Translator: A Package for Internationalization for Java-based Applications and GUIs

Guido Rößling
Dept. of Computer Science
Darmstadt U of Technology
64289 Darmstadt, Germany
roessling@acm.org

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education—*Computer-assisted instruction*

General Terms

Management

Keywords

Internationalization, GUI programming, Java

1. INTERNATIONALIZATION: ISSUES AND SOLUTIONS

Internationalization targets applications and graphical user interfaces (GUIs) that can be translated into different languages. It is an important issue for today's global economy - and therefore should concern all programmers, even native English speakers.

In Europe, the need for internationalization is quickly apparent, as most neighbouring countries use a different language. Think, for example, of Portugal, Spain, France, Germany, and Poland, to take a broad sweep from southwestern into eastern Europe. Some languages are also related but differ significantly—for example, the variants of German spoken in Germany, Austria, and Switzerland.

Internationalization is one of the challenges waiting for our students in the "real world". Most "obvious" approaches are inappropriate. Using lots of `if` clauses slows down the code and can easily lead to other problems due to a "dangling else". Branching the sources into the "English" and "French" (and ...) version makes sensible code maintenance and bug fixing close to impossible.

Typical issues for internationalization include the following:

- Translate a given fixed message into a different language.
- Format numbers "appropriately". For example, the US notation `1,000.73` would be `1.000,73` in Germany.
- Texts should be able to have parameters, e.g. for messages like "You have *n* new messages". If multiple parameters are present, their ordering may differ between languages.
- GUI elements should also be translatable. This concerns the labels for menus, menu items, buttons, mnemonics, tool tips, and in some cases, even the associated icon.

ITiCSE'06, June 26–28, 2006, Bologna, Italy.
ACM 1-59593-055-8/06/0006.

Java supports the first three listed issues with a set of the classes in the packages *java.util* and *java.text*. Programmers can use these classes to assemble their translations with relative ease. However, the last internationalization concern - the translation of GUI elements - is not addressed by the Java API.

We have designed a small Java package called *translator*, which addresses all four internationalization issues. The support for the first three problems—translating text, formating numbers or dates, and handling parameters—is accomplished with a single method invocation. A special GUI generator can create a large set of Java Swing GUI elements with one method invocation each, including *JButton, JCheckBox, JLabel, JList, JMenu, JMenuItem*, and translatable tabs for a *JTabbedPane*.

All properties for the elements—such as its label, mnemonic, icon, and tool tip—are placed in a locale-specific resource file, which is loaded once the target locale is selected. For a base name *resources*, the files *resources.en_US* and *resources.es_SP* will store the US-english and spanish resources.

The generation takes care of all additional method invocations, such as setting the label, mnemonic, icon, and tool tip, and may optionally also install an appropriate listener. Direct method invocations for action elements are also supported, and require the additional passing of the object on which the method is to be invoked.

The following excerpt shows the resources for a JButton including label, icon, mnemnoic, and tool tip.

```
ok.label=OK
ok.icon=ok.jpg
ok.mnemonic=o
ok.toolTipText=Press me to close the window.
```

Assuming that the generator has been initialized before, the user needs only one command to turn this specification into a full button:
JButton aButton = generator.generateJButton("ok");

All GUI elements created in this way are stored in a look-up table. The locale can be changed on the fly, causing the system to look for the appropriate resource file. If such a resource is available, the API will automatically translate all elements—including their labels, mnemonics, tool tips, and, where appropriate, even icons.

Using the package reduces the workload on both educators and students by allowing "detail obfuscation" where appropriate. It also allows educators and students to explore internationalization by focusing on the content, not on the technical issues. First tests also indicate that it makes generating GUIs easier, as the package takes care of several intricate Swing details and method calls.

The *translator* package is currently not available online, but we are happy to share it with any interested educator or student.