Native comments & change tracking in LibreOffice Online

By Pranav Kant
Software Engineer at Collabora Productivity
pranavk@collabora.com
Problems with previous approach

Trigger unnecessary rendering of document’s tiles

- Upon user selecting a comment, the anchor lines would become bold and trigger a tile invalidation in document
  - Not ideal for Online where every unnecessary invalidation has a huge performance cost

- Invalidation would happen even if the document content is unchanged
Problems with previous approach

Harder comment navigation

- Imagine a page full of range comments with range positions completely out of sync with comments’ position on the sidebar; finding which comment belongs to which text is hard

- Ideal would be to slide the comments to their range position as they are clicked
Problems with previous approach

Anchor lines all over the document

- Interfering with user experience
- Especially when there are lot of comments
This is a simple document.

This is a simple document.

This is a simple document.

This is a simple document.

This is a simple document.

This is a simple document.

This is a simple document.

This is a simple document.

This is a simple document.

This is a simple document.
Problems with previous approach

Harder to maintain

- Involvement of editing.
- Typing comment in one view interfering with other views
  - Special cases to handle that in LOK
- And many other problems; a lot of bug fixing involved around comments earlier
And no change tracking comments support
So, this is what we wanted

- Better UX
  - Quicker response times
    - Avoid unnecessary round-trips with every typed character
  - Animations!!!
- Better DX
  - Low maintenance costs
  - Less developer time fixing comments related bugs
  - Integrated change tracking comments
    - Easier to accept/reject changes, add comment, etc.
Implementation
Implementation

- Optional feature
  - Can be disabled by an option in LOK
  - Other LOK clients still using in-tiled comment rendering
  - Pass --enable-tiled-annotations in GTV

- Wrapped all comment functionality under the set of LOK APIs
Implementation

- Single LOK command to get all the comments (with their parent-child relationship)
  - `getCommandValues(ViewAnnotations)`
  - JSON array containing all the comments
Implementation

- Augmented existing UNO commands
  - For insertion/removal/modification of comments
  - .uno:InsertAnnotation, .uno:RemoveAnnotation, .uno:ModifyAnnotation now return/accept a ‘Id’ parameter to identify the comment
Implementation

- Callbacks to notify LOK clients
  - For insertion/removal/modification of comments
  - LOK_CALLBACK_COMMENT
    - With a JSON containing the information about the comment
```
{
  "comment": {
    "action": "Add",
    "id": "11",
    "parent": "4",
    "author": "Unknown Author",
    "text": "This is a beautiful comment",
    "dateTime": "2016-08-18T13:13:00",
    "anchorPos": "4529, 3906",
    "textRange": "1418, 3906, 3111, 919"
  }
}
```
Implementation (Writer)

- Only available module with support for reply comments
- Parse the internal data structures and convert it to JSON preserving the parent-child relationship
- Calculate the rectangles in twips for range comments
  - And put them in JSON
- Give unique Ids to each SwPostItField object to be able to identify comments from the LOK API
  - Needed to later modify or remove a specific comments
Change tracking

- Simple to expose them through a LOK API
  - Support for commenting was already present in core
- Intuitive UI – can accept/reject changes easily
- Add comments easily
- And animate them to their actual change
Client side
Using comments & change tracking API

- Interpret the JSON and feed it to the layouting algorithm
- Sort comments by their anchor position in the document
  - Keep them as closer as possible to actual content
  - and animate them when they get the focus near to their actual content
- Send data to backend only after comment is committed/saved
  - Whole comment data is sent in one go
  - Small performance win
- Use the change tracking API to fetch changes and their comments and integrate in sidebar with comments
More ideas for future

- Better root-reply comment relationship
  - In ODF, all the adjacent `<office:annotation>` are treated as reply comments
    - Limitation: Only one comment thread per anchor position
    - Give Id to each `<office:annotation>` element and link reply comments with a new attribute, `<office:parent-annotation-name>`
  - Ability to resolve comments
    - Some new `office:resolved` attribute?
More ideas for future

- Ability to reply to comments and resolve them in Calc and Impress
  - Calc: “Notes” vs. “Comments”?  
- Change tracking comment threads
  - ... instead of only one comment per change
Any Questions?

By Pranav Kant
pranavk@collabora.com