Mozilla Taiwan (Traditional Chinese Localization, 正體中文/臺灣本地化)

- 關於本站
- 網路資源
- **Wiki**
- 討論區
- 文件
- 下載

MozLCDB: Mozilla Locale Database

MozLCDB: What is it?

MozLCDB is a tool made for localization of various Mozilla (Gecko) based products. It is designed to resolve issues introduced since Mozilla Firefox 1.0 branch, and a replacement for MozillaTranslator.

The author and "Why a new tool?"

This is **Hung-Te Lin**, maintainer of **Traditional Chinese Mozilla Localization**, and the famous **"mozip"** (a tool to make Win32 localized builds).

I have been using MozillaTranslator for years (whether the official builds or my own tweaked versions) since Mozilla M14 (Milestone 14, not 1.4. The M series are older than Mozilla 0.6). During these years MT works O.K although more and more extra works have to be done because Mozilla locale structure changes all the time.

Since the landing of Mozilla Firefox 1.0PR, localizers of Mozilla* suffered from the huge changes in Mozilla locale structure and new CVS requirements. Unfortunately, MozillaTranslator does not work anymore. Although MT has released new versions to solve issues, after trying MT5.04 I think it's time to develop a new tool which is more robust or I have to manually do many extra work for all versions. With my experience in localization, I proposed what I need and made a new tool, named MozLCDB. This name is a little weird because the better name 'MLDB' is already taken by others. Hmmm. Maybe MLD?

I'm not an perl guru so this script is not perfect nor elegant now, but it helped me to generate a working Firefox 1.0PR locale without too much effort (in comparision to fight with MozillaTranslator).

Features of MozLCDB

MozLCDB is focused to Mozilla Localization. In comparsion to MozillaTranslator, you'll notice that:

- MozLCDB is just a perl script with command line only, no GUI.
  You can (and you have to) edit all stuff with your favorite editor.
- **Targeted for Multiple Product maintaince.**
  That is, you can really localize Mozilla, Firefox, Thunderbird, Netscape, and keep only **one instance** for same chrome node.
- Works for both CVS and downloaded ZIP language packs
- The stored database is more human readable/editable
- Single locale per database file. Does not support multiple locale (MT does).
  You have to use multiple database files if you need multiple locale.
- Easily modified if you understand Perl.

Requirements

You need these stuff:

- Perl 5.8 or later versions
- Your favorite editor (with the ability to open/save UTF-8, like **VIM** which is my own choice)
- (optional) Any ZIP archiver if you want to use ZIP/JAR languagepack or if you want to import MozillaTralsator glossary
- (optional) Python 2.0+ if you want to import MozillaTranslator glossary
- (optional) Knowledge of Perl if you got problems or want to modify it

Download

Get these files:

- Latest version: **Wed, 26 Oct 2005 03:28:39 CST** (hack on your own!)
- Releases:
  - **Version 0.4** (Stable, approved with Fx1.5b/moz1.8)
  - **Version 0.3** (Stable, approved with Fx1.1trunk,Tb1.1trunk)
  - **Version 0.2** (Stable, approved with FF1.0/TB0.8)
  - **Version 0.1** (Deprecated)

Background Knowledge

This section explains the basic idea about MozLCDB. It helps you to understand it more although you may

use it without knowing anything explained here. Yes, you can go to "Usage:" sections if you are in a hurry. However it is highly recommended to get things clear before you start.

Workflow

I'm trying to explain the workflow of MozLCDB here. Don't worry about the terms quoted, they will be explained in following subsections.

Basically, MozLCDB works in this manner: MozLCDB tries to 'import' from a 'prepared directory' (pulled from CVS or ZIP language pack), then save and merge into a '(glossary) database'. If any new locale string or updated ones are found, these entries will be written to an 'input (editing) table'. You can then edit the input table to do localization. After everything is done, you have to 'update' the '(glossary) database' with your 'input (editing) table'. Then you can 'export' the locales to another 'prepared directory'. Finally, the 'prepared directory' will contain all the localized stuff.

File Format and Default Filenames

MozLCDB tries to record everything in a text file named '**mozlcdb.txt**'. It is saved in perl Dump format, however it is still readble (if you know a little perl) and also called as the 'glossary' (due to MT) or 'database'. And there's another format of input that MozLCDB can read: the input (editing) table. It is more human readble and looks like DOS/WIN INI or GNU PO format. The default file name of this kind file is '**current.txt**'. When importing from and exporting to Mozilla locale directories, currently MozLCDB recognizes only DTD and ".properties" files.

Import, Export, and Update

import means to read locales from Mozilla files into the database.
export means to write locales from database into real Mozilla files.
update means to read locales from an input (editing table) into the database.

Directory Structure

Mozilla locale is stored in a hierarchical directory structure, also refered as chrome node. So you will see a very long path like "**locale/en-US/global/global.dtd**" in the directory you pulled from CVS or extracted from a language pack. MozLCDB mostly runs with the level just after where 'en-US' sits, so whether you're working with CVS or language pack, you must move directories (or move mozlcdb program and files) and assign correct ROOT path to fit thie rule. Then it can be called as a 'prepared directory'. (P.S: Actually, you can run in any level of directories. But you may get problems if you work in that way.)

ROOT

What's the ROOT? It's a PATH to help MozLCDB find corresponding locale path. For example, if my current directory is D:\Mozilla\LC (Win) or /home/moz/lc (UNIX) and my locale is in full path of D:\Mozilla\LC\locale\en-US\global\... or /home/moz/lc/locale/en-US/global/..., then you have to use ROOT as **locale/en-US**.

The procedure of 'Import'and Multiple Product

What really happened when you 'import'? MozLCDB will scan and walk through all files inside the 'prepared directory' assigned by ROOT. Currently, only ***.dtd** and ***.properties** will be processed. All other files will be ignored.

When one DTD or properties file was found, MozLCDB first reads its content for the 'key's and original locales. If a key was not found (never seen), it will be inserted into database as a 'new' entry. If a key was found, MozLCDB will determine what to do depending on the original (en-US) locale message. If it is exactly same as some version recorded in database, it is considered a 'same' entry with no new entries introduced. If no entries with same original could be found, MozLCDB will append one entry with timestamp into database. This is the magic to make multiple product possible. We keep all versions of original messages.

The procedure of 'Export'

What really happened when you 'export'? MozLCDB will scan and walk through all files inside the 'prepared directory' assigned by ROOT. Currently, only ***.dtd** and ***.properties** will be processed. All other files will be ignored.

When one DTD or properties file was found, MozLCDB first reads its content for the 'key's and original locales, then try to lookup database to decide what localized string to put in. If you export with '-x', MozLCDB assume you are working with a clean original copy exactly just what you have imported, so it will check for existence of keys and original locale message. However, you may want to export to a working (trial) copy. In that usage, -x will warn and stop because original message is filled with your localized strings. In this case you may use '-X' which will not check for keys.

Usage: Syntax Overview

- Run 'mozlcdb.pl -h' to see its syntax:

```
usage: mozlcdb.pl [-uUixXen] [PATH] ...

[-u]    : (default) update editing table (current.txt) to database
-U [files...]: import/update from editing table
-i ROOT: import files from ROOT(jar file or directory root)
-x ROOT: extract and update files in ROOT.
-X ROOT: extract and update files in ROOT, ignore errors.
-e      : generate latest version of full table from database
-n ROOT1 ROOT2: init with ROOT1 as en while ROOT2 as tr [CARE]
-c      : check database
-r aa-BB DispName: update "contents.rdf" locale entries
         from en-US to aa-BB with DispName
```

Usage: Import from MozillaTranslator

**If you don't need to import from MozillaTranslator, skip this section.
You must have python working before you start the procedure.**

I know this is weird: why python, not perl? Because this python script is modified by another python script I wrote years ago and I'm tired of making another perl version.

- First, unzip your glossary.zip which is in same directory of MozillaTranslator. You will find a "glossary.txt". Then run **mt2mldb.py** which is included in the program archive in same directory.

  ```
  python mt2mldb.py
  ```

- If no errors occurs, You will see a "mldb.out".
- Run the main program:

  ```
  perl mozlcdb.pl -U mldb.out
  ```

  Usually you'll get some warnings about "Not sync". Don't worry about that.
- If you have a "**mozlcdb.txt**" which is not 0 bytes in your directory now, congradulations, you have successfully imported MT glossary.
- Check mozlcdb.txt. Entries of chrome nodes should be look like "browser/metaData.dtd". If you see "locale/en-US/...." of "en-US/...", please use the replace feature of your editor to fix this error. This is caused by some different MozillaTranslator.

Usage: Import from an already localized language pack

If you have an already working language pack, say aa-BB, you must get its matching version of en-US package or importing may be useless.

- Unzip your language pack jar files (both for aa-BB and en-US).
- If region directory is included as BB and US, move all stuff inside BB/ and US/ to aa-BB/ and en-US/
- Move directories so that you can see en-US/ and aa-BB/ and your **mozlcdb.pl** in **same** directory.

-
  ```
  perl mozlcdb.pl -n en-US aa-BB
  ```
- If it stopped and giving you errors, try to identify what file stopped it. Try to resolve on your own if you know perl (feedback welcome), or mail me.
- If you have a "**mozlcdb.txt**" which is not 0 bytes in your directory now, congradulations, you have successfully imported MT glossary.

Usage: Import from CVS directory

Remember to backup your glossary database (mozlcdb.txt) before you import because you can rollback if anything goes wrong.

- First, you must pull from CVS directory. I'm not going to explain details here. **Check here.**
- Now make sure what's your ROOT. Look at the previous section of 'Background Knowledge'. If you have CVS pulled as 'mozilla/toolkit/locales' and 'mozilla/browser/locales' in your current directory, please use ROOT as "mozilla/toolkit/locales/en-US/chrome mozilla/browser/locales/en-US/chrome".
- Run this command:

  ```
  perl mozlcdb.pl -i ROOT
  ```

  Example:

  ```
  perl mozlcdb.pl -i mozilla/toolkit/locales/en-US/chrome mozilla/browser/locales/en-US/chrome
  ```

- You should see a 'current.txt' generated now.

Usage: Import from ZIP(JAR) language pack

Remember to backup your glossary database (mozlcdb.txt) before you import because you can rollback if anything goes wrong.

- Extract en-US.jar (and en-[win|unix|mac].jar, US.jar if you have them) to some place
- Now make sure what's your ROOT. Look at the previous section of 'Background Knowledge'. If you are working with Firefox 1.0PR, you'll see a directory named 'locale'. That's the ROOT.
- Run this command:

  ```
  perl mozlcdb.pl -i ROOT
  ```

  Example:

  ```
  perl mozlcdb.pl -i locale
  ```

- You should see a 'current.txt' generated now.

Usage: Working with input(editing) table

The input(editing) table is named '**current.txt**'. You should see it after you imported something (except importing from MozillaTranslator or already localized language pack).
The format is very simple. Take this for example:

-
  ```
  ;comment
  [browser/aboutDialog.dtd]

  id=aboutVersion
  en=version
  tr=PUT YOUR LOCALIZED STRING HERE.
  ```
- All lines stared with '**;**' are comments.
- **[ ]** marks the chrome node. i.e., the file path after ROOT. DO NOT CHANGE THIS.

- **id=** means the 'key' in file. DO NOT CHANGE THIS.
- **en=** Original en-US locale string. DO NOT CHANGE THIS.
- **tr=** Translated locale. This is really what you are going to work with.

  P.S: Updated and newly found strings will be ";tr=" commented at first. Remeber to remove the ";" mark when you localized one.
- **kp=** Just like keep in MT. kp=1 will ignore tr= when exporting.
- **cm=** Comment. Similiar to 'localization note' in MT. Does nothing.

Keep in mind that you should use UTF-8 file encoding.

Usage: Update database with an input table

When you finished editing, you have to update database. This is the default action if you give no arguments to MozLCDB.

-
  ```
  perl mozlcdb.pl
  (or) perl mozlcdb.pl -u
  ```

Usage: Export database to CVS

Before you export, you have to request for CVS account. Although you may already checked in your locale module, for example aa-BB, but we're not going to use it. Please review the section of 'Background Knowledge' for 'The procedure of Export'. You have to use the original dtd and properties files from en-US.

- Prepare a directory of original files. You may just copy from original en-US directory, like
  ```
  cp -R mozilla/toolkit/locales/en-US/chrome aa-BB-toolkit

  cp -R mozilla/browser/locales/en-US/chrome aa-BB-browser
  ```
  The commands (UNIX only) above will generate ROOT of "**aa-BB-toolkit aa-BB-browser**".
  If you use Windows, you can use Explorer to copy.

  For me, I'd like to copy original files into aa-BB CVS directory: (the aa-BB is exactly the CVS directory of locale aa-BB in the example, and the commands are witten with bash. You have to modify it to fit your own environment.)
  ```
  find aa-BB -name "*.dtd" -delete; find aa-BB -name "*.properties" -delete;
  for X in `find en-US -name "*.dtd"` ; do cp $X ${X/en-US/aa-BB} ; cvs add ${X/en-US/aa-BB}; done
  ```
- Run the export command.
  ```
  perl mozlcdb.pl -x ROOT
  ```
  For example:
  ```
  perl mozlcdb.pl -x aa-BB-toolkit aa-BB-browser
  ```
- After your first export, you may try (see the next subsection)the exported stuff. Then you can modify the input table and 'update' database again. And for the second time you want to 'export', things get easier. You don't have to prepare directory again. just use same ROOT and change -x to -X:
  ```
  perl mozlcdb.pl -X ROOT
  ```
- Finally, after you get a stable version, pleas make sure you have all *.dtd and *.properties added into CVS. And CVS commit!

Usage: Export database to a standalone language pack, or JAR files

Before you start, there's something you have to know. Currently we don't have a really good method for changing locale in Firefox (maybe we will never do), and the bug of "using extensions without corresponding locale leads to crashing" is still there (maybe will be solved in 1.0 release), so trying to provide language packs for end user is **really a mess**.

The better way is to export to CVS files and let mozilla.org build a official localized build. And if you want to make end user happy, due to the issues I mentioned above, here I described is kind of 'hacked' language pack. I'll export files, and make them with en-US chrome node path, and save to an 'en-US.jar' to replace the original en-US.jar.

This is **NOT GOOD**, but seems like making most end user happy. If you don't want to make such evil language packs, you have to go manually edit every *.rdf in directories, and prepare install.rdf or install.js.

OK... But if you are happy with evil packs, or **you're just trying your newly localized stuff** (this is mainly what I use this feature now for) , here's how to make it work:

- Prepare a directory of original files. You may extract from en-US.jar, and mostly you will have a directory named 'locale'. Now identify your ROOT. In the case above, it is just 'locale'.
- Export with:
  ```
  perl mozlcdb.pl -x ROOT
  ```
  For example,
  ```
  perl mozlcdb.pl -x locale
  ```
- If you are building a jar with new locale instead of replacing en-US, use
  ```
  perl mozlcdb.perl -r aa-BB "Language Name" ROOT
  ```

which will update all the 'contents.rdf' in ROOT. However, using a new locale name will require a mechanism of locale switching.

- Package your ROOT to a jar(ZIP) just like the layout of en-US.jar.
  For Windows, use your favorite archiver (WinRAR, or WinZIP, ... etc)
  For UNIX:

```
zip -r0 en-US.jar locale
```

- Now move the newly created jar to where your Firefox's en-US.jar is, replace it, and restart Firefox! You should be able to see your results, a localized Firefox now.
- After your first export, you may try the exported stuff. Then you can modify the input table and 'update' database again. And for the second time you want to 'export', things get easier. You don't have to prepare directory again. just use same ROOT and change -x to -X:

```
perl mozlcdb.pl -X ROOT
```

- If you want to create language pack or localized builds (again, this is NOT EASY and NOT SUGGESTED if you want to release files in this way. You should use CVS and let mozilla.org do this work. Care.... files made in this way cannot be called 'Official' due to new **L10N policy**), please see **Creating a Localized Installera** and **Localizing Firefox** .
  An easy hack is just to put the new en-US.jar into a Firefox installer, anyway.

Usage: Things left

Some commands are not explained in previous sections.

- -e can export whole keys in a database (used if your input table is broken).
- -U can update database with editing tables with less checking.
- -c can check database with some rules to help you find buggy phrases like a typo of %s and %S or "&nbsp;" without the trailing ';'.

Usage: Tips

- This program is still under development. Please backup your database (mozlcdb.txt) before doing any big changes, like -U or -i.
- Each time you have imported something, the 'current.txt' will be overwritten by a new one. So before you do an import, make a copy of last edited 'current.txt'.
- If you find any file with all-new locales (the information was displayed when you import/...etc), try to take a look at it. Maybe it's file just being renamed/moved. You may edit database to change/duplicate a copy of that to avoid doing all stuff again.
- A smarter rule of guessing new values is on plan.

Epilogue

This program is still under development. I used it sucessfully to make a working language pack (well, the en-US.jar) for Mozilla Firefox 1.0PR, Traditional Chinese. I'm not sure whether it is easy enough for most people or not, but it is designed for all what I need. So here I present it to all who needs to localize new Mozilla products but could not find any working tools to go on.

If you have any suggestions/problems, feel free to contact me. You can also use mozilla-l10n mailing list.

Copyright