



J2Me vs. .Net

White Paper

Abstract

This paper describes the Sun's Java 2 Platform Micro Edition – J2ME, and compares it to the .NET managed environment to Windows-only mobile devices with the .NET Compact Framework (CF).

J2Me vs. .Net

Revised by Chris Rindal
QA Technician, Tometa Software, Inc.

Tometa creates custom software for you

Tometa Software designs and develops robust software solutions for virtually all industries including in-house (vertical market) and retail software, some of which is on the shelves at your local software store. We focus our unique combination of creative, technical, and problem-solving skills on meeting our client's objectives. Because of our clarity of purpose, commitment to process, and broad professional skill sets, we are able to provide our clients with world-class solutions that are functionally superior and fully aligned with our client's strategic focus.

Balancing development speed, quality and cost is what we are all about. Tometa combines agile development practices with fixed pricing, so you know what the cost, end product, and delivery time table look like—up front. If we underestimate the effort, we complete the overrun on our dime. Simple as that. That's why large enterprise firms like Alcoa and NASDAQ choose Tometa.

Tometa's agile development expertise and low-overhead US location keep our prices comparable to offshore vendors – without offshore challenges. Using a fixed pricing model, we provide upfront visibility into a project's ultimate costs, end product and delivery schedule. Our clients like knowing that we have “skin in the game” – a fixed price that aligns our goals with yours, incenting us to get the job done right and fast.

Lastly, as a Microsoft Certified Gold Partner, Tometa Software, can customize its products or create custom web, client/server, and traditional applications. With programming experience in C#, C++, Visual Basic, CGI, HTML, RPG, Delphi, Java and many others; Tometa Software is uniquely positioned to meet your needs as a development firm.

[Check us out today](#)

Mobile commerce has grown in leaps and bounds in the recent years. Demand for smart phones, PDAs, pocket PC phones and PCs is no longer the sole prerogative of mobile professionals. Mobile devices have now effectively moved into the realm of consumer and corporate markets.

For mobile application developers, there are essentially two platforms on offer today. Sun's Java 2 Platform Micro Edition – J2ME – has long been established as a platform of choice for mobile clients and embedded devices. Since the fall of 2002, Microsoft has extended its .NET managed environment to Windows-only mobile devices with the .NET Compact Framework (CF). As a mobile application developer, how do the two platforms measure up? What are the features, capabilities and weaknesses, of each platform? More importantly, which one is best suited for developing and deploying your next mobile application?

This article offers a high-end comparison between the two platforms, with an emphasis on important features and supported third-party add-ons and development tools.

Platform Support

.Net CF runs only on one operating system – Windows. However, thanks to the Common Language Runtime (CLR), .NET applications are portable to more than 200 devices that support the Windows CE and Pocket PC-based operating systems. Other implementations of the CLI, such as Potable .NET and Mono, give limited portability of .NET applications on Unix-like operating systems, such as Linux, Mac and FreeBSD.

However, there is a plethora of devices that run on non-Windows operating Systems. Symbian, iDen, Brew and vendor-specific operating systems are some of the most widely used platforms on cell phones. Palm OS and RTOS (real-time operating system) are the most prevalent in lower-level PDAs and embedded devices. Even in higher-level PDAs, where Windows OS has a great market share, there is a growing shift towards Linux Symbian and IBM-based platforms.

This is exactly where J2EM excels. The Java motto of "Write one, Run Everywhere" holds true for mobile development. The multitude of mobile platforms mentioned above has built-in Java support. Third-party runtimes from Insignia and IBM port the code to all sorts of mobile platforms, including Windows. Developers who use J2EM can write their code without worrying about cross-platform portability.

Language Support

Cross-platform portability comes at a price... that of convenience and ease of use. Developers using the J2ME platform run into the thorny standardisation problem across various platforms, which can quickly get confusing. Different vendors have sought to extend the capabilities of J2ME by plugging their own proprietary extension packages. Features, such as multimedia playback and SMS (Short Message Service), supported by J2ME extensions and optional packages are not available on all devices. Even the standard J2ME platform itself – i.e. MIDP – can be implemented differently by various vendors.

.NET compensates for its lack of reach with ease of use and convenience. Developers already familiar with any of the languages supported by Microsoft Visual Studio - such as Visual Basic .Net and C# - can easily migrate to a .NET CF mobile environment. Existing libraries and standard Windows controls are available for mobile clients, meaning that developers can be up and running with a mobile project using their already existing knowledge.

The specification process

Because the .Net CF is closely associated with the Windows OS, Microsoft has taken a homogenous approach to new release specifications. There are no lengthy debates on features to incorporate into new releases, nor is there a customer-centred approach to implementation. Any new technology that comes to the fold conforms to existing APIs and tools.

By contrast, the cross-platform nature of J2ME means that specification and implementation are very much distinct processes. The JCP (The Java Community Process), an industry-wide committee encompassing various mobile solution providers, oversees the development of all new J2EM standard APIs, configurations, optional packages and profiles. This ensures portability and the continuation of the cross-platform promise of J2ME. When it comes to the deployment of the platform, every solution provider is free to differentiate and innovate by having its own implementation. This is the biggest challenge facing J2ME: implementing standardization, while giving mobile vendors a free hand to differentiate and innovate.

Consumer Applications

Consumer wireless applications are quickly catching the attention of mobile developers. Multimedia messaging on smart phones and mobile games on NTT networks are hot topics on development forums and circles.

Both platforms have strong support for consumer applications. Native APIs from the Windows Media Player and the Java Media Framework (JMF) enable multimedia playback. .Net CF rich Windows Forms UI library supports button remapping, direct canvas draw and double buffering. For its part, the MIDP on the J2ME includes animation and game control features. CDC devices have also the ability of supporting 3D games thanks to the Java Game Profile.

However, when it comes to the crunch of high-performance video-gaming, the two platforms do not size up due to the lack of direct hardware access. The emphasis from both Microsoft and Java is on enterprise mobile applications.

Enterprise Applications

Here we focus on two aspects crucial to mobile enterprise development: database and web services support.

Database Support

The two platforms support on-device databases for all offline capabilities. .NET CF has support for a large subset of ADO.NET (Active Data Objects). The standard relational database access on the J2ME Personal Package is JDBC (Java Database Connectivity). Third-party proprietary database implementations are also supported on the CLDC platform.

However, to fully leverage smart offline capabilities, on-device databases are hardly sufficient. There is a need for full synchronization and consolidation with enterprise backend databases. Both platforms lack a standard API for standard synchronization and rather rely on vendor's specific synchronization engines. Several vendors have devised their own synchronization engines, optimization mechanisms and support features. Here we look at some of the leading mobile database solution providers:

Microsoft SQL Server CE Microsoft SQL Server CE has full support for ADO.Net and VS.Net, making it the ideal choice for .NET CF. The database is lightweight (only 1.5 MB) but only buffers and synchronizes data to backend SQL Enterprise Databases.

Sybase iAnywhere Solutions iAnywhere's SQL Anywhere Studio is one of the most popular mobile databases in the market. This is largely due to the impressive UltraLite mobile database technology, which provides just the right functionality for mobile applications. This produces a small-footprint database, yet provides an impressive console for creating and monitoring your data.

The database runs on both .Net and J2ME platforms. It supports both .NET APIs (through VS.Net) and Java (through the Palm OS). There is also powerful assortment of database objects for both the ADO.Net and JDBC standards.

Synchronisation with iAnywhere and third-party enterprise databases (support for IBM DB 2, Oracle 9i and SQL Server) is managed and monitored through a set of APIs and a server, called MobiLink.

IBM DB2 Everywhere This lightweight version of DB2 Enterprise database supports JDBC and ODBC APIs. On MIDP, DB2 Everywhere provides a relational layer on the RMS, called FastRecordStore. Through IBM Synch, you can establish and manage synchronisation with most back-end databases.

Oracle9i Lite Oracle 9i Lite provides a small-footprint (50 K) and a rather large memory footprint (1 MB) for prototype and full-blown development environments. There is full support for JDBC and ODBC APIs. The database is implemented over the RMS on the MIDP, and provides access through SODA (Simple Object Database Access) and object-oriented technology. The synchronisation process is limited to Oracle enterprise back-ends only.

Web Services Support

XML Web Services are crucial for future enterprise application integration. SOAP (Simple Object Access Protocol) is now the established protocol for enterprise back end component access.

In the Microsoft camp, Web service classes have long been adopted and promoted to integrate with mobile devices. The .NET CF framework fully supports SOAP version 1.2 and provides a set of classes for common communication requirements. Developers can use these classes to model the remote service instead of creating their own code. There is also improvement in the performance front, with faster XML serialisation and deserialisation through the XMLSerializer class.

In the Java camp, SOAP client support has only been recently standardised through the J2ME Web Services API (JWSA). The specification is implemented in the J2ME development environment and some third-party vendors are now offering optional J2ME APIs in their mobile devices.

The J2ME Web services provide general Web Services implementation and specification access via profiles from J2ME CDC or CLDC. They come in two optional packages: one for XML Parsing (JAXP or Java API for XML Processing) and one for remote service invocation (JAX-RPC or Java API for XML-based RPC).

Development Tools

One of the most decisive factors in choosing either mobile platform is to leverage both existing knowledge and development tools. Developers can use existing knowledge when taking up new projects, thus reducing the learning curve and greatly increasing productivity. Additionally, development tools for use on both platforms can be assessed on cost and familiarity with the existing interface.

Developer Groups

Prospective .NET CF developers are most likely to come from a .NET development background. VB application developers will be familiar with the compiler, debugger and other development tools. .NET desktop application developers will easily migrate into productivity features, such as the GUI (Graphical User Interface).

For J2ME, most J2SE client application developers will be familiar with the APIs. Some learning may be required for the User Interface and Micro edition APIs.

.NET CF Development Tools

Microsoft's Visual Studio.NET IDE is an excellent development tool. The product consists of a .NET framework, a runtime engine and class libraries for rapid application development (RAD). From a .NET CF perspective, the unified development model provides similar features for both desktop and mobile applications. User Interface designs in a desktop environment can be migrated to .NET CF by merely copying visual components to a new designer Window. Non-visual components, such as timers and message queues can be visually composed using the Visual Component Designer (VCD). It is then only a matter of "drag and drop" to configure the required components.

The run-time environment features a radical shift in Microsoft's strategy of language-dependency. VS.Net offers a wide choice of programming languages, such as VB, C++ and Java. Built-in ADO.net tools also target a variety of databases, including SQL Server and Oracle. Another key feature is Web-services, with strong XML support across a variety of platforms and languages.

However, Visual Basic.Net is not without its drawbacks. First, it is by no means a cheap product. A fully-supported development version could run into thousands of dollars. Moreover, development and deployment are only limited to Windows environments. Another challenge is the complex documentation and multiple compilers.

J2ME Development Tools

The fact the J2ME targets a variety of mobile platforms and devices, has spawned a great number of vendor specific toolkits and command-line tools. For experienced developers, a fully-fledged IDE is essential. Some of the leading IDE's in the market include:

JBuilder with MobileSet: The renowned Java IDE supports the J2ME platform through an add-on module called MobileSet. The enterprise edition supports a rich set of features for developing, testing and designing mobile enterprise applications. It supports multiple runtime environments and JVMs, greatly simplifying multiple-platform development. It has also a rich UI, good UML design support and full end-to-end development capabilities.

jVise: These are a set of Eclipse modules to extend support to the J2ME platform. Eclipse itself only offers the core IDE functionality in the runtime engine. MIDP libraries, compilers and runtime environments act in as Eclipse plug-in modules to enable J2ME source code compilation and execution. The real value of jVise comes through the proprietary built-in tools for J2ME code reduction, obfuscation and application verification. For example, jVise provides multi-emulator support to test J2ME applications on a variety of devices before being test on a mobile device.

Metrowerks CodeWarrior Wireless Studio: The IDE only runs on Windows but has an impressive array of third-party packages, such as PointBase SQL databases and Agea mobile business acceleration suite. CodeWarrior supports many third-party MIDP out-of-the-box SDKs, as well as PersonalJava application development.

Each one of these IDEs provides reasonable application development support for J2ME mobile development. The big challenge is the lack of standardisation across development tools. Each vendor has its own SDK, editing tools and device emulators. Mastering all these tools and testing different emulators can prove tedious. Developers can also run the risk of developing a set of skills tied to a single IDE.

Conclusion: It All Depends on Your Business and Development Needs

Both the J2ME and .NET CF are excellent platforms for developing mobile applications. However, each platform has its own strengths and weaknesses. J2ME

beats .NET CF to the punch when it comes to portability across a variety of devices, and support for mobile databases and server products. J2ME vendors also offer a wider selection of add-ons and development tools. It is the best platform for mobile solutions on low-end devices, geared towards homogenous environments.

However, developing and deploying applications on the J2ME platform can be both challenging and tedious. The .NET CF is simple and ensures lower development costs and a minimum learning curve for developers. It is also well integrated within Microsoft's solutions and functionality. It is by far the platform of choice for developing high-end PDAs running on Windows-only platforms.