OpenOffice.org 1.1

Localizing the Help Files
Using the Mini Framework

Contents

Revision: August 2003

Owner: Frank Peters (fpe@sun.com)

Prerequisites

Operating System

The script uses Windows (2000) system commands.

Java Runtime Environment

You need Java JRE 1.3.1 or higher properly installed.

Perl

The script requires Perl 5 to be properly installed. It was tested using Perl 5.6.1. The `benchmark` package must be installed to use the timing functions.

Installation

1. Create a directory for help file assembly, e.g. `SO6_Help`
2. Extract the archive `L10NHelp.zip` into that directory preserving the subdirectory structure.
3. Add the subdirectory `{SO6_Help}/program` to your `path` variable.
4. Adjust the path to your perl interpreter in the first line of the perl script `createhelp.pl`:

   ```
   #!path/to/your/perl.exe
   ```

5. Adjust the following entries under `Set global variables` in the perl script `createhelp.pl`:

   ```
   #-----------------------------------------------------------#
   # Set global variables
   #-----------------------------------------------------------#
   # adjust these:
   $root_dir = 'C:\SO6_Help\\';
   $java_dir = 'C:\JavaSoft\JRE\1.3.1';
   $CLASSPATH = "$java_dir\\lib\\rt.jar;$java_dir\\lib\\tools.jar";
   ```

   Enter the directory where you extracted the zip file under `$root_dir`, e.g.

   ```
   $root_dir = 'C:\SO6_Help\\';
   ```

   Be sure to use **single quotes** (') and trailing **two backslashes**.

   Enter the directory where Java JRE is installed under `$java_dir`, e.g.

   ```
   $java_dir = 'C:\JavaSoft\JRE\1.3.1';
   ```

   Be sure to use **single quotes** (') and **no trailing backslash**.

   Enter the paths to `rt.jar` and `tools.jar` under `$CLASSPATH` separated by a semicolon, e.g.

   ```
   $CLASSPATH = "$java_dir\\lib\\rt.jar;$java_dir\\lib\\tools.jar";
   ```

   Be sure to use **double quotes** and **double backslashes**. Separate the paths using a semicolon and use the variable `$java_dir` defined above to designate the „root“ directory of your Java installation.

   You do not have to change the other entries at the top of the perl script if you use the given directory structure.

Usage

What's that directory structure for?

When you have extracted the zip file into a directory of your choice you will see a subdirectory tree with following branches:

```
{INSTALLDIR}
   etc
   input
      etc
      {lang}
         etc
         text
         picture
   output
      {lang}
   program
      classes
   stylesheets
      Indexing
      PreTransformation
   source
   installset
      {lang}
```

`etc`	This directory contains auxiliary files like the DTDs for the help documents which are not themselves needed for help production. You can use these DTDs to check the validity of the localized help files.
`input`	This directory contains the input files, i.e. the localized help files. The script takes the files from this directory.
`input\etc`	This directory contains auxiliary files needed for help production for *all languages* like the main transformation style sheet `main_transform.xsl` and the `STOPLIST` which contains words to be excluded from the indexing process.
`input\{lang}`	This directory has to be created by you and holds the branches with the xml help files `text\`, with the pictures and the directory containing language-dependent auxiliary files `etc\`.
`input\{lang}\etc`	This directory contains language specific auxiliary files, like the platform specific topic databases *.txt, the treeview files the configuration files, the custom stylesheet etc.
`output`	This directory contains subdirectories with help files which can be used for testing purposes.
`output\{lang}`	This directory is created when the script starts. Assuming the input files are placed in `input\newspeek`, the directory `output\newspeek` will be created.
`program`	This directory contains libraries and programs used by the script.
`program\classes`	This directory contains java classes used by the script.
`stylesheets\ Indexing`	This directory contains stylesheets used by the script to create the index files.

`stylesheets\ PreTransformation`	This directory contains stylesheets used used by the script to transform the xml files.
`source`	You can place the source tree in your favorite source language here for your reference. The directory structure found here can be used as a reference for the structure of the input directory tree.
`installset`	This directory contains subdirectories with the final help file output.
`installset\{lang}`	Here is what you probably want: the help archives that can be used for building a version.

What does the script do?

The perl script strongly depends on the correct directory structure (*cf.* above) and valid xml files in the help directories. It performs help generation in (1+)6 steps. Please also refer to the perl script itself for comments on particular procedures.

The output files are placed in language dependent directories (named like the input directory). In that way multiple languages can (in theory) be processed at a time (if your computer allows) by starting the script more than once with different parameters. The different languages should not interfere (untested).

Preparations

After setting the global variables – some of which have to be adjusted, see "*Installation*" – the target directories for the output files in `output` and `installset` are created.

Step 1: Update the topic databases

The platform dependent ASCII topic databases found in `{install}\input\{lang}\etc`, e. g. `swriter_WIN.txt` or `sbasic_UNIX.txt` are read and all titles contained therein are updated using the titles read from the localized files in the `{install}\input\{lang}\text` tree. The updated files are saved as, e. g. `swriter_new_win.txt`.

These freshly created files are converted to binary databases using the `txt2db.exe` program from the `{install}\program` directory. You end up with databases like `swriter.db` in the `{install}\output\{lang}` directory. These are required in the next steps.

Step 2: Update the treeview files

The existent files `*.tree` from `{install}\input\{lang}\etc` are read and all elements
`<topic id="some_id">Unlocalized string</topic>` and
`<node id="some_id" title="Unlocalized string">`
are updated using the ASCII databases updated in step 1 to show the new topic or node title.

Please note that some parts of these files have to be localized manually *(cf. Rules for localizing the files* on page 8).

The processed files *.tree are placed in the `{install}\output\{lang}` directory, e. g. `{install}\output\{lang}\swriter.tree`.

Step 3: Create the picture archive and database

The images contained in the picture branch of the input tree, e. g. `{install}`

`\input\icelandic\picture` are zipped into an archive file and the picture database `picture.txt` from the `{install}\input\{lang}\etc` directory is converted into a binary format using `txt2db.exe`.

The picture archive `picture.jar` and the picture database `picture.db` are placed in the `{install}\output\{lang}` directory.

Step 4: Create the indexes

The index files needed for the fulltext search are created using the java programs and libraries found in `{install}\program` and `{install}\program\classes` as well as XSL stylesheets found in `{install}\stylesheets\indexing`.

For successful completion of this step, Java 1.3.1 must be properly installed. This step takes the longest time, so please be patient.

The index files are placed in subdirectories `{application}.idx` in the `{install}\output\{lang}` directory, e. g. `{install}\output\icelandic\swriter.idx`, and contain a bunch of files required by the search engine.

Step 5: Transform and zip the xml files

The xml files in the text branch of the input tree are transformed using the java programs and libraries found in `{install}\program` and `{install}\program\classes` as well as XSL stylesheets found in `{install}\stylesheets\pretransformation`. This step also creates the application archives and places them into `{install}\output\{lang}\`, e. g. `{install}\output\icelandic\swriter.jar`.

Step 6: Finalize the process

All extra files are gathered together in `{install}\output\{language}\`, these include the configuration files `*.cfg`, which have to be localized manually, the stylesheet for help file display `custom.css` which may need adjustment depending on the language/font requirements and the error document`err.html` which also has to be localized by hand.

The required files are then compiled into archives and placed in the `{install}\installset\{language}\` directory. Form there they can be copied to the corresponding source tree. Please note, that regardless of what language you produce you will always get `*99.*` files in the `{install}\installset\{language}\` directory.

What you end up with

After the script has finished successfully you end up with a set of files in the `{install}\installset\{language}\` directory ready to be placed in the source tree.

You will also have a set of files and directories in the `{install}\output\{language}\` directory that can be copied to the `help\{language}` directory of a working StarOffice installation for testing purposes (remember to backup the original directory contents if you do so!). The new help should be available after restarting StarOffice (don't forget to quit the quick starter!).

What doesn't the script do?

The script is not particularly error-proof. It does not check for existing and proper installations of Java or other needed things like existing directories. It also overwrites any output file already present in `{install}\output` or `{install}\installset`.

If at first you don't succeed...

Nothing works as it should

...check for a proper Java installation (Java 1.3.1), JRE should be sufficient but I'm not a Java expert and I work with the JDK.

...check for proper Perl installation (Perl 5.6.1). The benchmark package should be installed (default). If you don't have access to the benchmark package or don't want that feature, simply remove all corresponding lines containing strings like "`...new Benchmark`", "`...timediff`", "`use Benchmark`".

...did you have adjusted the paths in the top of the perl script correctly?

...did you have adjusted the path to your perl interpreter on the first line correctly (might not always be necessary)?

Archiving fails

...do you have properly installed Java 1.3.1? The jar.exe in the {install}\program directory seems to require that version.

The indexing or transformation step fails

...did you check all xml help files for validity using the DTD provided in `{install}\etc`?

How to create the help

Prerequisites

Localize the data

You can find details for rules to localize the data in "*Rules for localizing the files*" on page 8.

STEP 1: Provide the input files

- **Create a directory `{lang}` for the localized files** as a subdirectory of `{install}\input`. The directory name is not important but we suggest that you use some kind of descriptive name, e. g. `{install}\input\`**`icelandic`**.
- **Create a directory `etc`** as a subdirectory of `{lang}`, e. g. `{install}\input\icelandic\`**`etc`**.
- **Copy all auxiliary files** to that subdirectory `etc`. These include `*.cfg`, `custom.css`, `err.html`, `*.tree`, `*.txt`. Please note, that not all of these have to be localized manually (*cf.* above)
- **Copy the localized help content tree** - beginning with the subdirectory `text` - into that subdirectory, e. g. `{install}\input\icelandic\`**`text`**.
- **Copy the localized help picture tree** - beginning with the subdirectory `picture` - into that subdirectory, e. g. `{install}\input\icelandic\`**`picture`**.

STEP 2: Run the Script

▷ **Start the Perl script** using the following syntax:

```
perl createhelp.pl [lang] [platform] [lang_iso] [country_iso]
[v]
```

where

`lang`:	the subdirectory `{lang}` (child of `{install}\input`) where the input tree (starting with the directory text) is found, e. g. `icelandic`. This can actually have any name you like. It only refers to the directory where the input files are found and the temporary output files are saved.
`platform`:	the target platform for which the help is to be created, select either `WIN` or `UNIX`. Use only these two values.
`lang_iso`:	the language code for the language to be produced according to ISO 639, e. g. `is` for `icelandic`. This parameter is necessary for creating the index using the correct locale.
`country_iso`:	the country code for the country to be produced according to ISO 3166, e. g. `IS` for `Iceland`. This parameter is necessary for creating the index using the correct locale.
v:	(optional) create slightly more verbose output

▷ **Wait for the script to finish**

Depending on the speed of the computer this will take up to about 1 hour.

STEP 3: Provide the files for the installation

You should end up with **eight zip files** in the directory `{install}\installset\{lang}` which can be transferred to the corresponding directory for building StarOffice. These are:

shared99.zip, sbasic99.zip, scalc99.zip, schart99.zip, sdraw99.zip, simpress99.zip, smath99.zip, swriter99.zip

Rules for localizing the files

The following files have to be localized manually:

▷ The help topic files **`*.xml`** in **`{install}\input\{lang}\text`** have to be localized completely. This includes the file title in the `<dc:subject>` tag of the document, because this is where the script gets the topic title for the databases. Please **do not use html entities** like `&apos;` in this tag, instead use UTF-8 encoded characters. Also update the language encoding element `<dc:language>en-US</dc:language>` to match the new language.After the localization process, the validity of the files can be tested using the DTD `{install}\etc\so-help.dtd`.

▷ The screenshots in **`{install}\input\{lang}\pictures`** have to be localized, if required. The localized files must have the **same name**, must be placed in the **same directory** and must have the same **width by height size** (the byte count doesn't matter). If the width by height size differs the images will most likely be displayed distorted, if the size is not adjusted in the corresponding help files which reference the image.

▷ The following parts of the treeview files **`{install}\input\{lang}`**

\etc*.tree must be adjusted manually:
The `title` attribute of the `<help_section>` tag.
The `title` attribute of any `<node>` tag with an id attribute starting with "`N`" or "`C`". like `<node id="C01" title="Unlocalized">`

- The application configuration files **input\{lang}\etc*.cfg**. You only have to adjust the `title` and the `language` attributes here.
- The error document **input\{lang}\etc\err.html.**

The following files are localized automatically:

- The help topic databases **{install}\input\{lang}\etc*.txt** are updated by the script. The original files are left untouches, the updated files are named like `swriter_new_win.txt`. If any of the latter exist when starting the script they will be overwritten.
- The topic specific information of the treeview files **{install}\input\{lang}\etc*.tree** are updated by the script. However, some parts of the files have to be localized by hand (*cf.* above).

The following files may need adjustment:

- The **{install}\input\{lang}\etc\custom.css** stylesheet may need adjustment if the localized language requires different fonts to be used. It may also be adjusted to satisfy national peculiarities.
- The **{install}\input\etc\STOPLIST** contains a list of words to be ignored in the creation of the fulltext search index. Unfortunately, this file is not language specific, so you will have to add all words to be ignored for all languages to be produced in that file. Initially, the file contains the most common english words to be ignored.

The following files need no localization or adjustment:

- All files above **{install}\input\{lang}** with the exception of **{install}\input\etc\STOPLIST** may be left untouched.
- The picture database **{install}\input\{lang}\etc\picture.txt** can be left untouched.