Wireless Java™ Technology: The Time is Now

This paper discusses why Java™ technology is the direct path to delivering the wireless Internet to mobile devices.

Mark Sears

Within the next few years, more than one billion cell phones will be in use — more cell phones than PCs will be connected to the Internet. Industry leaders in the wireless carrier, mobile device manufacturing, and content-service arenas have been quick to respond with plans to extend Internet-based services to wireless devices. Even mainstream media recognize that the move to a mobile Internet is the next big thing. But growing demand for such access includes the caveat that the technology must deliver personalized services and performance that match, or better, what consumers have come to expect from the wired Internet.

Navigating the Post-PC Paradigm Shift

Consumers who conduct more and more of their lives "on the fly" want full access to Internet-style services that are as personal and mobile as they are. This dynamic, connected lifestyle is pushing mobile devices into a new era of utility as network appliances; consumers expect them, at a minimum, to support e-mail, e-commerce, gaming, and Internet browsing.[1] Such diverse functionality requires an open, standards-based, networked technology platform that can bridge the gap between wireless carriers, device manufacturers, content providers, and application developers.

Until recently, the dissemination of wireless technology was constrained by closed platforms. The accessibility of networks and individual devices of the same corporate brand was, and in many cases still is, limited by proprietary technology and applications. Current technology offers two choices for moving to an open, standards-based delivery of Internet applications to wireless devices: using a markup language, such as WAP/WML or cHTML (i-mode), or using Java™ technology. Both of these approaches have benefits and limitations, but Java technology is the clear winner in its ability to provide a single, open platform that allows developers to create dynamic, personal, functionality-rich applications, which service providers can then deliver to the mobile device market.

Dynamic, Personalized, Interactive Content

The whole point of wireless mobility is that the context is very personal. Consumers want applications that can deliver individualized information. And factors other than personal preferences, such as location and circumstance, play a part in determining what exactly

[1]Among the many benefits Java technology brings to the mobile Internet market is the ability to establish a new paradigm for "surfing" the net, since the traditional PC-based browsing model is not an appropriate "fit" for small-display, limited memory devices such as cell phones, webpads, and PDAs. For more information, see the *Leveraging High Data Rates* white paper by G. Comeau (publication pending).

“individualized” means. For example, a tourist stranded at Heathrow airport in London, whose plane has just been cancelled, will have different information needs than when en route to the office. In the former case, important information might include the availability of other flights, the location of nearby hotels and restaurants, or even information about entertainment that evening. In the latter, important tasks might include booking a meeting, checking e-mail, or doing some on-line trading.

Markup languages like cHTML and WML are based on technology that cannot offer the kinds of interactive services that full application development platforms, such as the Java platform, can provide. Today’s WAP devices act as dumb terminals to which text files are sent and rendered into static web pages. Thus they are limited to delivering static web browsing with basic graphics capabilities, a model that precludes the delivery of dynamic, interactive, personal content.

Think of a wireless device as you do your desktop; sometimes you open a web browser to surf the web, while other times you want to use actual applications, such as spreadsheets, word processors, games, and graphics applications. The same dichotomy of choice is needed – indeed, demanded – for mobile devices. Markup languages do bring Internet browsing to mobile devices, as well as the ability to download and save a .zip or setup.exe file to a download folder and then to install and run that application. But this static model brings the desktop paradigm to the mobile device without changing its context to suit the limited memory, limited power, small display size environment of mobile devices.

With the Java platforms comes the ability to bring interactive applications to such devices. You can surf the web and, with one click of a button, choose to save the application to your device or only use it that one time. Java technology enables developers to write richer, more dynamic, personalized versions of applications already in use on existing mobile devices. Java technology-based applications support more content-rich graphics and faster interaction than browser-based environments (such as WAP or i-mode), enabling devices on the move to perform tasks as varied as e-mail, news updates, downloadable games, calendaring, and access to business data. Client-generated graphics can be vector-based, enabling a richer viewer experience and reduced network demand. As well, these applications can be downloaded and run locally on the client device, and upgrades or new applications can be downloaded “on the fly.” In essence, Java technology brings a brain to the mobile device — both the application and the code are brought to the device to interact with the user, and thus can be made dynamic, personal, and enriched by the context.

Security and Disconnected Access

While the Java platform is open and allows developers to create applications that are portable, dynamic, and personal, these key features are not the only reason the Java platform is best for the mobile Internet. Java technology also offers the ability to ensure data security and provide disconnected access — critical differentiators in the expanding wireless market.

The Java platform is inherently secure. Built-in features such as bytecode verification, no direct memory pointers, and digital signatures for dynamic applications provide a good security base. There are also multiple third-party extensions for the Java platform, covering areas like authentication and cryptography. As well, portable Java applications can easily be written to be distributed over TCP/IP, eliminating the potential for security breaches during gateway protocol conversions, a risk that currently exists with some wireless application gateways.

Java applications can also be designed to provide disconnected access so they can run even when they are out of coverage areas. Unlike browser-based devices, Java technology-enabled devices will allow applications to reside on the device itself so that processing is not interrupted when the network is unavailable. For example, the device or user can put certain tasks on hold and these tasks can be automatically triggered when the network connection is reestablished.

As well, Java code can be dynamically and securely downloaded from remote servers, providing the proper environment to run applications locally on client devices, regardless of whether the necessary library files were shipped with a particular device.

Why Java Technology Now?

A review of the previous strong arguments in support of the Java platform for the mobile Internet begs the question: Why not Java technology? While the Java platform allows for the development and widespread distribution of portable, content-rich Internet applications to the desktop (e.g., games, investment portfolios, and calendars), the portability of such applications to memory-constrained, low-power mobile devices, while possible, is not practical in terms of performance. So performance has been the chief obstacle to the widespread adoption of Java technology in this expanding mobile services market — Until now.

With the release of the Java 2 Micro Edition (J2ME™) platform, and particularly the Connected Limited Device Configuration (CLDC) from Sun Microsystems™, a small-footprint Java runtime environment developed directly for the mobile device market is now available. CLDC specifies the features of the Java virtual machine (VM)[2] and the core developer libraries to be used for small, low-power, memory-constrained devices. The J2ME platform was also built with closed environment security in mind, so it can leverage the platform security features available in existing security infrastructures such as WAP or end-to-end solutions. The J2ME technology does not attempt to create another separate Internet for wireless devices; instead it provides a direct path for tying existing wireless devices and applications into *the* Internet.

The J2ME Mobile Information Device Profile (MIDP), layered on top of CLDC, includes frameworks for application life cycles, HTTP networking, persistent data storage, and graphical user interfaces. MIDP is backed by technology heavy hitters like Motorola, Nokia, Sun, Siemens, Sony, NEC, and Symbian. Both CLDC and MIDP allow Java applications to deliver high-end security and graphics capabilities, along with the ability to download MIDlets (applications written using the MIDP framework) and store them on the device. And just like other Java applications, MIDlets are easy to create, install, upgrade, and port to different platforms.

---

[2]For more information about Java VMs, see the *Java Implementation Technologies* white paper by Mark Levey.

Java technology-enabled handheld devices will offer dynamic and secure delivery of applications and services in real time. Through CLDC and MIDP, J2ME provides a modular, scalable architecture to support the flexible deployment of technology to devices with diverse features, functions, and memory/performance capabilities. Embedding Java technology in the wireless device is the next logical step in maximizing the portability of applications and the performance and memory of mobile devices.

Wireless Java Technology is Here to Stay

Within three years, an estimated 60 percent of new cell phones sold will be capable of running Java technology-enabled applications. By next year, the major wireless development technology will be based on the Java platform. Java MIDlets running on Java technology-enabled devices offer the best method of delivering content-rich applications, more information, and more control over that information.

For carriers, the Java platform enables traditional functionalities like e-commerce, stock trading, and banking to be extended into the wireless realm. For device manufacturers, Java technology means their products will be capable of running a whole new generation of applications and services. For content providers, the cross-platform capability of Java programs reduces time to market and provides access to an entirely new set of customers.

The list of name companies embracing Java technology is impressive. They include wireless carriers such as NTT DoCoMo, Vodaphone, and Bell South Wireless; device manufacturers like Motorola, Nokia, Siemens, Sony, and Samsung; and applications/content developers as varied as BBC Broadcast, Bonita, Fidelity Investment, and the Weather Channel. All have endorsed the J2ME platform or are working on next-generation wireless Java applications.

Delivering Java Technology to Today's Devices, on Today's Internet

Java technology's benefits are broad-ranging: it's easy to use; offers cross-platform architecture; employs a simple language; represents an open, widely available model; and offers a large pool of developer talent to create content. As more access to information "on the fly" becomes the standard, Java technology will be there to deliver it. The question remains: who will deliver these Java technologies to the market?

While most current mobile devices still run on proprietary platforms, such as Microsoft's WindowsCE^{TM} or PalmOS^{TM}, Zucotto Wireless, with its Xpresso Java native processor and supporting software and hardware development platforms, is leading the drive toward making Java technology the standard open platform for mobile Internet device applications. The Xpresso processor, which natively executes Java bytecode and supports an optimized and enhanced J2ME/CLDC/MIDP framework, delivers to mobile devices the most efficient, optimized balance between high speed and low power consumption. By designing the processor to integrate with existing technology, Zucotto is creating a path for service providers and device manufacturers alike to "future proof" existing mobile devices by allowing them to seamlessly run the exciting, rich content of next-generation phones.

The Zucotto platform leverages Java and Jini™ technologies to provide widely available and robust applications for Java technology-enabled thin clients. The platform is designed to reduce the long development cycles caused by hardware and software integration issues, enabling OEMs to bring new, advanced wireless devices to market more quickly and with significant development cost savings.

Zucotto Wireless is the world leader in merging wireless connectivity with Java technology to create a unique, market-driven, semiconductor-based platform. In the coming five years, Zucotto products will connect a billion wireless devices with our products.

About Zucotto Wireless Inc.

Zucotto develops Java semiconductors, software, and tools for the wireless world. Through the combined power of the Jacknife™ Development Suite and the Xpresso family of semiconductors, our integrated wireless Java platform greatly accelerates time to market for manufacturers of new and innovative wireless devices and services.

About the Author

Mark Sears is a Technology Evangelist at Zucotto Wireless, where he works with customers and in the standards community to bring the power of Java technology to mobile devices. Sears has also contributed to the development of the Zucotto software layer solution for the Xpresso Java native processor. Currently Sears is involved with the Java Community Process as an expert group member in Java 2 Micro Edition (J2ME). He is helping to develop Java specifications for technologies such as Bluetooth™ Wireless Technology, RMI, Personal Digital Assistants, and the J2ME Platform Architecture in general.

Before joining Zucotto Wireless, Sears worked at both Sun Microsystems and AudeSi technologies as an engineer on applications and frameworks for embedded Java devices. Sears studied Computer Science at the University of Alberta, Canada.