SCREENSHOTTING YOUR WAY THROUGH THE CODEBASE

KATARINA BEHRENS, ARMIN LE GRAND
CIB LABS
LIBREOFFICE CONFERENCE BRNO
AUGUST 24, 2017

SCREENSHOTTING YOUR WAY THROUGH THE CODEBASE

1. WHY SCREENSHOTS, ANYWAY?
2. WHAT HAVE WE DONE AND HOW?

WHY SCREENSHOTS, ANYWAY?

IF ALL ELSE FAILS, READ HELP (F1)

Helpful?

Better :)

Figure 5: Text Import dialog
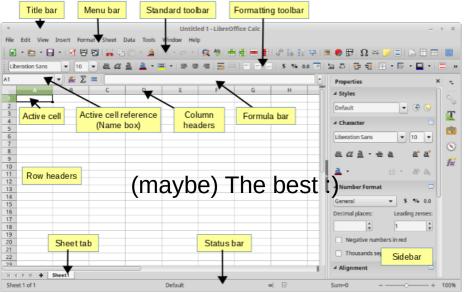
IF ALL ELSE FAILS, READ HELP (F1)

Figure 1: Calc main dialog

Menu bar

When you select an item on the Menu bar, a sub-menu drops down to show commands. You can also customize the Menu bar; see *Chapter 14 Customizing LibreOffice* for more information.

- **File** – contains commands that apply to the entire document; for example, *Open*, *Save*, *Wizards*, *Export as PDF*, *Print*, *Digital Signatures*.
- **Edit** – contains commands for editing the document; for example, *Undo*, *Copy*, *Changes*, *Fill*, *Plug-in*.

JOURNEY THROUGH THE LAYERS OF LEGACY HELP CONTENT

➙ Help content in „self-invented" .xhp format

➙ Tedious, hard-to-maintan and hard-to-extend transformation XHP ➡ XSLT ➡ HTML

➙ Help-specific images part of icon themes

➙ not sustainable for larger amount of graphical content in many languages

➙ Challenging to insert more complex content (screenshot, image etc.) in an efficient, easily repeatable way

➙ for multiple platforms

➙ for multiple languages

GENERATING SCREENSHOT MUST BE

➜ Repeatable

➜ Generate an updated set of screenshots with every new release

➜ Platform- and environment-specific

➜ Different sets of screenshots for different platforms (Linux, Win, Mac) …

➜ … and different icon sets, themes

➜ Localizable

➜ Screenshots to match the locale of the help text

WHAT HAVE WE DONE AND HOW?

OPEN THAT DIALOG!

WAIT – THERE IS A FRAMEWORK OPENING DIALOGS ALREADY …

➔ UI-testing framework based on PyUNO (by Markus Mohrhard)

➔ Opens dialogs (by dispatching corresponding UNO commands)

➔ Simulates mouse/keyboard events on misc widgets

➔ But …

- performance issues

- verbose, fragile code

- challenging to debug PyUNO

SOLUTION: USE ABSTRACT DIALOG FACTORY

→ Open an empty document from cppunit test

→ Instantiate abstract dialog factory (contained in Writer/Calc/Impress docshell already)

→ Create any (fake) objects the dialog requires in order to open – SfxItems, strings etc.

→ Open the dialog

→ Use OutputDevice to take a screenshot in .png format

... BUT IT WAS MORE WORK THAN WE EXPECTED

➔ Fallback to ui-previewer (small C++ binary for previewing LibreOffice .ui files)

➔ To some extent usable screenshots, but ...

  ➔ Not coping with some custom widgets so well

  ➔ No way to specify input data

➔ Some dialogs blacklisted, crash with no input

STORE THE SCREENSHOTS

➜ Results in */workdir/screenshots* directory

➜ Directory structure to copy storage of .ui files

  ➜ */workdir/screenshots/cui/ui/areatabpage/AreaTabPage.png*

➜ Make them all: '*make screenshot*'

  ➜ a separate gbuild target

ENHANCE THE HELP TEXTS

GET 'EM IN

➔ Copy generated screenshots from *workdir* to a dedicated folder in help repository

➔ Extend Perl scripts packaging the images to collect content also from help repository

  ➔ previously only possible for *icon-themes* folder

➔ Embed images into help files

  ➔ I ❤ HelpAuthoring extension :P

➔ Images implemented as *vnd.libreoffice.image://* URL

  ➔ localization for free (with matching directory structure)

HIGHLIGHT AND ANNOTATE

LIVE DEMO

NEXT STEPS

WHAT'S NEXT?

- Improve the annotation process and markup
  - Possibly HelpAuthoring integration
- Automate updating the screenshots
  - for multiple languages
  - for multiple platforms

**THANK YOU THE DOCUMENT FOUNDATION
AND ITS DONORS
FOR FUNDING OUR WORK ON THIS**

THANK YOU!