Translation support Scripts documentation

Version 1.1 July 29th, 2010 by Simon Ginsburg, simon.ginsburg@bluewin.ch

1. Introduction to the translation scripts

This document covers a set of scripts that have been developed so simplify the task of translating a large JAVA project (JMRI). The author denies any implicit or explicit liability for the correctness of the code or the outcome of the functions. It's provided as is and it's the responsibility of any user to verify it's results. Please feel free to feedback any encountered problems and your suggestions back to me.

The functions follow a typical workflow for translating the *.properties files in any JAVA project that includes international language and character support:

1) Find the files that need translation and also all translation files already existing
2) Initialize the project by defining
    a. A set of reference files as where to start work
    b. Files that do not need to be translated
    c. Text items that do not need to be translated
3) Store all necessary files without altering the original code files
4) Provide a function to statistically analyze all files
    a. How many items exist in any file
    b. How many have already been translated
    c. How many text items are obsolete
    d. Store all files in dedicated separate directories so translation is parallel of JAVA code development
5) Allow to automatically import translated documents
6) Depending on the outcome of the above also an export function exists to copy back a finished translation file
7) Prepare a CVS command template to enter the finished files in the main repository for an international redistribution

2. Installation of the scripts

1) The scripts are written in Python 2.5 but are tested using Version 2.7
2) Check that either Python 2.5 or 2.7 has been installed on your machine and also that the package Tkinter is available for this version
3) They need to be copied into the JMRI project folder:
   $MAIN_JMRI_DIR$/java/Translatingscripts
4) When used they will create a project directory:
   $MAIN_JMRI_DIR$/java/translation
5) Within that directory a series of subdirectories will be created by demand:
   a. $MAIN_JMRI_DIR$/java/translation/Curr:
      Local copies of the translation files
   b. $MAIN_JMRI_DIR$/java/translation/Defaults
      In this directory all items will be stored that do not need a translation:
      i. The file "NonTranslatable.txt" contains all filenames when a file does not need to be translated at all. Currently these are the files:
         1. jmri-web-miniserver-Html
         2. jmri-web-miniserver-Services
         3. jmri-web-miniserver-servlet-fileservlet-FileServletPaths
         4. jmri-web-miniserver-servlet-fileservlet-FileServletTypes
         5. jmri-web-servlet-frameimage-JmriJFrameServlet
      ii. Any fullfilename (see below for the syntax) containing the keys which are excluded from translation.
   c. $MAIN_JMRI_DIR$/java/translation/Import:
      Here finished translation files can be copied in order for them to be automatically be included in the system
   d. $MAIN_JMRI_DIR$/java/translation/Ref:
      Local copies of the original files kept as a reference point
   e. $MAIN_JMRI_DIR$/java/translation/Status_$Date_of_Creation$:
      Here all statistics created by the scripts are stored by language:
      i. Complete:
         These translations are complete
      ii. Incomplete:
         These files contain either missing or obsolete items
      iii. New_Files:
         These files were newly created but still contain either missing or obsolete items
      iv. Non_Translatable:
         Copy of the files that are classified as non translatable
      v. Not_Existing:
         For these original files there's no translation file (yet) for the given language
      vi. CVSfile_$language_key$.txt:
         This is the command template to be used with the result of this automated statistic run

vii. Statisticfile_$language_key$.txt:
An overview statistic of all files separated into the categories:
   1. Non Translated files (with number of items)
   2. Not existing files (with number of items)
   3. New files (not existing as reference file)
   4. Incomplete files (with number of missing, obsolete, non translatable items)
   5. Complete files (with number of items)
   6. General Overview of number of items and number of translated items in all files
   7. Total number of original files and translated files

These two files are stored directly in the directory mentioned under e. They were created with the “Export” function:

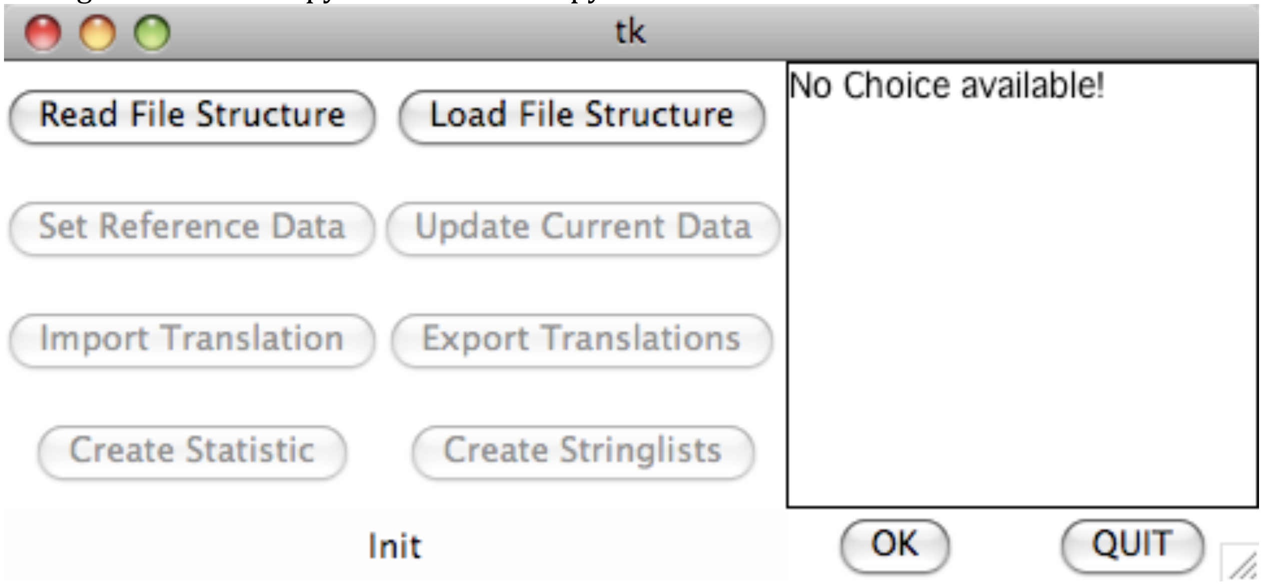
viii. CVSfile_$language_key$.txt:
This is the command template to be used with the result of eport run

ix. CVSfile_$language_key$.txt:
This is the logfile for the Export

6) Start main class using “python Testframes.py” from the command line

3. Using the scripts

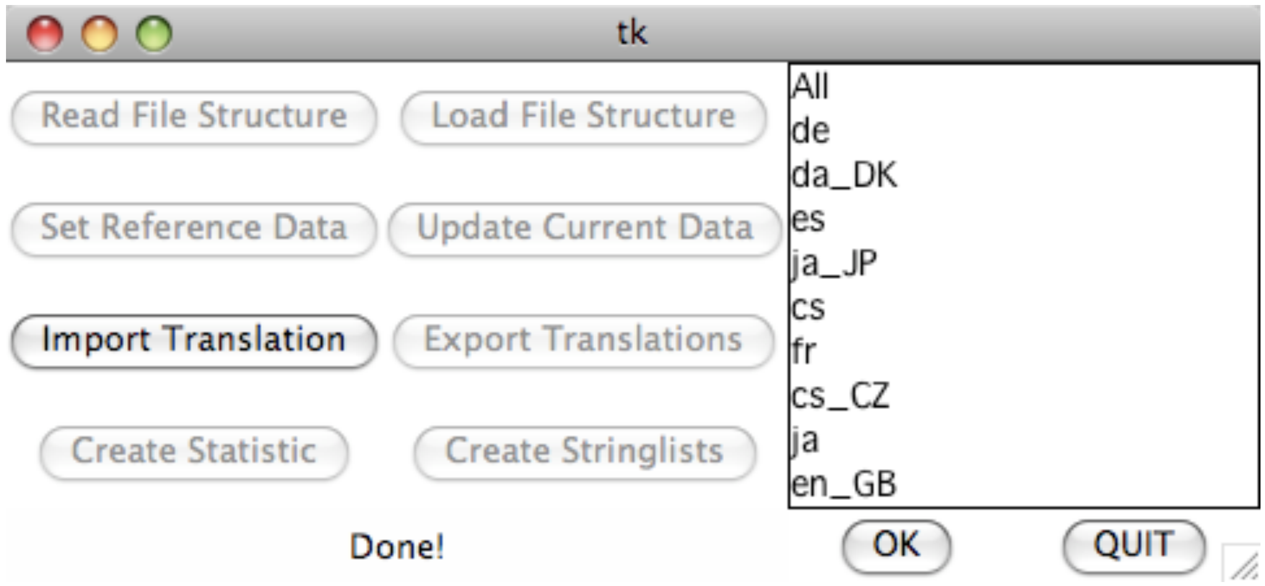
Calling the command “python Testframes.py” will create this GUI:



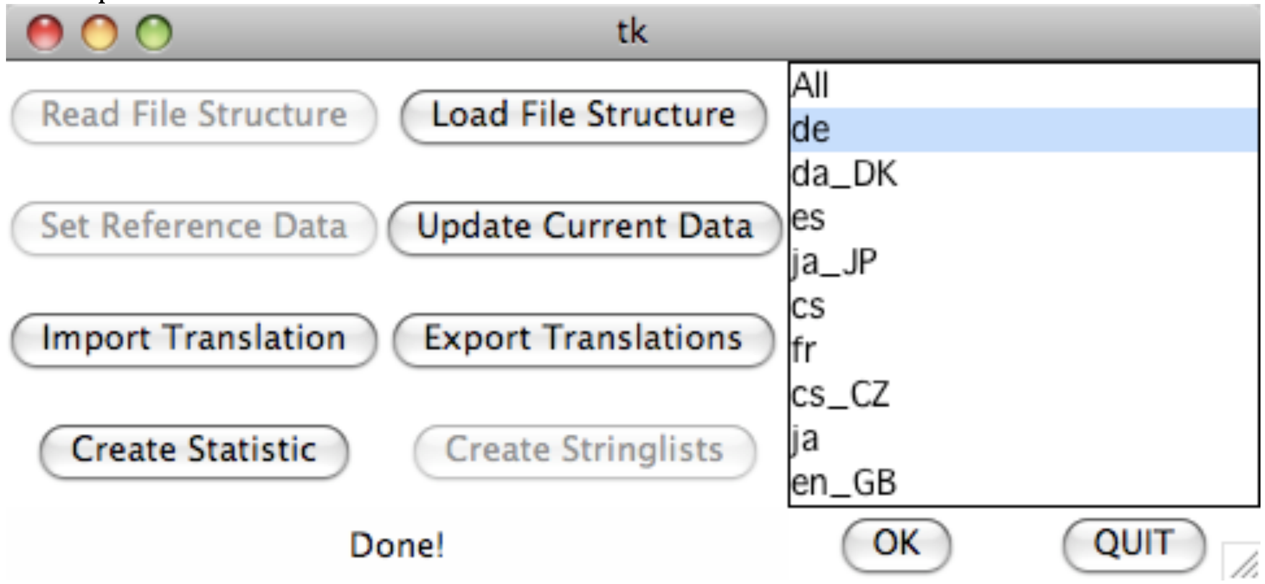
At left there are 8 commands which are partially grayed out and inactive if the prerequisites for a function call are not met. At right there will be a list of available languages. The status info at this moment is “Init”.

First time usage

When using the scripts for the first time first click on “Read File Structure”. This function will traverse the complete JAVA structure and keep a RAM copy of all file with the extension “.properties” which contain all internationalization info.
When this function has successfully terminated and the status info will be “Done!”.



At right all available languages will be displayed. The first entry is “All” and means that ALL translation information for all languages will be generated.
The language to be further used needs to be selected. Clicking on “OK” will now allow a few more options.



All files can be stored as reference using “Set Reference Data”. Also all files can be initialized in the *Default” Directory using “Create Stringlist”. This will vreate an empty file in the “Defaults” subdirectory. Entering one key ID per line will define them as non-translatable.
Using “Update Current Data” also the current set of files will be stored locally.
The Function “Create Statistic” will create a directory with the current date where depending of the selected language one or several subdirectories will contain all info for the current set of files.
The Function “Export Translations” will do exactly that. At the moment ONLY files that are considered “Complete” by the statistic will be copied back to the original file location and also receive the extension“.properties” again.

Continuous work

After the initial read out using the function “Read File Structure” will always copy the most current content of the JAVA project. Sometimes this is not desired during the translation task. Using “Load File Structure” only the local files will be copied into RAM.
In any of the “Status_$Date_of_Creation$” subdirectories you will find all files categorized depending on how much has already been translated into the desired language. It’s recommended to copy files into a “Work” directory during the creation of the translation along the “_TODO.txt” file that has a description of which lines still need to be worked out. Finished translation files can be copied into the “Import” directory. The Function “Import Translation” then can be used to import this file into the RAM. Selecting the “Update Current Data” will then copy its content into the “Curr” directory. If later the export function is run (using the “Export Translations” button) with any file that is complete in the local directory but not yet in the original file location the following activities will be performed:

- The file is copied into both the “Curr” and “Ref” directory.
- Also it’s copied back into the JAVA directory structure
- A CVS template entry is prepared in the “CVSfile_$language$.txt” in order to Check it into the main repository.
- A logfile will report any files that have been copied back.

Don’t forget to use “Update Current Data” in order to update the local directory after a new reimport.

Naming convention of the “Fullfile” name

In order to simplify life and still be able to have a bidirectional trace for the files in the JAVA project and my local copies I developed the Fullfile name. The syntax is fairly straight forward:
The base is the $MAIN_JMRI_DIR$/java path. Any subdirectory below will become part of the fullfile name. Subdirectories are separated by a “-“ and will be terminated by the filename without extension. After the language specific extension separated using “_” the extension “.txt” will be used so any text editor can recognize these files.
These are examples:

- “apps-ActionListBundle_de.txt” is the local copy of “$MAIN_JMRI_DIR$/java/apps/ActionListBundle_de.properties”
- “jmri-configurexml-SaveMenuBundle.txt” is the local copy of “$MAIN_JMRI_DIR$/java/jmri/configurexml/SaveMenuBundle.properties”

Error Handling

If a *.properties file contains irregular syntax, the import will NOT finish, but the status field will display the fullfilename and also the linenumber where an error occurred. Retry to import after you corrected the error!