Manual

JLokalize 1.1

Tool for the internationalization (i18n) of Java applications.

Website

Download

Developer

Last updated: 14. November 2011 (ver 17)

Author: **Trilarion**

Content

1. Getting started

Windows

A double click on JLokalize.jar is the preferred way to start the program. In case this does not work, search the internet for tips of fixing the double click automatic start of jar files.

Linux

Either from a console with [java -jar JLokalize.jar]

or by double click within a file manager if the executable bit was set before

[chmod +x JLokalize.jar] and open-jdk or sun-java-jre is configured to start jar files automatically.

Mac OS

I haven't tested it, but either the command line: [java -jar JLokalize.jar] or double click should work if the jar launcher is configured to open jars.

JLokalize sets the Look and Feel of itself to match the present OS system, so you can expect to use it the way you use programs normally on your OS.

Troubleshooting

In case *JLokalize* does not start or closes unexpectedly with a crash, you can do the following:

- Make sure Java is installed and the jar file is started, i.e. other Java applications should be starting as usual.
- Send the console output, if there is any, to the author.
- Find the [JLokalize.log] file in the folder [user home/JLokalize.config] where [user home] is typically something like [C:\Users\Username\] on Windows and [/home/username/] on Linux. The path to [user home] is also printed as one of the first console outputs of *JLokalize*.
- Contact the author and describe the problem carefully.

Feedback, Requests, Suggestions

Can be sent to the author via email **trilarion@sourceforge.net**

2. Choose language of *JLokalize*

Upon the first start of *JLokalize*, the program will ask for its language. Later you can change the language at any time by clicking on the menu item [Help/Choose Language]. The changes will be instantaneous.

Add another language to JLokalize

Any help in porting *JLokalize* to other languages or fixing existing languages is very welcome. It is very simple to do so – *JLokalize* was localized by using itself. The languages of *JLokalize* are stored in [user home/JLokalize.config/lang] where [user home] depends on the OS system (see also Troubleshooting). Now you can open, modify and save them like any other project in *JLokalize*. New languages can be added or existing languages can be modified. Afterwards please send all modified files to the author so that they can be incorporated in a future version of *JLokalize*.

3. Project Level

The following chapter explains all possible actions on the project level. First some general words.

A project is a set of related languages, each of which consists of a list of keys and associated texts. They can have a hierarchy, meaning that you can typically define country-specify sub languages and again even variant-specific sub languages. Example: language [german] with code [de] specific for Switzerland with country code [CH] and again specific for some variant [Science] with a project with base name [Firefox] would result in the file name [Firefox_de_CH_Science.properties] which is conform with Java specifications.

The idea is now these files with a common base name form a tree and that each node of the tree contains pairs of key and value that are the same in all sub nodes of this node, i.e. we get more and more specific. When retrieving values later in the application by specifying a key and after loading one path to the root within the tree corresponding to the selected language, the search is that keys are first searched for in the most specific language file and then in its parents until we arrive at the root. Ideally somewhere there should be stored a value for every key.

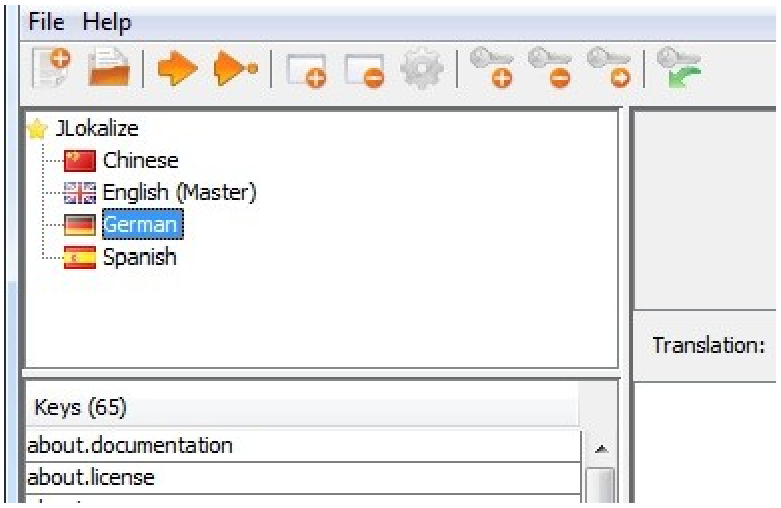


Figure 1: Language Tree

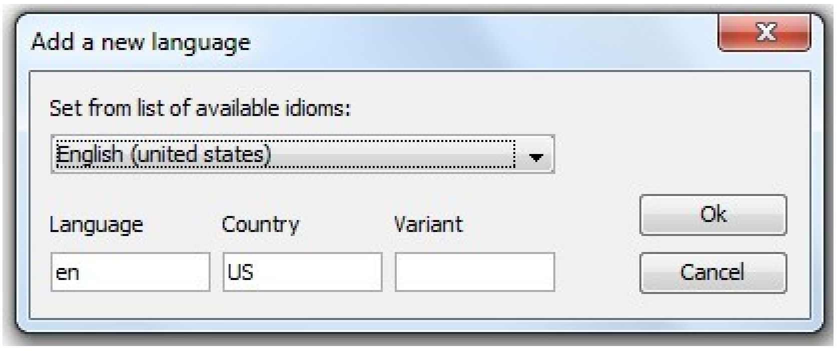
To facilitate keeping track of where the keys were found we incorporated statistics into the whole process. More about that in the Developer section.

Create new project

There is a menu item and a tool bar button for creating a new project. Simply specify the new base name. It can be changed later when using the [Save As] command. Upon creation a empty project with one node in the language tree, the root node corresponding to the base name, is created.

Add a new language

This function can be found as a tool bar button. It allows to add new nodes in the current languages tree, just specify a language, country and variant type (all except language can be left empty). A list of commonly chosen codes from the available Locales in Java is also created, but choosing from this list just copies codes in the input fields below. After confirmation a new node is created. If nodes in between would be missing, they are created too.


Add a new language

Set from list of available idioms: English (united states)

Language: en

Country: US

Variant:

Figure 2: Add a New Language

Remove a language

Just select a language from the language tree and press the corresponding button on the tool bar. Please be aware that the root node or languages that still have sub nodes cannot be removed but instead all keys are cleared. Also the associated language (properties) file is not removed from the file system (and will also not be removed during [Save]).

Open or Save a project

When you invoke the open project menu item or press the tool bar button a standard file open dialog is displayed. All files ending with [.properties] are shown. Just select one of these files and confirm the dialog, then all files with the same base name in the same directory will be loaded as a project. When the save project menu item is pressed, each language in the language tree will be saved as [.properties] files in the directory from where they were also loaded. Modified languages will result in modified files, added languages in added files, but removed languages will not be removed from disk. If you want to change the base name and/or the directory to store the project use the menu item [Save As] instead.

4. Keys Level

A language consists of an unordered list of pairs of key and content, both are represent by some text. The keys have to be unique, i.e. not two keys in one language can be the same. The idea is that every place in a program where normally some text would be displayed is instead be marked by a key and a separate module of the program is responsible for filling all places with the right content with respect to the key of this place and the currently selected language. Additionally we offer to store comments with the content each key, so more information can be given to the developer creating the content or for translators. However this comes at a price. Here it is that we stick to the <key, content> pairs and comments are stored under special keys, namely the normal key plus [.comment]. In other words, no normal key can end on [.comment] (*JLokalize* will refuse to add such keys), which should be only a minor disadvantage.

Add a new key

Simply click on the corresponding tool bar button. If a key is selected in the table on the left side, its name is already copied to the new key's text box, however you are free to choose any name, except you cannot add names that are already there.

Editing a key

When a key is selected in the key list on the left side, just edit the text areas on the right for value and comment to modify the entries of value and comment for this key. Changes will be saved once the selection in the keys table changes or the selection of the language in the language tree changes.

With the buttons ['Use' - English localization] above the text areas the content of the master language, if one is set, can be copied to the actual fields.

In the text areas spell checking might be available (see chapter Spell checking). Also the shortcuts [F2] and [F3] are reserved for moving to the next key in the keys list [F2] (implies saving of the actual content) or moving to the next (not yet here, only in master) key in the keys list [F3]. An alternative to use these function is via the tool bar buttons.

Remove a key

After removing a key by pressing the corresponding tool bar button, it still might be visible in the keys list. This will be the case if a master language was set and the key was also present there.

Rename a key

You can also rename a key. Internally this is handled by adding a new key, copying the content and followed by removing the old key. That means that the renaming action cannot be reversed as such, but each of the adding and removing actions it consisted of. Also you cannot rename a key to a new key that is already existing in the language.

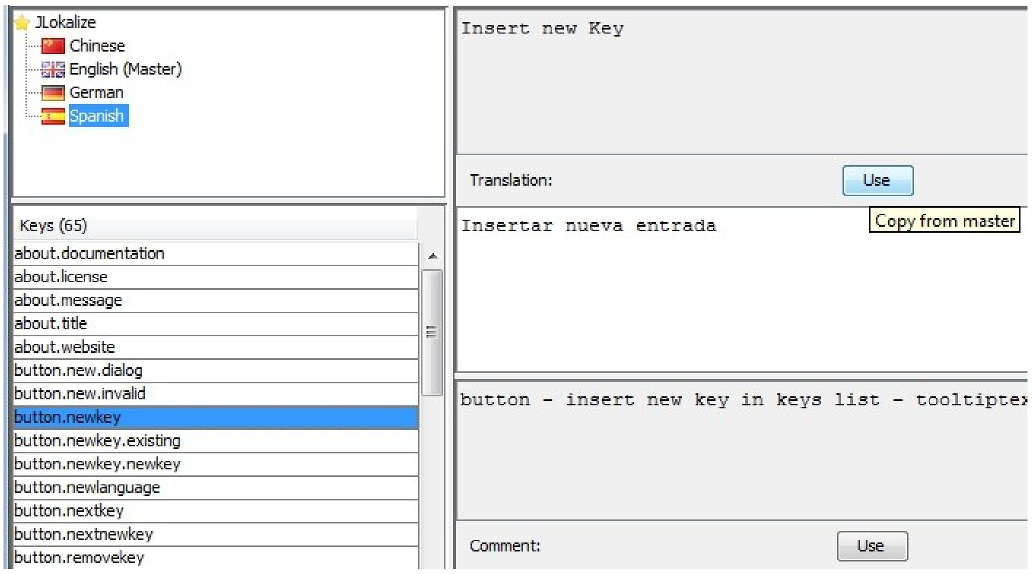
Reverse changes in a key

All keys that are changed during the run of the program are displayed with a small red bar next to the key's name in the keys list on the left side. Each of them can be reversed to the original state, i.e. the state after loading the project or after the last save of the project, by pressing the corresponding button in the tool bar. This action cannot be undone, the changes are lost and the original state is restored.

The key might be removed completely from the list during this operation. Saving the project will result in clearing all modification information, so only actual changes since the last saving operation can be restored.

Set a language as master language

Any language on the first level of the language tree or the root language can be set as master language. The sole purpose is to provide a template after which translation of the actual selected language can be completed. The key's table on the right side will contain all keys of the actual language and the master language, the status (existing in both, only in master, only here) will be color coded and displayed in the tool tips. The content of the master keys will be displayed in the read-only text areas on the right side.



```
Insert new Key
```

```
Insertar nueva entrada
```

```
button - insert new key in keys list - tooltipte
```

Keys (65)
about.documentation
about.license
about.message
about.title
about.website
button.new.dialog
button.new.invalid
button.newkey
button.newkey.existing
button.newkey.newkey
button.newlanguage
button.nextkey
button.nextnewkey
button.removekey

Figure 3: Master Language content is displayed and can be copied.

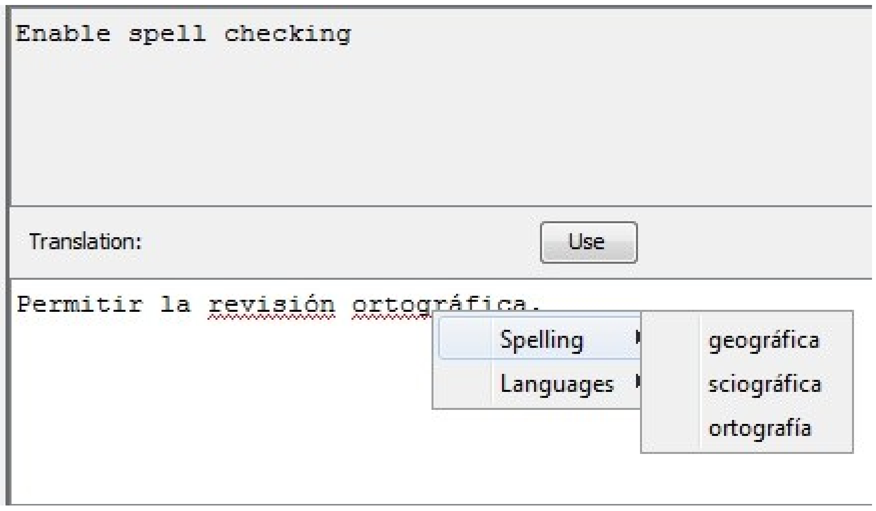
Keys Level

Go to next (new) key

This function is especially helpful when completing a translation, i.e. translating all the new keys in a master language of your choice, that weren't yet translated to the actual selected language. It is equivalent to visit all keys in the keys table which are shown in blue. Alternatively you can go to the next key which will just go to the key below. Shortcuts [F2], [F3] can be comfortably used when in the text areas.

5. Spell checking

This program used the [**JOrtho**] library to provide simple spell checking abilities. Spell checking can be enabled in the [Options] menu item, however only if the dictionaries [JLokalize-Dictionaries.zip] were downloaded and unzipped to the same folder as [JLokalize.jar]. The following dictionaries are available: [de, en, es, fr, it, nl, pl, ru]. Then a context menu [right click] will be available in the text areas where you can select the available language for spell checking, as well the available localization of the context menu itself. The integration with *JLokalize* is not that good, so they have to be chosen manually and changed manually for each newly selected language again. Unrecognized words are underlined in red.



Enable spell checking

Translation:

Permitir la revisión ortográfica.

Figure 4: Spell check context menu

6. Developer info

The whole concept of *JLokalize* is completely compatible with Java Properties and ResourceBundle classes, even uses Properties internally (although it is a flawed class), so integration with existing projects should be very smooth. Just have a looks at classes [Property] and [PropertyWithStatistics] in my library [**JTools**] (**Browse** source code). So either use the Java implementations of Property and ResourceBundle classes or your own compatible solution or I would be delighted if my classes would find use also in other applications than *JLokalize*.

Here are some hints for using [Property] and [PropertyWithStatistics]: Each [Property] object can have a parent, a static chain loading method working on language code, country code and variant (to load all of them) is available. Additionally it uses Resources for specifying file locations (my own implementation, allows to read from within archives). And some convenience methods are present for storing numbers instead of text. The main idea is that the get method searches for a given key and either returns the content or delegates to the parent if the key is not present. In [PropertyWithStatistics] additionally a statistics file is loaded and written. There we keep track of which key is wanted and how often. The main purpose is to find out, which keys in the language database are outdated and which keys still have to be added.