Java Is Jumpin', This Time for Real

Alexander Wolfe, Science Writer

There's perception, and then there's reality. Even though the frenzied hype over Java has died down since the Internet bubble burst, Java is becoming hugely popular in the wireless space. Several events highlight its emergence. Most recently, in December, Texas Instruments opened a research operation in France to focus on the integration of Java apps into the next generation of wireless devices.¹

And throughout 2003, Java originator Sun Microsystems made some serious moves aimed at expanding the technology's horizons. Meanwhile, in the developer community, open source efforts and the applications needs of handset manufacturers are spurring grassroots projects.

Indeed, after several years of Java doldrums, 2003 marked a year where Sun gained some serious technological traction for Java on mobile phones and PDAs (personal digital assistants). For developers, this translates into the availability of heavier-duty tools for building Java-based wireless apps.

The premiere Java tool for developing apps for downsized devices is J2ME (Java 2 Platform Micro Edition). J2ME is billed as a set of tools and an optimized, open Java runtime environment, complete with the emulation environments, documentation, and examples needed to develop Java apps targeted at consumer wireless devices, ranging from phones and PDAs to smart cards and set-top boxes.

Developers can download the version 2.1 beta release of the toolkit free from Sun at <http://java.sun.com/developer/earlyAccess/j2mewtoolkit/index.html>. J2ME arrives as a wrapper filled with an alphabet soup of profiles and APIs (application programming interfaces). The package contains tools such as MIDP 2.0 (Mobile Information Device Profile), a new specification/API that supports the core functions required by mobile applications, including the user interface, network connectivity, and local storage. The second key profile in J2ME is the CLDC (Connected Limited Device Configuration), which handles the intermittent connectivity requirements of cellphones. Also in the J2ME mix are profiles such as J2ME Web services, Mobile Media APIs version 1.1, and the Wireless Messaging API.

The J2ME runtime

ENVIRONMENT SEEDS
CELLPHONE APPS.

An odd twist—or perhaps a recognition of reality on the part of Sun—is the fact that the officially supported platforms on which J2ME runs are Windows XP and Windows 2000. J2ME will run on Linux and Sun's own Solaris, but it's not “officially supported” on those Unix clones. (This bifurcates the wireless development/target platform equation; the Symbian operating system, used by Nokia, is the most popular handset operating system, while Microsoft's Smartphone systems software² is a market also-ran. So, paradoxically, cellphone developers may end up creating code for Symbian handset targets on Microsoft PCs.)

Practically speaking, the key to writing apps that are portable across heterogeneous handsets is to write your Java code in the form of a *MIDlet*. That's the name applied to mobile Java applications that are created with the J2ME toolkit and that conform to the MIDP 2.0 profile. A MIDlet will run on any device that implements MIDP; this goes beyond just cellphones to encompass handheld computers and hybrids. Importantly, unlike standard Symbian phone apps, MIDlets are “sandboxed”—that is, their memory usage is isolated to prevent contamination by malicious virus code.

(In the real world, portability isn't yet quite as universal as promised. The idiosyncracies of the not-yet-standardized, resource-constrained cellphone platform—not to mention the variations in user interfaces tailored for tiny screens—mean tweaking is usually required. Nevertheless, copying your MIDlet files to another platform is a starting point that sets you more than halfway on the road to completing any port.)

The MIDlet emulator supplied with the toolkit is perhaps its centerpiece and is the key to testing programs before they're deployed to the target handset. (Typically, third parties such as Nokia offer separate utilities to handle downloading the executable to their respective handsets.) I've found that coding those apps—which is pretty much a pedestrian process of loading up your module with calls and passing parameters to Java classes that access required functionality—entails what's largely

a trial-and-error learning curve. To speed skills acquisition, there are several useful app notes that explain how to write your first app and run it in an emulator. Sun offers one at <http://wireless.java.sun.com/midp/articles/wtoolkit/>. Nokia provides an excellent overview of Java on mobile devices, along with an example and source code of how to create a MIDlet application, at <http://ncsp.forum.nokia.com/support/?body=detail&aid=183>.

COLLABORATIVE COMMUNITIES

In the broader software world, wireless Java is evolving very much as a grass-roots political campaign. Consider the recent formation of the Java.net community, a loose, collaborative collection of mostly open source projects (see <https://www.dev.java.net/>). Programmers of all stripes have been signing on and posting requests for people to join their proposed efforts.

The nexus of wireless-tools interest at Java.net is in the "Java for Communications" sub-community (<https://communications.dev.java.net/>). This site touts the availability of Java components for the construction of VoIP (voice over IP) apps, IM (instant messaging) libraries, and other mobile/wireless utilities.

The chief project in that orbit is CellphoneJava (<https://cellphonejava.dev.java.net/>), a nifty collection of applets and components. Also of interest is SIP (session initiation protocol) Communicator, a client with media and IM capabilities that supports audio/video sessions over IPv4 and IPv6 (Internet Protocol versions 4 and 6).

On the corporate front, Sun and Oracle have banded together in a bid to co-op the community banner.³ They've formed the Java Tools Community,⁴ a group that seeks to enlist tool vendors to create an alternative framework to Microsoft's .NET⁵ and Visual Studio integrated development environments. Borland, IBM, SAP, and SAS have also joined in a bid to add tools that will support Web services.

HANDSET ANGLE

For their part, the handset vendors have the most active developer-support operations, particularly in the form of online forums where developers can pepper their peers with questions relating to specific handsets. As might be expected from the largest handset vendor, Nokia's forums (<http://ncsp.forum.nokia.com/>) see heavy traffic.

For smaller vendors like Sendo, the stakes may be higher. The European handset manufacturer, which bolted out of Microsoft's Smartphone camp last year to embrace Symbian and Java, has mounted a major push to convince third-party developers to create apps that will put its phones on the map. Sendo offers tools and tips to developers who register for free at its site at <http://www.sendo.com/dev/>.

Although writing J2ME apps for individual handsets is the first step in the wireless programming progression, it may no longer be the most important step. Now, connecting J2ME-enabled devices to back-end services is taking on increasing importance because cellphone apps no longer exist in isolation.

To that end, Sun earlier this year announced plans to "launch an end-to-end mobile enterprise development platform to help service providers and enterprises quickly extend their current IT investments into 2.5 and 3G wireless networks."⁴

As part of the push, Sun is reenergizing its efforts to promulgate Jini, the network-connectivity software technology that's able to pick wireless clients out of the ether and connect them to back-end servers.

Jini, unveiled in January 1999, has been positioned as a platform-independent Web "dialtone," which enables users to connect any roaming consumer smart device into a company's network from anywhere. In hyping the technology at its launch, Sun raised expectations that

Jini-compliant smart devices would soon see widespread deployment.

That never happened. Jini has made slow but steady progress in the developer community, however. By making it easier for programmers to fold Jini connection capacity into cellphones and PDAs, Sun is hoping a new crop of Jini developer kits may spark a critical mass of further developer adoption.

Developers can download Sun's Jini technology kit version 2.0 at <http://www.sun.com/software/communitysource/jini/download.html>. V2.0 implements a new Jini security model, supports version 2 of the Jini discovery protocols, and has a new remote-methods invocation programming model with customizing APIs for each layer

As part of the push,
Sun is reenergizing
its efforts to
promulgate Jini.

of the protocol stack.

Finally, Sun has been expanding its wireless-Java hardware horizons beyond cellphones to embrace the Intel embedded processors that are increasingly seeing service in non-cellphone devices such as handheld computers and MP3 players.

The move has involved adding to J2ME support for Intel's XScale processors. Sun spun an optimized version of its HotSpot just-in-time Java compiler, targeted at XScale and called it CLDC HotSpot (after the CLDC profile in J2ME). Upon closer inspection, CLDC HotSpot seems to be an updated implementation of Sun's old-line embedded Java KVM (K virtual machine). Nevertheless, this is an important move in broadening Java's applicability, in no small measure because it's a free tool for an Intel target. (There's a useful white paper on CLDC HotSpot at http://java.sun.com/products/cldc/wp/CLDC_HI_WhitePaper.pdf)

REFERENCES

1. Walko, J. TI, INRIA collaborate on Java competence

center in France. *EE Times*, (Dec. 4, 2003); <http://www.eetimes.com/story/OEG20031204S0021>.

2. Wolfe, A. Microsoft's compact framework targets smart devices," *ACM Queue* (Oct. 2003), 10-12.
3. Taft, D. K. Java tool vendors unite in battle against .NET. *EWeek*, (Nov. 24, 2003); <http://www.eweek.com/article2/0,4149,1394340,00.asp>.
4. See reference 2.
5. See reference 3.
6. Sun press release. Sun Microsystems launches new products and programs to unite wireless java community and accelerate the deployment of mobile data services. (June 10, 2003); <http://www.sun.com/smi/Press/sunflash/2003-06/sunflash.20030610.2.html>.

ALEXANDER WOLFE received his electrical engineering degree from Cooper Union in New York City. A science writer based in Forest Hills, New York, he has contributed to *IEEE Spectrum*, *EE Times*, *Embedded Systems Programming*, and *Byte.com*.

© 2004 ACM 1542-7730/04/0200 \$5.00

JAVARESOURCES

SUN

Java 2 Platform Micro Edition

Version 2.1 beta release of the toolkit (free download)
<http://java.sun.com/developer/earlyAccess/j2mewtoolkit>

Useful App Notes

About writing your first app and sending it through an emulator
<http://wireless.java.sun.com/midp/articles/wtoolkit/>

Jini Technology Kit v2.0

Download Jini network technology
<http://www.sun.com/software/communitysource/jini/download.html>

CLDC HotSpot

Useful white paper about free tool
http://java.sun.com/products/cldc/wp/CLDC_HI_WhitePaper.pdf

NOKIA

Developer/Support forum

Provided by largest handset vendor
<http://ncsp.forum.nokia.com/support/?body=detail&aid=183>

JAVA.NET COMMUNITY

Java.net Community

Recently formed, loose, collaborative collection of mostly open source projects
<https://www.dev.java.net/>

Java for Communications Sub-community

Nexus of wireless-tools interest at Java.net
<https://communications.dev.java.net/>

Cellphonejava

Chief Java.net project
<https://cellphonejava.dev.java.net/>

Sendo

Sendo offers tools and tips to developers who register for free at its site
<http://www.sendo.com/dev/>