

TranslationTable

1 License

Copyright 2005-2008 by Olaf Panz

<http://www.apache.org/licenses/LICENSE-2.0>

2 Overview

This small utility let you define Translations in OpenOffice 2. Usually the translation-tables of an application are maintained in Java- Resource- Bundles or property-files. This approach has some problems:

- A plain text file like a Java file or a property file are usually stored in ASCII-Format and although it is possible to store this files in a Unicode-Format, you usually will quickly find somebody in your project who did not know about Unicode and will store the file in ASCII (experience proofed). Another solution is the use of \uXXXX Tags and specify all Unicode character by their number. The resulting text will be perfectly readable in English, it will be readable in German, where you only have a few special characters, but it will be completely unreadable in Russian or Greek.
- The Translation of one piece of information in different languages is in separate files. In a longer project you will usually forget to add a translation in one file and forget to remove a second translation from a second file.
- You need Java-Development-Skills to maintain these tables; it's not a job for a Translator.
- The default Java-Mechanism uses String to lookup a translation. You will not know, if this key is correct unless you test your code. The compiler won't find it.
- A property file can only contain texts. By using a ResourceBundle-class, the management of language dependent Object's is possible, like Accelerators and Mnemonics.

The solution to this entire problem TranslationTable provides is to maintain translations in an OpenOffice file. TranslationTable create Java-Source-Code by using the ResourceBundle mechanism, encapsulated with a TranslationManager. Not String's but an enumeration is used to lookup Texts. With this approach, the compiler will find keys in use that do no longer exist in translation table.

Default Languages: You can define a default language for the whole table. This is the first column is the default language that is used if the TranslationManager is asked for an unhandled Language. Secondly you can define in Row 8 a default locale. This is resolved per line. E.g. you already have German and you want to translation for Swiss-German, you can use German as default for Swiss-German and translate only terms those are special. All not-translated terms are used from the default language.

3 Object Types

TranslationTable is able to translate different types of language dependent elements:

- Text: Plain text, handled as strings. Key's to Text get the prefix "TX_" for JDK 1.4 and are stored in the group TX for JDK 1.5 and higher.
- Tooltip: Same as text but with prefix "TT_" for JDK 1.4 resp. group TT for JDK 1.5 and higher.
- Mnemonic: Mnemonics are stored as characters. They get the prefix "MN_" for JDK 1.4 and are in the group "MN" for JDK 1.5. Can be specified with:
 - A single character, e.g. x
 - A Unicode constant, e.g. \u0123
- Accelerator: Accelerators are stored as KeyStroke object. They get the prefix "AC_" for JDK 1.4 and are in the group "AC" for JDK 1.5. Specify an accelerator:
 - One or any of this flags: SHIFT, CTRL, META, ALT combined with +
 - A single character, e.g. x
 - A Unicode constant, e.g. \u0123
 - A constant from class java.awt.event.KeyEvent with full name, e.g. VK_ENTER
 - A constant from class java.awt.event.KeyEvent with reduced name, e.g. ENTER
 - Examples: x, SHIFT+F4, CTRL+SHIFT+F4
- Eclipse-RCP: Generates eclipse-rcp property files to translate language dependent parts in plugin.xml. The property files are created over all translations of a code-generator-run. Files are written by default to current working directory. To active the RCP-translations-mechanism, a line needs to be added to MANIFEST.MF: "Bundle-Localization: plugin".

3.1 Client / Server Mode

The client mode handles one translation-table at the time. The table might change during the run of the program by setting `Locale.setDefault(...)`. In server mode, a translation-table is loaded on request and not removed. All translations tables are always available. Server mode needs more memory as client mode. A `clear()` method in server-mode cleans up all resources. The translation manager in server mode is thread save, in client mode it is not thread save.

4 Program Start

4.1 Commandline

Start the program with:

```
java -jar TranslationTable.jar <arguments>
```

```

Parameter:-h:      Print this help text.
                 -as <path> Specify a absolute target base path. This path is extended by package
structure. Default is ods-file's path + 'source/main/java'.
                 -rs <path> Specify a relative target base path. This path is relative to user.dir
                 -ae <path> Specify a absolute rcp plugin base path. Default is path of ods file.
                 -re <path> Specify a relative rcp plugin base path. This path is relative to
user.dir.
                 -axe <file> Export loaded tables in xml format. Specify target absolute.
                 -rxs <file> Export loaded tables in xml format. Specify target absolute.
                 -jdk <4/5>      Create source code for jdk 1.4 or 1.5. Default is 1.5.
                 -server        Create source for server environment, default is client.
                 -client        Create source for client environment, this is the default .
                 -rcp <RCP Root> Set the Eclipse-RCP base path needed to write i18n property files.
                 -ods <file>     Specify OpenOffice Calc (ODS) file for processing.
                 -xml <file>     Specify XML file for processing.
                 -codegen        Generate code with parameters specified before.
Parameters can be added in any order. A target path is used until the next is specified.

```

Example:

```
java -jar TranslationTable.jar -r src/main/java -jdk 14 a.ods -jdk 5 -s c:\temp b.ods
```

This call processes two files:

- A file <user.dir>/a.ods with JDK 1.4 compatibility. Created files are added to <user.dir>src/main/java/<package>
- A file <user.dir>/b.ods with JDK 1.5 compatibility. Created files are added to c:\temp/<package>

4.2 ANT-Task

The tool can also be started as an ant-task. At first you have to define the I18N-Task:

```
<taskdef name="i18n" classname="de.olafpanz.translationtable.I18NTask" />
```

Example of usage:

```
<i18n ODSFile="TestRCP.ods" environment="client" jdk="1.5"
rcpPluginRoot="." sourceRoot="src/main/java" />
```

Description of properties:

Name	Description
ODSFile	The source open-office-2.0 file.
environment	Define the environment to that the code is created. Valid are: "client" and "server".
jdk	Defines JDK to create code for. Valid are: "1.4", "4", "1.5", "5".
rcpPluginRoot	Root path of rcp plugin, is used to create RCP-Translations. This property is optional.
sourceRoot	Place where to generate source-code to. The individual package of a class is added to this path.