Designing your MIDlet for JavaVerified

Overview

The **JavaVerified** program is an initiative by the wireless-industry leaders Sun Microsystems Inc., Sony Ericsson, Motorola, Nokia and Siemens to provide a single and uniform environment for the testing and certification of MIDlets.

The testing program consists of three parts:

- Free, automated pre-testing
- UTC Test (Unified Testing Criteria)
- Digital signing of the MIDlet

Although the tests don't cover all aspects of a well-working MIDlet, the testing is quite extensive and is divided into 10 areas:

- General MIDP Application Launch
- General MIDP User Interface
- General MIDP Functionality
- General MIDP Security
- General MIDP Network
- General MIDP Localization
- JTWI MIDP 2.0 User Interface
- JTWI MIDP 2.0 Operation
- JTWI Wireless Messaging API 1.1 (Wireless Messaging API)
- JTWI Bluetooth

A good knowledge of the testing criteria, and keeping these in mind during the design phase of the project, will almost certainly help in passing the tests.

A very good walk-through of the testing criteria, together with design and coding best practices for passing the tests, can be found in the JavaOne 2004 BOF presentation "JavaVerified Program – How do I make sure to pass the test and get my MIDlet signed?" by Hanz Häger, Sony Ericsson's representative in the J2ME Executive Committee.

Hint 1

For example, the presentation explains that for general MIDP localization, you should try to use appropriate standard classes for handling location-specific information such as the date and time. Lacking standard classes for some cases, you'll have to implement them yourself for different locations. For example, you'll often have to write your own classes to handle numbers and currency formatting.

Hint 2

When speaking about localization, a good, general programming practice is to extract location-specific data from your MIDlet's Java code and put it in resource files instead. This is especially true if you plan to use Java Verified and have multiple-language versions of your MIDlet. You should then store all location-specific information, such as help texts, in resource files, one or more for each country. This way you can distribute exactly the same Java code for all different language versions and the same code will be executed independent of language.

Besides the obvious benefits of having one common Java code structure to manage all MIDlet versions, you might also save some money when it comes to Java Verified testing. This is understandable if you look at the alternative to using resource files: putting your location-specific data in your Java classes. If you do this, you'll have to go through, and pay for, a UTC test for each language version, as it's impossible to prove that two versions only differ in the language used. On the other hand, by extracting your location-specific data to external resource files, you'll only have one MIDlet to be tested – possibly with the additional check of all the different resource files as valid.