Technology Evaluation Centers

Taking Multilingual Support to the Next Level

Ketharaman Swaminathan - August 17, 2009

Traditionally, multilingual software applications and Web sites allowed users to select a language from a list containing multiple languages at the set-up and log-in stage. The language was then installed and displayed on all screens, menus, and help text. That sufficed when the application's use was confined to a single language at a time.

With the increasing globalization of the twenty-first century, co-workers who are spread across many countries in multinational corporations need to work cohesively as one unit. The traditional level of multilingual support no longer suffices. Multinational corporations need the next level of multilingual functionality from their software applications to boost productivity and, in some cases, even meet compliance with local regulations.

Product managers in product companies and solution architects of custom-developed applications can enhance the marketability of their software applications by creating a roadmap for incorporating the sophisticated multilingual requirements that such multinational corporations need.

Multilingual Support in Software Products

Let's take the case of an analysis and research application for European *asset management companies* (AMCs) spread across France, Germany, Italy, Spain, the United Kingdom (UK), and other European countries.

It would be typical for a branch of one such AMC in one country to have employees belonging to another (e.g., the Paris office would have a few German employees). As a result, not only would the product be expected to support multiple languages (e.g., French and German), but it would need the capability to permit an employee to choose the language (e.g., German) instead of installing the French-language version by default.

In many AMCs, investment recommendations for funds traded in one country would need the approval of the *chief investment officer* (CIO), located at the headquarters in another country, before they could be presented to customers. This means that after logging into the application in the German language, the asset manager in the AMC's Frankfurt office might need to switch over to the English language later while reviewing the data with his UK-based CIO prior to obtaining his or her approval. Once the approval is obtained, it might be necessary to recast all data, reports, and charts into the French language before presenting them to a prospective customer in Paris as part of an investment recommendation. Therefore, the application should not restrict users to the language chosen at the time of logging in, but let them toggle between multiple languages "on the fly."

Apart from the multilingual screens, menus, and help features, the application will have to cater to data entry, querying, and reporting needs in the different supported languages. This means support for various accents such as the German *umlaut*, French *grave*, *acute*, *cedilla*, circumflex, and the Spanish *tilde*. In addition, sensitivity to different date formats such as *15 June 2006* in the UK versus *15. Juni 2006* in Germany; and decimal separators such as a *comma* in the UK versus a *period* in Continental Europe (e.g., *1,000 versus 1.000*) are respectively prevalent in different European languages.

Now, when a London-based asset manager queries for all French funds containing a certain letter in their name, he or she would expect the application to return funds containing not just that letter by itself but its various applicable accents like *acute* (e.g., "ć"), *grave* (e.g., à), *cedilla* (e.g., ç), and *circumflex* (e.g., ĉ). This means that the application will need to support multilingual query literals, characters, and messages.

Design and Technology for the Software Product Manager

While different technologies are available to implement the next level of multilingual support in software applications, we examine one design approach based on **.NET**'s technology elements.

Satellite assemblies can be used to provide multilingual support for menus since they contain localized resources. Microsoft Developer Network defines a satellite assembly as follows:

> A .Net framework assembly containing resources specific to a given language. Using satellite assemblies, [the developer] can place the resources for different languages in different assemblies, and the correct assembly is loaded into memory when the user opts to use that specific language.

Using satellite assemblies, designers can place resources for different languages in different assemblies. Depending upon the language selected by the user, the corresponding assembly will get automatically loaded into memory. The application will need to incorporate individual satellite assemblies for each specific *culture* (the combination of a particular language with a particular country). For example, the culture "fr-FR" refers to the French language as used in France, whereas "fr-CA" refers to the French language used in Canada. These satellite assemblies can be placed in a specific location and loaded by the parent framework based on the culture setting of the software.

Support for multilingual messages can be provided by storing the culture-specific user messages in separate culture-specific message files (one file per culture). Message files appropriate to the specific culture setting of the software will be used to retrieve and display messages to the user.

By leveraging the *locales* facility, developers can build support for different date formats and decimal separators. Microsoft Developer Network defines a locale as follows:

> A collection of rules and data specific to a language and a geographic area. Locales include information on sorting rules, date and time formatting, numeric and monetary conventions, and character classification.

While the user-selected default locale would be applied by default to all information presented to the user, the user can also select any other locale from a list of pre-defined locales to view the same information in another format.

Multilingual support for special characters in queries can be incorporated by maintaining separate culture-specific mapping files (one file for each culture). A mapping file maps a character in the chosen culture to the special accent characters in the other cultures supported by the software. The application would scan the mapping file corresponding to the language setting (culture) of the software and construct additional search phrases for all other mapped cultures.

By consciously designing the application so that it enables the user to select the language culture at the time of installation, one can prevent a "forced install" in the local language culture derived from the PC's regional settings. By designing the application to install and load all—and not just the user-selected culture files, designers bestow it with the capability of being re-started in another culture.

By leveraging the facility available to set the *user interface* (UI) culture property in report controls, reports generated in the native locale can be toggled into another locale. As a result, the application will be able to support recasting of data, reports and charts generated in one language into another.

In this manner, product managers and designers can combine sophisticated features available in the .NET technology stack with an innovative design approach in order to deliver the next level of multilingual support in a software product, with the overall goal of improving usability and boosting productivity.

Multilingual Support in Custom-developed Applications

Apart from usability and productivity, compliance considerations will increasingly raise the bar on the level of multilingual support demanded from software applications. This trend has already been noticed during implementations of the *Single Euro Payments Area* (SEPA) regulation among banks and corporations operating in different countries in Europe.

At one such pan-European SEPA implementation, the system landscape was comprised of a payments product processor, a trans-European liquidity manager, a *Society for Worldwide Interbank Financial Telecommunication* (SWIFT) gateway, and a sanctions-screening product, which was meant to block cross-border fund transfers suspected of having links to terrorism, money laundering, and other nefarious activities.

Since SEPA intrinsically involves cross-border fund transfers, all transactions had to be screened by the sanctions

screening application, which would approve a transaction or block it depending upon the outcome of verifications being implemented against various hot lists. Having been built with a fairly sophisticated level of multilingual support this application supported the extended character set of national languages of all the European countries complying with SEPA. Therefore, it was able to handle German umlauts and other accent symbols in various European languages. For example, it could treat the German name "Müller" differently from "Muller", which is derived simplistically and incorrectly by ignoring the umlaut symbol.

Now, since the existing legacy payment product processor did not support an extended character set, there was a strong potential for mismatch between the various systems included in the system landscape whenever a term contained umlauts or other special characters (`, ´, °, ˜, etc.) commonly found in non-English European languages.

What Options Does the Solution Architect Have?

The simplistic approach of dropping all special characters (e.g., by treating "Müller" as "Muller") will lead to: (a) passing of a transaction that is meant to be blocked; or (b) blocking of a genuine transaction. In the first case, the company risks non-compliance with regulations relatin to terrorism and money laundering. By introducing a "false positive," the second case leads to the risks of customer dissatisfaction and higher repair costs when the error is realized later and has to be manually fixed. In other words, a solution architect cannot afford to take a simplistic approach.

On the other hand, by incorporating extended character sets into all applications, the solution architect can help the company avoid these risks altogether since there won't be any mismatch between the various applications in the SEPA system landscape in the event that fund transfer instructions contain special characters.

But if this change turns out to be too far-reaching to implement immediately, the solution architect could, in the short term, tweak all applications to use generally-accepted English equivalents for non-English accents. For example, "Mu**e**ller" for "Müller" in the above example is derived by substituting the umlaut symbol with an "e" as per standard convention instead of ignoring the umlaut symbol.

In the medium term, the solution architect could evangelize the adoption of latest standards and technology stacks that support extended character sets and concurrent multilingual support.

About the Author

Ketharaman Swaminathan is founder and CEO of **GTM360 Marketing Solutions Private Limited**, a business-to-business technology marketing solutions company that helps high-tech companies maximize value from their ideas, products, and capabilities.

Ketharaman has a unique IT products and services background acquired from a career spanning more than two decades at **Fujitsu**, **Oracle**, **Ramco**, **Wipro**, and other leading technology companies in India and abroad.

Throughout his career, Ketharaman has developed frameworks related to differentiators, marketability, price forecasting, usage-based pricing, and other elements of go-to-market. He has also authored several articles and blog posts on go-to-market, Web 2.0, software usability, *return on investment* (ROI) of *enterprise resource planning* (ERP), and ERP customization.

Ketharaman graduated from Indian Institute of Technology Bombay with a bachelor's degree in technology. He subsequently earned a master's degree in marketing management from Jamnalal Bajaj Institute of Management Studies Bombay. He is based in Pune (India), and can be reached at s.ketharaman@gtm360.com.

7596.609