What's new in mobile and embedded Java ?
A Technology Update.

Speaker:
Terrence Barr
Senior Technologist for Mobile & Embedded Technologies
Oracle

1 Introduction

Java is pretty much everywhere these days, covering a broad range of platform and devices. They range from low-range smartcards, through embedded systems, e-book readers, ATM machines, copiers, through entertainment devices, such as settop-boxes, TV's, blu-ray player, to vast numbers of mobile devices and even automotive control systems.

This point is also driven home by some of the Java metrics. Today about 7 billion Java Cards have been shipped. There are 3 billion Java based handsets in the market today, with about 800 million new phones shipping with Java implementations every year. Five of the top-five mobile handset manufactures ship with Java runtimes and there are around 45.000 Java ME applications today. About 80 million Java-based TV devices are in the market. All Blu-ray Disc players ship with a Java implementation, and there are a large number of embedded devices that also use the Java runtime.

2 Java Platforms, Java Language and Java VMs

Looking at the Java platform from a broad perspective, the Java platform ranges from low-end smartcards to enterprise class server machines. They all feature the Java runtime or can all be programmed in the Java language. Some of the low-end devices might have a subset of the Java language, such as the Connected Limited Device Configuration, short CLDC, but by and large you can use the same programming methodology and the same development model across all these platforms.

With regards to the Java Virtual Machine, (Java VM) level, many of these platforms run some version of the HotSpot Java VM which is a highly scalable, very robust and mature virtual machine that has been available for many years. On the low-end device and platform scale there might be some scale down lightweight versions of the Java VM, such as CLDC hotspot implementation or others that are specifically designed to run well in constrained environments.

Up one level, at the API level, Java SE, the standard Java SE desktop APIs are available on many of these platforms. There are also additional APIs for verticals available such as Java TV and certain Java ME segments. The Java Card has another set of specific APIs for that application space. On the server and enterprise class environments there are also the Java Enterprise Edition API s, and then for many of the consumer devices there are Java FX APIs for creating rich and media-rich and graphically intensive user interfaces.

3 Announcements related to Java for mobile and embedded devices

At JavaOne a number of announcements were made related to mobile and embedded devices. Here are some of the objectives:

The primary objective is to modernize Java for mobile and embedded devices. This is specifically

the Java ME.next project which will be discussed in the presentation. Another objective is to deliver Java and web applications to all consumer devices, i.e. integrating HTML, Javascript, CSS and other web based technologies into the java stack.

As well as this we aim to add new device APIs to access hardware and operating system features on these platforms, such as specific graphics, implementations, near field communication, sensors, payment, telephony, location and instant-messaging services. Further we plan to continue to optimize the java implementations for small footprint and cpu-efficiency specially for the JavaCard, TV and mobile spaces. So, for example on phones further optimizations will be pursued for the ARM7 & ARM9 chipsets and for limited memory environments. On TVs there will be optimizations to blu-ray Java, DVB multimedia and Tru2way cable. And on the Java Card there will be additional optimizations and improvements in application spaces such as personal identity verification, national ID and Health Care and other application spaces. And finally Oracle will continue to provide consistent tooling and emulation across Java devices.

4 Java ME and project Java ME.next

We will be also looking at some of the highlights of project Java ME.next. Oracle is committed to modernizing the Java ME platform and a proposal has been discussed with the Java community process expert group.

The key elements of this modernization will include adoption of some of the features of the Java developers' kit, JDK 1.6, in particular in the following areas: the language, the VM and runtime. This will lead to more compatibility between CDC and CLDC. Updates or extensions to APIs in existing or new packages will be added as appropriate. All this is to be achieved while maintaining backward binary compatibility and without disrupting the business. Oracle intends to support JavaME.next in future products.

Further details about the Java ME.next project: The CLDC stack will include common enhancements to the language, VM features and libraries. An equivalent level of functionality will be retained, for example the security model will be the same as for CLDC 1.1.1. There will be no dynamic loading of classes and the generic connection framework will continue to be used for networking and similar features. For the Connected Device Configuration, the CDC stack, the same common enhancements to the library, the VM and the language will be used. The CDC and the foundation profile FP will be combined, and the CDC platform will receive a number of updates from Java SE in the areas of security, unicode, new input/output, networking and the logging optional packages.

5 What does this mean for Java for embedded devices?
Java SE and CDC platform stack

Java for embedded is comprised of two distinct configurations. One is the Java SE embedded configuration or platform stack, and the other is the CDC platform stack. The Java SE embedded side is basically a complete Java SE 5 or Java SE 6 platform with specific embedded features. This can either be a headless configuration that means without any graphics capability or the headful configuration, which has support for graphics adapters and the appropriate graphics packages.

The Java SE embedded stack with all the latest APIs and language features is targeted at mid-range high-end embedded devices. The other platform configuration is the CDC platform. CDC is currently based on a JDK 1.4 subset, so it does not contain the latest Java 5 & Java 6 features. And it is targeted towards lower-range and mid-range embedded devices. The CDC configuration also has a wide-range of optional packages that can be put on top of it. And CDC is of course the basis

for multiple embedded and TV industry standards.

Currently Oracle offers a number of supported platforms for embedded Java.

Version	CPU	Operating System	Headful/Graphics Libs	RealTime
CDC 1.1.2	ARM, MIPS, PPC, X86	Linux Kernel 2.6	Headless	
	Intel CE3100	OCAP MPEOS 1.1.4	DirectFB	
	Broadcom MIPS 7420/7440	OCAP MPEOS 1.1.4	DirectFB	
SE 5u10	PPC, X86	Linux Kernel 2.6+	Headless	
	x86	XP Embedded	Windows	
SE 6u10	ARM v5/v6/v7 SoftFP/HardFP	Linux Kernel 2.6	Headless	
	ARM v5 SoftFP	Linux Kernel 2.6	X11 R6+	
	ARM v6/v7 HardFP	Linux Kernel 2.6	X11 R6+	
SE 5u20	x64, SPACR, SPARCv9	Solaris 10 Update 6,7,8	X11	
	x86	RedHat MRG 1.2: kernel version 2.6.24.7-139.el5rt	X11	Yes
	x86	Novell SLERT (SP2) update 6: kernel version 2.6.22.19-0.35-rt	X11	Yes

The table Configurations - based on CDC or JavaSE, source: Oracle Inc 2010 lists a number of configurations, based on whether the Java stack is a CDC stack or an embedded Java SE stack. This is also based on the CPU-architecture and the operating system and whether or not the stack is headful or headless and whether it is real-time or not. Therefore Oracle offers a number of choices for supported platforms.

6 Java for embedded currently, improvements

As previously mentioned a number of platforms are supported, both by Java SE embedded and by the CDC platform. Ported to a variety of Industry Standard Architectures (ISAs), operating systems and graphics adapters, the new CDC release has many VM improvements, better portability, bug fixes, performance improvements and contains ports for additional platforms. On the product side a solution for TV is now available called the Oracle Java Media Client and a number of other embedded solutions. More information on these can be found on Oracle's embedded Java website which contains off the shelf downloadable embedded binaries and evaluation versions for a number of platforms.

7 Java for mobile devices and mobile developer challenges

As developers seek to build new applications in Java they face a couple of challenges.

One of these challenges is to leverage web content that is available on the web today and to leverage that content within the Java platform. This includes things like rendering of web content within the Java application. As well as this, developers want to access device capabilities which is something Java ME does very well with its optional packages and to access these device capabilities from JavaScript. That means accessing personal information like the address book or calendar on device, the location, sensor of GPS, accelerometers, bluetooth and so on.

Another challenge is to allow integration between Java and JavaScript. And finally once developers

use different programming paradigms and different languages, and differe nt runtimes such as Java and JavaScript and web-based content, the system has to be able to manage these multiple content types and runtime environments and to support different development environments. Just as there are Java applications and web applications and web-widgets, there might be applications that blend these two technologies together, java and web technologies. And it all has to integrate together with an application management system on the runtime.

8 Technology options for blending Java ME and web technologies together

There are a couple of technology options to blend Java ME and web technologies together.

One option is to have the Lightweight User Interface Toolkit (LWUIT) with an xHTML component bound to the application as a library, which allows the application to render xHTML content directly from within the application. And that is available on any Java ME device today, as long as the application developer uses the Lightweight User Interface Toolkit.

Another option is to use the java runtime with JSR290. This is the Java language and xml user interface mark-up integration which allows a java application to render web content such as a dom, a xml and html content including CSS and javascript within the java application. The last step of integration in terms of technology options would be a fully integrated Java ME and web runtime solution. This would allow applications to be blended where there is a Java component and a web based component that is blended within the same application. And that would also allow content both from a Java perspective and from a web perspective to be bundled into a single application with an integrated solution. That is something which is coming in the future.

9 Java ME + web the proposed architecture.

On the left are the main components, the standard Java libraries that are used by the Java applications. On the right we have a web or widget container, that contains all the components which are needed to execute web applications on the platform. The two communicate via Java-javascript-bridge, the web content is rendered by a web engine, the Java content by the Java VM. On top we have three different types of applications, a standard Java application, a web application and a blended application.

10 Oracle Java Wireless Client

An example of a product that Oracle offers in the space is the Oracle Java Wireless Client , which is a best-in-class Java runtime for the mobile market. The Oracle Java Wireless Client contains all the latest Java ME platform evolutions and there a a number of product highlights. For example it is optimized for applications that use the lightweight user interface toolkit. There is a path to application monetization using the built-in mobile store client. The Oracle Java Wireless Client is also optimized for a catalogue of leading mobile applications. The Oracle Java Wireless Client product is bundled with major feature phone operating systems and chipsets already in the market today. It offers best-in-class multitasking performance and footprint and support for a number of advanced features such as skinning, touchscreens, screen rotation support and much more.

11 Models within the Java ME and web technology

Within the Java ME + web technology architecture these three models are available. There is the model that uses the lightweight UI toolkit and renders xhtml content, that is available today. The next option is Java ME integrated with JSR290 which will be available as a product in 2011. And the final option is a fully integrated Java ME + web runtime which is planned for the future.

12 What is the lightweight UI toolkit or LWUIT?

The lightweight UI toolkit is an advance lightweight UI library that allows applications to deliver a compelling user interface which is consistent across many different devices. It is designed for today's handsets and devices. It is very portable, meaning there are versions of the lightweight UI toolkit which work for the MIDP, for the blackberry platform, for Android, for various CDC platforms, Java SE platforms, TV platforms and others. From a programming paradigm it is inspired by swing. It should be very familiar to swing developers. It has been optimized and scaled down in functionality to target the mobile and embedded space. This is also supported by a number of tools. It is open source under a GPLv2 license plus the classpath exception which means it is free for commercial use. All the source-code can be downloaded and inspected; and changes can be made to it. The community is very active, many developers and soft ware vendors around the world are using the lightweight UI toolkit for their application development.

13 Demo

We will look at a demo of the lightweight UI toolkit to showcase some of the high-level features.

13.1 What are some of the key benefits of the lightweight UI toolkit ?

One key benefit is rapid development, because the lightweight UI toolkit has a familiar API and is clean and simple. It is easy to get started. The lightweight UI toolkit makes deployment much easier, as in many cases a single application .jar-File or application binary is sufficient to cover a wide-range of devices and platforms. The lightweight UI toolkit takes care of many of the device differences. It is consistent and flexible and it is very customizable and extendable. As you can see in the screenshot below this is a LWUIT application for a TV set-top box and you can see it looks very different from the other screenshots that we have seen so far.

So you can tell that the lightweight UI toolkit is very customizable, it also offers what we call "Filthy Rich" UI, meaning you can make the user interface really engaging and visually compelling. The lightweight UI toolkit is also brandable and ~~it~~ is designed for mass-market devices. It is tested on a broad-range of hardware, meaning your LUID-based application can cover at wide-range of platforms; ranging from the low-end thru to the mid-range and ~~to~~ the high-end.

13.2 Some more key features of the lightweight UI toolkit.

It has a Swing-like module view controller. It features layouts, scalable fonts, a number of rich widgets, it has 3D and scalable vector graphics integration, it offers touchscreen support; animations and transitions are also available. It features a pluggable look and feel and theming, and it offers internationalisation and localisation support and much more.

13.3 Cross platform content.

One of the major benefits of the lightweight UI toolkit is that it automatically handles many device-specific details for the application developer. In many cases you can deploy identical application codes across multiple platforms and the application looks and feels exactly the same way. The three screenshots on below show the same LUIT demo application, the same application binary, running on three entirely different Java platforms. And you can see that they all look the same and feel and behave the same.

14 Device and web services mash-up demo

Rather than talking about the lightweight UI toolkit, its features and the features of the Java ME platform, we will look at a demo application that was built using the lightweight UI toolkit and other Java ME libraries and features. This demo is a device and web services mash-up demo which uses different features and functionalities from both the device as well as a number of web services to create a social networking mash-up application. The objective was to build a useful demo application that integrates a number of services that are on the device, such as GPS-location information and addressbook information and other things. And combine those with external web services such as mapping, direction information and local search information and mirror these in a demo application that is rich and compelling and showcases what you can do with the Java ME platform and the lightweight UI toolkit with fairly little effort.

15 Where can you find out more about the lightweight UI toolkit?

Go to the URL http://java.sun.com/developer/technicalArticles/javame/lwuit_v1_3/ for an article on LWUIT. The last article posted there offers an introduction to LWUIT and some of its features and a fairly large list of resources and links and other places where you can find more information

including tutorials, videos, developer guides. Also the demo mash-up application will be published as an open source project soon. So using that project you can browse the demo application source code and use it as a starting point for your own project. Please check the following blog http://terrencebarr.wordpress.com for updates and availability.

16 Summary

Oracle is committed to upgrading Java ME. Oracle is also committed to moving forward on making it easier for web developers and Java developers to work in a common environment. Oracle is also committed to increasing investment for embedded use of Java. Please check the URLs on the screen for further details in particular the Java ME roadmap details, Java ME embedded homepage and the Java ME homepage.

Java ME roadmap details:
oracle.com/technetwork/java/javame
Java Embedded home:
oracle.com/technetwork/java/javame/overview/index.html
Java ME home:
oracle.com/technetwork/java/embedded/overview/index.html

Contacts:

Terrence Barr (Speaker, Senior Technologist)
Oracle
Zettachring 10 A
Stuttgart, 70567
Telefon: +49 711 72098185
E-Mail terrence.barr@oracle.com

Barbara Ann May (Writing/Editing, Product/Field Marketing)
Oracle
Ampèrestr. 6
D- 63225 Langen
Telefon: +49 (0) 6103 752 195
Phone UK: +44 (0) 207 193 1552

Internet: http://www.oracle.com/technetwork/java/embedded/overview/index.html