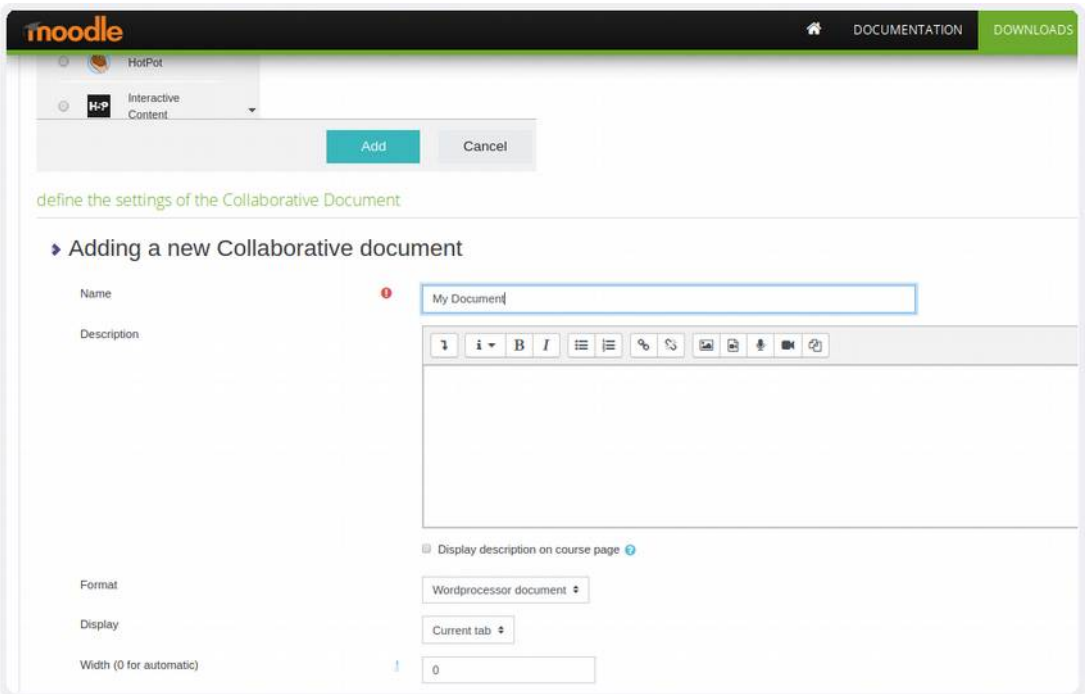
- https://github.com/CollaboraOnline/collabora-mattermost
- Written in Go
- Supported document formats can be opened from the chat in a popup editing window
- Collaborative editing works
- Result is saved back to the attachment



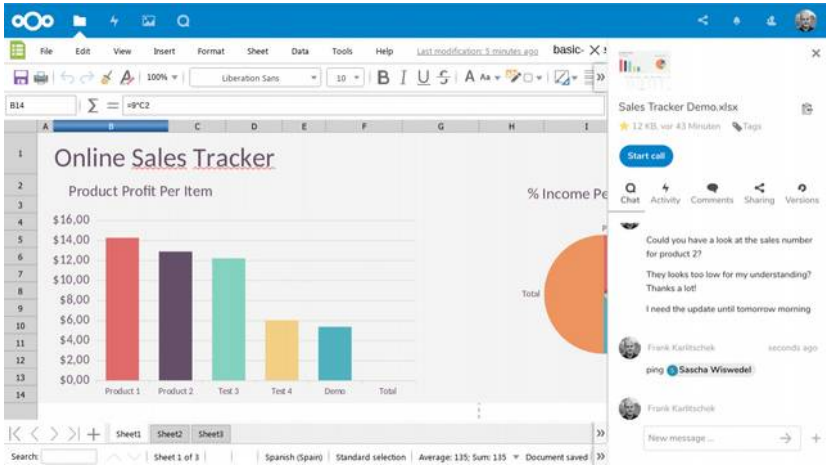
Moodle

Plugin to add a live document editor to the Moodle e-learning course

- https://github.com/learnweb/moodle-mod_collabora
- students and teachers can have “in-class collaboration” on documents



Nextcloud



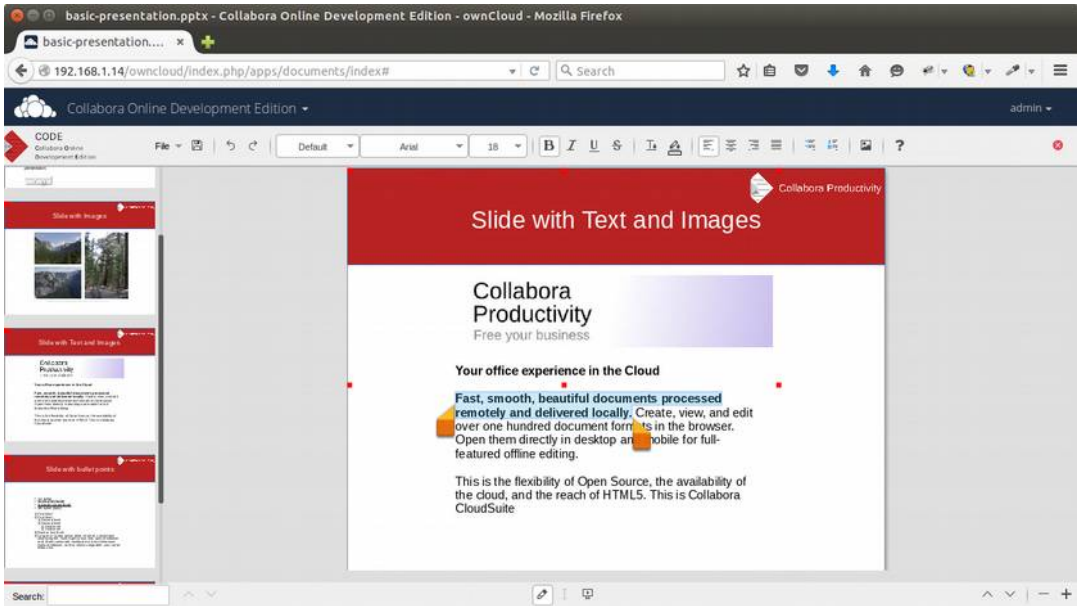
Connector originally developed by Collabora, but now with features from Nextcloud

- https://github.com/nextcloud/richdocuments
- Implemented in PHP and JavaScript
- Many features in the Online were implemented to match the Nextcloud's integration needs, like:
  - Specification of avatars in CheckFileInfo
  - Support for templates – TemplateSource in CheckFileInfo
  - /hosting/capabilities – for early check of whether a particular version supports some features or not
  - And many more…

ownCloud

Originally implemented by Collabora, now with features from ownCloud.

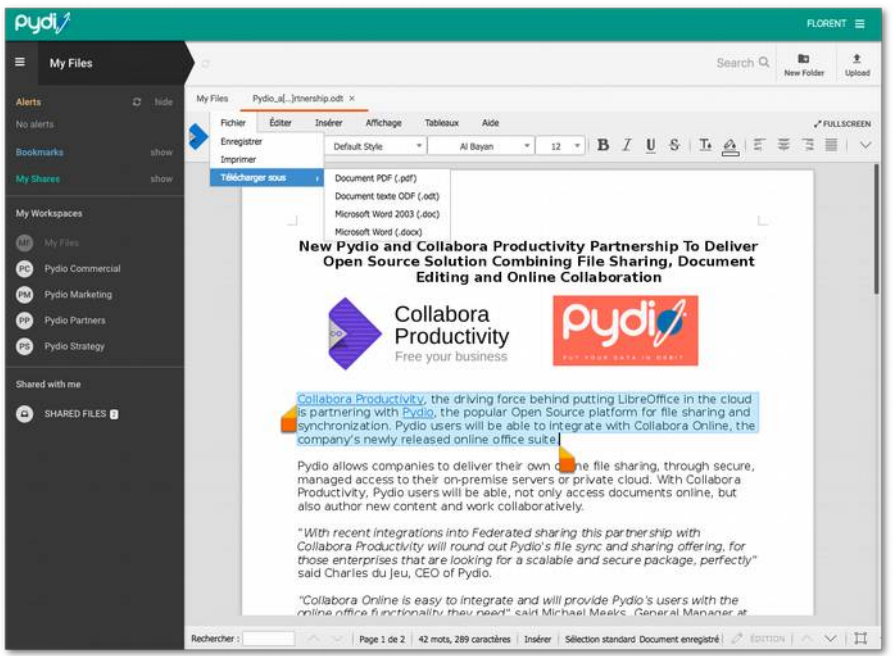
- https://github.com/owncloud/richdocuments
- Implemented in PHP and JavaScript
- To match their integration needs, we are improving Watermarking substantially
  - So that each user in a document can have a different, dedicated watermark



Pydio

Available thanks to Pydio

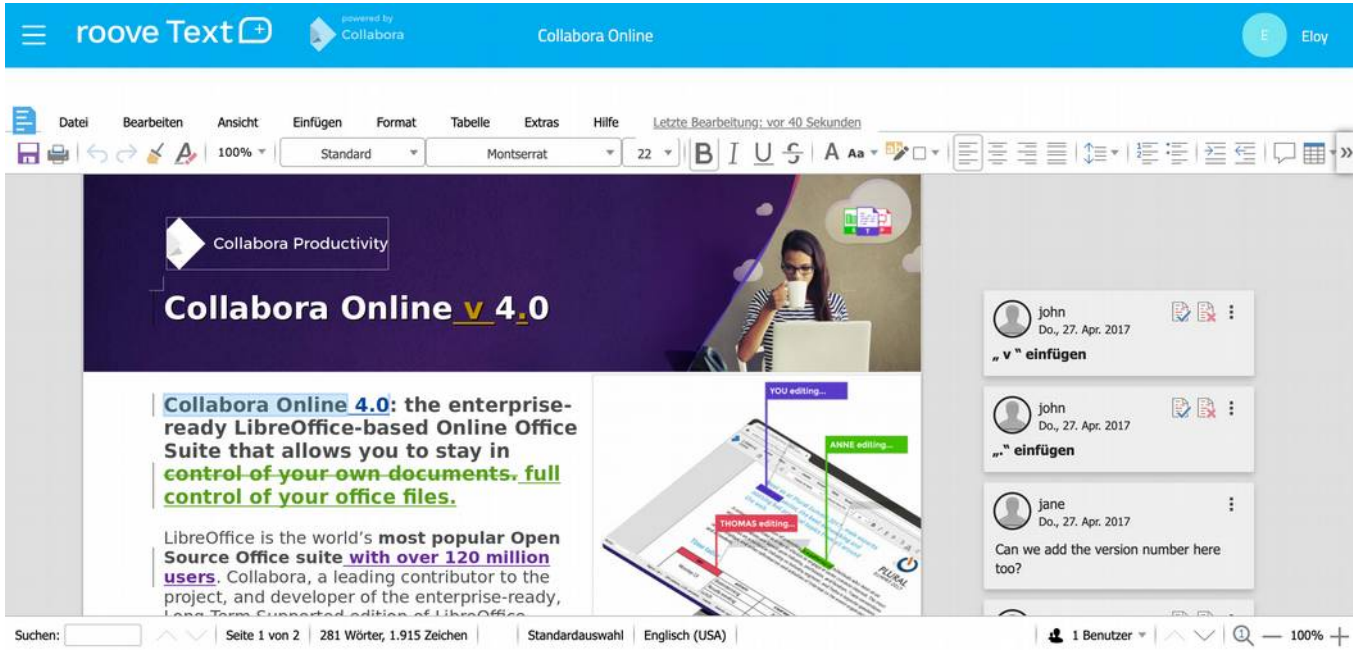
- https://github.com/pydio/pydio-core/tree/develop/core/src/plugins/editor.libreoffice
- Implemented in PHP and JavaScript
- To match their integration needs, we had to extend what characters can be used in the access_token



Roove

Connector implemented by SoftDistribution

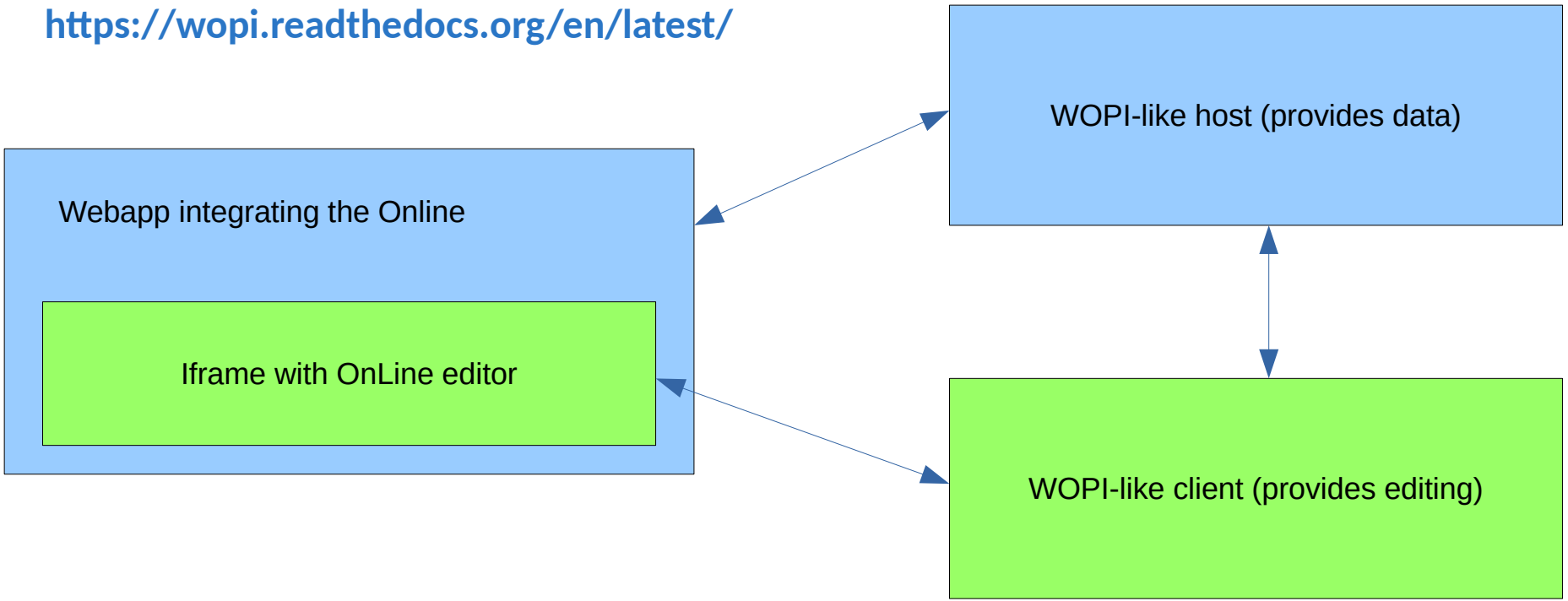
- For their integration, we had to improve Save as and full-screen handling of presentations.



Need to create an own integration?

The Online uses a WOPI-like protocol

https://wopi.readthedocs.org/en/latest/



REST endpoints

To download the file:

- GET https://<WOPI host URL>/<...>/wopi*/files/<id>/contents?access_token=<token>

Upload back:

- POST https://<WOPI host URL>/<...>/wopi*/files/<id>/contents?access_token=<token>

CheckFileInfo to provide information about the file:

- https://<WOPI host URL>/<...>/wopi*/files/<id>?access_token=<token>

Security token

To be able to access files securely, an authentication token has to be passed to the Online

- From the Online point of view, it can be any random number / string that will be passed as part of the URL when accessing the document storage

Should be generated according to the user who is logged in + the document

- But don’t make it possible to guess the token!
- The ideal is to maintain a list on the server + have a random number (mapped to the element in the list) as the token

JavaScript part – embedding the iframe

The Connector connects to the discovery service

- To get the information about how to initiate the iframe, the 1st call has to be an access to a “discovery” xml
  - In theory, different document types can be served by different servers

Then create the iframe that contains the Online

- Has to be provided with the access token via a POST request
- And then only gets called back when the document is closed again

Recommended steps

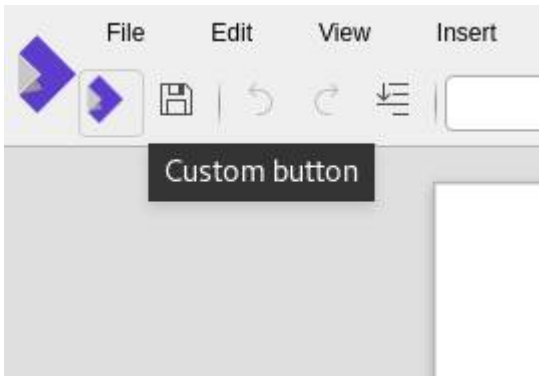
- Add WOPI REST endpoints to your web service, for the moment returning only a “Hello World” message.
- Implement the CheckFileInfo endpoint
- Now you can see a constructed document containing just “Hello World” in the Online in a simple HTML page:
  - <html><body>
    <form action="WOPISrc=https://<WOPI host URL>/<...>/wopi/files/<id>" enctype="multipart/form-data" method="post">
    <input name="access_token" value="test" type="hidden"/>
    <input type="submit" value="Load Collabora Online"/>
    </form></body></html>
- In your webapp, create an iframe that embeds the Online
- Implement the real access_token handling
- Update the REST endpoints to actually load real data
- Implement the PutFile endpoint to really save the data

Need to further tweak the functionality?

Extending without Touching the Online Code

Sometimes the functionality is too specific to the integration via postMessage

- It’s possible to add a button to the toolbar via a message
- And react on the user pressing it
- See loleaflet/reference.html



postMessage API

For signalling from the iframe back to the parent frame.

- You can use this after all the JS is loaded – which is signalled by App_LoadingStatus with Status “Frame_Ready”. When that happens, the parent has to send Host_PostmessageReady.
- Then for inserting a button, the host can use Insert_Button
- Each time the button is pressed, the host receives a message Button_Clicked with the id of the button.

CheckFileInfo extensions – to enable or disable features

Most of them were added thanks to a customer request:

- **HidePrintOption, HideSaveOption, HideExportOption:** Hide the appropriate options in the file menubar
- **DisableExport:** Disables export functionality in backend. If set to true, HideExportOption is assumed to be true
- **DisableCopy:** Disables copying from the document in libreoffice online backend. Pasting into the document would still be possible. However, it is still possible to do an "internal" cut/copy/paste.
- **DisableInactiveMessages:** Disables displaying of the explanation text on the overlay when the document becomes inactive or killed. With this, the JS integration must provide the user with appropriate message when it gets Session_Closed or User_Idle postMessage's.
- **DownloadAsPostMessage:** Indicate that the integration wants to handle the downloading of pdf for printing or svg for slideshows or experted document, because it cannot rely on browser's support for downloading.
  - When this is set to true, the user's eg. Print action will trigger a postMessage called Download_As, with the following JSON in the Values:
    `{ Type: 'print'|'slideshow'|'export', URL: ...url you use for the actual downloading... }`
- **EnableOwnerTermination:** If set to true, it allows the document owner (the one with OwnerId =UserId) to send a 'closedocument' message (see protocol.txt)
- **UserExtraInfo:** JSON object that contains additional info about the user, namely the avatar image.
  - Example: 'UserExtraInfo' => ['avatar' => 'http://url/to/user/avatar', 'mail' => 'user@server.com']
  - Note: There is strict Content Security Policy that restricts image resources (img-src), therefore the avatar URL must not violate the CSP, otherwise it will show as broken images.
- **WatermarkText:** If set to a non-empty string, is used for rendering a watermark-like text on each tile of the document
- See also: https://github.com/LibreOffice/online/blob/master/wsd/reference.md

Partner with Collabora!

Partner with Collabora

We will help you when you have any specific needs

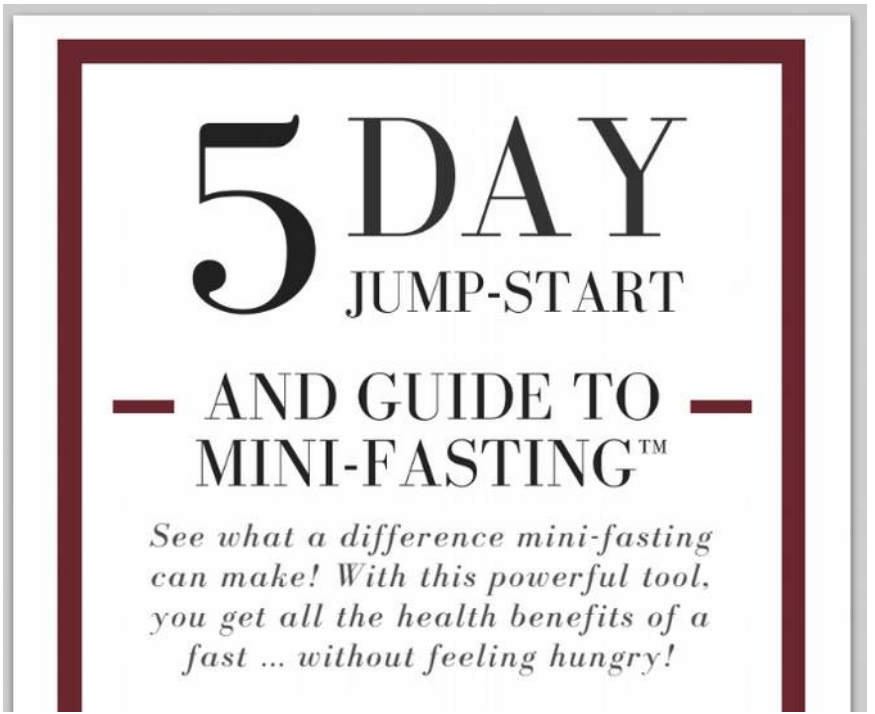
- Many features were implemented thanks to customer or partner demand

Additional examples not covered in the previous slides:

- PutFile extensions: Additional headers, like X-LOOL-WOPI-IsAutosave, X-LOOL-WOPI-IsModifiedByUser
- New authentication mechanism
  - access_header instead of access_token
  - And a related cookies handling
- Many more smaller ones…

PDFium – better way how to render PDF in the Online

Client needed a better rendering of PDF's, Collabora implemented it:



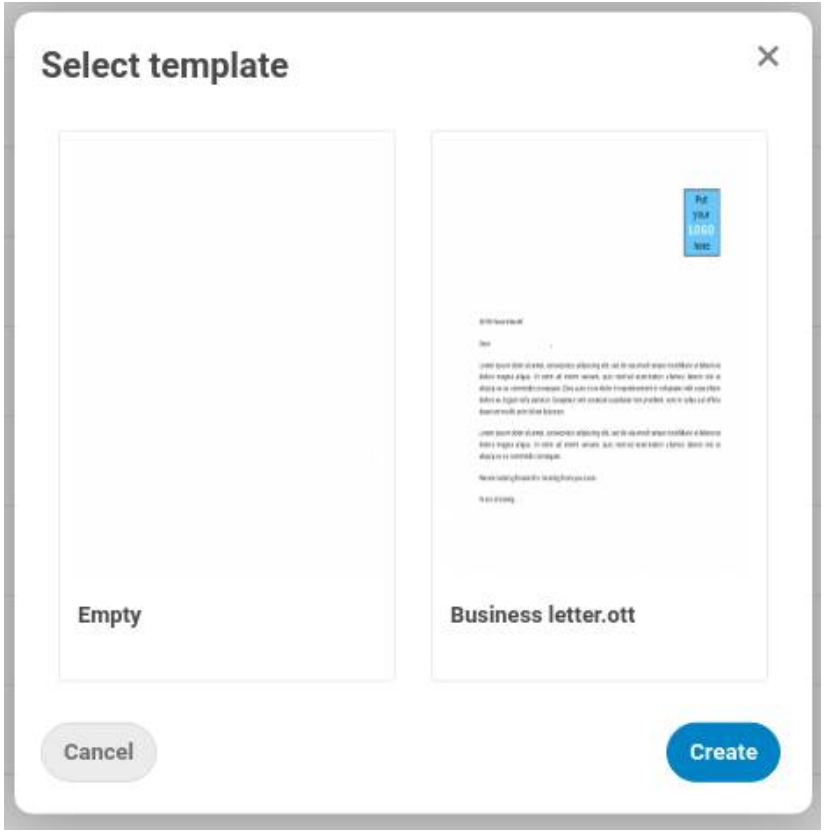
Creating a new document from a template

Now the integrations can create new documents from templates too

- Provide the template location as TemplateSource in CheckFileInfo
- Then the GetFile only informs about the ID, but actually the file is created from the TemplateSource template, and saved into the ID

/hosting/capabilites – similar to /hosting/discovery created to inform if TemplateSource is available (and other useful things)

- Returns a JSON with hasTemplateSource: true if the functionality is available



Collabora Productivity

Summary

- Use an existing integration or create your own
- Tell us what you are missing
- We will prioritize and implement
- Thank you for listening!

By Andras Timar

andras.timar@collabora.com @timar +timar74