LibreOffice from COLLABORA

Overview of Localization Infrastructure of LibreOffice

Andras Timar <andras.timar@collabora.com>

timar, #libreoffice-dev, irc.freenode.net

LibreOffice

Topics covered

- For developers
  - how to make strings (non-)localizable
- For translators
  - error checking
  - l10n workflow

Tips for developers

.ui files

- 'translatable' attribute
  - “yes” or “no”
  - when missing → not translatable
  - Glade sets translatable=”yes” in most cases, but not always
    - `$ bin/ui-translatable.sh <property name="label">_Edit...</property>`

      `Source: cui/uiconfig/ui/applyautofmtpage.ui^`
  - set translatable=”no” in order to avoid false positives
    - `<property name="label" translatable="no">toolbutton1</property>`

Help (.xhp) files

- 'localize' attribute
  - “true” or “false”
  - when missing → translatable
  - set localize=”false”, if something is not for translation
    - HID bookmarks
    - Code examples
- 'l10n' and 'oldref' attributes
  - obsolete, do not use them in new help text

Palettes

- colors, gradients, hatches, etc.
- XML files in extras/source/palettes/ are not directly localizable
- localized when loaded into a control, from resource (.res) files
- in .src define reference string as in XML and translatable string (English)

/core/extras/source/palettes/

standard.soh

```
13 <draw:hatch draw:name="Black 45 Degrees Wide" draw:style="single" draw:color="#000000"
draw:distance="0.2inch" draw:rotation="450"/>
```

/core/svx/source/dialog/

sdstring.src

```
1250 Text = "Black 45 Degrees Wide";
1305 Text [ en-US ] = "Black 45 degrees wide";
```

In case of unlocalized string

- Check:
  - whether the string is marked for localization in source code
  - whether the string is extracted from source code to. po
    - l10ntools/source/localize.cxx
    - .hrc files should be added list[] in passesPositiveList()
    - toplevel modules should be added to projects[] in includeProject()
  - whether the localizations are merged during build
    - make sure that the module is built with the right rule in makefiles
  - whether the localized targets are copied to install set
    - check workdir/InstallScriptTarget/setup_osl.ins or setup_osl.inf
    - add missing targets to .scp files in scp2 module
  - whether the string is actually translated in .po file

Tips for translators

Classification of translation errors

- Build breakers
  - e.g. XML errors in help or readme (missing/broken tags)
- Run-time errors
  - e.g. different spreadsheet functions have the same translation
- Cosmetics
  - Mistyped placeholders ($ARG1, %PRODUCTNAME)
  - Missing/added hotkeys (_)

Checking for translation errors

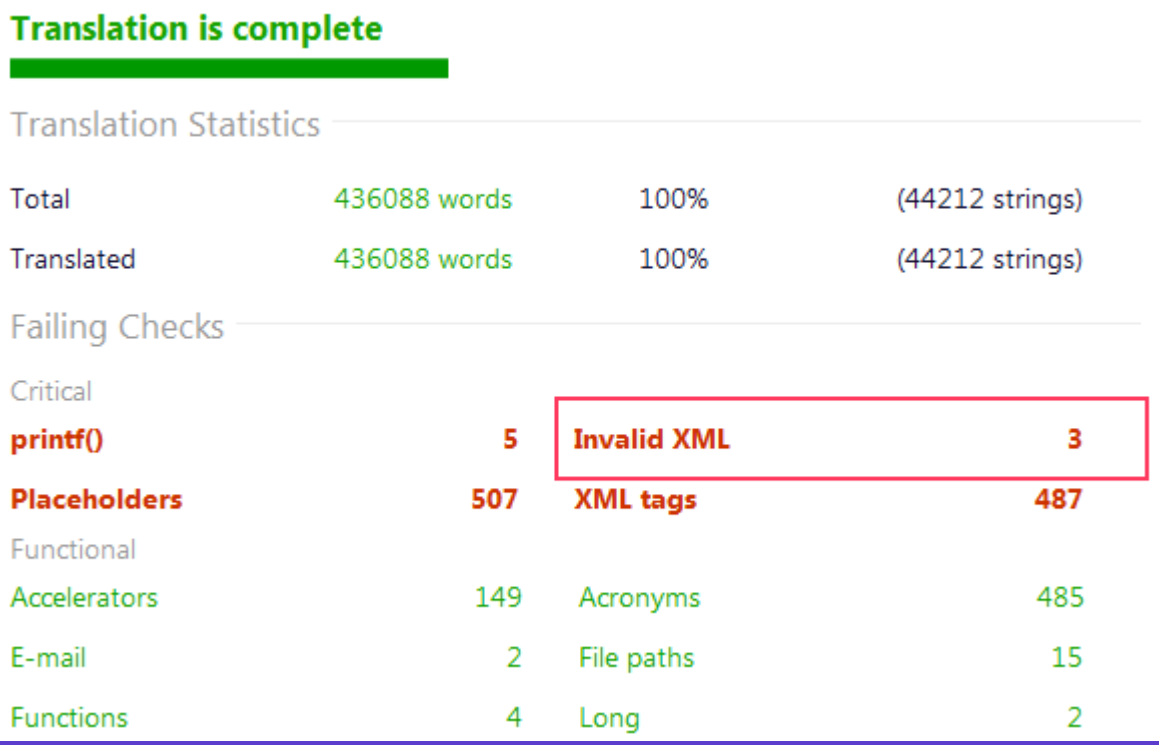
- Automatic check for build-breaker translation errors build-time (XML validation with libxml2)
- Invalid XML segments are discarded, en-US segments are used

Checking for translation errors

- Check of .po files before git commit/push
  - `LD_LIBRARY_PATH=instdir/ure/lib make cmd cmd=workdir/LinkTarget/Executable/pocheck`
  - Checks 4 types of translation errors
    - Translated style names must be unique (STR_POOLCOLL* and STR_POOLNUMRULE* in /sw/source/ui/utlui.po)
    - Translated spreadsheet function names must be unique
    - In instsetoo_native/inc_openoffice/windows/msi_languages.po where an en-US string ends with '|', translation must end with '|', too.
    - In starmath/source.po Math symbol names (from symbol.src) must not contain spaces

Checking for translation errors

- Pootle checks
  - many false positives, some make sense
  - most important: “Invalid XML”



LibreOffice l10n workflow

- make translations – creates fresh .pot files in workdir/pot
- Upload pot files to Pootle, update existing po files with new pot files.
- Let translators work on the files.
- Download po files from Pootle
- Format po files with a sed script to minimize size of diffs
- Run pocheck tool
- Commit and push

Impress Remote

- Source code went to a separate repo
  - http://cgit.freedesktop.org/libreoffice/impress_remote/
  - Android, iOS, Firefox OS
- Separate modules in Pootle
  - Impress Remote (Android)
  - Impress Remote (iOS)
  - Not all locales are supported by all mobile OS's

Impress Remote (Android) localization

- Use android2po tool to convert strings.xml to template.pot.
- Upload template.pot to Pootle, Impress Remote (Android) project.
- Update .po files (Update from templates).
- Let translators work on the files.
- Download files from Pootle, put them to locale/ folder, and run `a2po import`.

Impress Remote (iOS) localization

- Use prop2po tool from Translate Toolkit to to create pot files from strings files. e.g.:
  ```
  prop2po -P
  ios/iosremote/iosremote/en.lproj/Localizable.strings
  Localizable.strings.pot
  ```
- Upload pot files to Pootle, Impress Remote (iOS) project.
- Update .po files (Update from templates).
- Let translators work on the files.
- Download files from Pootle, convert them back, e.g.:
  ```
  po2prop -t ios/iosremote/iosremote/en.lproj/Localizable.strings
  Localizable.strings.po
  ios/iosremote/iosremote/fr.lproj/Localizable.strings
  ```

Questions