Java 2 Mobile Edition (J2ME)

Mobile Computing ist bereits fixer Bestandteil unseres Alltages. Dabei verändern mobile Computer fortlaufend ihr Aussehen: Abgesehen von mittlerweile schon „klassischen mobilen Computern" wie Tablet-PC und Personal Digital Assistants (PDAs, Handhelds) sind diese in immer mehr Alltagsgegenständen (Waschmaschine, Fotoapparat, Kaffeemaschine usw.) zu finden und werden oft gar nicht mehr als Computer erkannt. Der mittlerweile am meisten verbreitete Computer ist „unser Handy". Um für die Benutzer einen echten Mehrwert zu bieten, ist es allerdings erforderlich möglichst benutzerfreundliche Anwendungen zu programmieren. Die Java 2 Micro Edition (J2ME) hat sich dafür als Plattform für mobile Anwendungen etabliert und erhält zunehmend Unterstützung verschiedenster Hersteller, wie z.B. Nokia, Motorola, Siemens, Sony-Ericsson usw. Um mobile Anwendungen mit der J2ME realisieren zu können, braucht es neben Kenntnissen der Programmiersprache Java und der J2ME APIs, vor allem ein gutes Verständnis des Umfelds mobiler Anwendungen und ihrer Integration mit verschiedensten Systemen auf der Serverseite.

Andreas Holzinger

Ein gutes Beispiel für die Schlagworte *ubiquitous* (allgegenwärtig) und *pervasive* (alles durchdringend) ist das Mobiltelefon, besser bekannt als „unser Handy": es ist immer und überall anzutreffen und die Verbreitung wächst weiter (siehe z.B.: http://www.wko.at/telekom/inter/tk_statistik.pdf).

Immer leistungsfähigere mobile Endgeräte – so genannte *„Smart Phones"* (**Bild 1**) erlauben es – in Verbindung mit entsprechenden „mobilen Applikationen" – Geschäftsprozesse und Arbeitsabläufe *(Workflows)* zu verbessern und dadurch Mehrwerte für Endbenutzer zu generieren.

Als „mobile Applikationen" werden Anwendungen bezeichnet, die auf solchen Endgeräten laufen. Eine wichtige Anforderung an diese mobilen Applikationen ist eine Anbindung an bestehende Infrastrukturen. Die Entwicklung mobiler verteilter Applikationen, die mehrere Systeme umfassen und auf möglichst vielen mobilen Endgeräten laufen sollen, ist eine anspruchsvolle Aufgabe. Als Grundlage ist es erforderlich, die Interaktion der einzelnen autonomen Endgeräte zu koordinieren und die funktionalen Anforderungen der Anwendung zu erfüllen. Das ist schon bei der Entwicklung von Anwendungen für den Einsatz zwischen stationären Systemen eine Herausforderung.

Bei der Entwicklung von „mobilen Applikationen" müssen stets folgende Faktoren berücksichtigt werden:

- **Begrenzte Ressourcen der Endgeräte** (z.B. Speicher, Performance, Energiereserven, aber insbesondere auch Displaygröße);
- **Unzuverlässige Netzverbindungen** mit stark schwankender Dienstgüte und Übertragungsgeschwindigkeit;
- **Unsichere Datenübertragung** über unsichere Netzwerke;
- **Komplexe Kommunikation** mit anderen Systemen.

Die *Java 2 Micro Edition* (J2ME, siehe: http://java.sun.com/j2me) ist eine von Sun Microsystems bereits 1999 entwickelte und speziell auf Endgeräte mit begrenzten Ressourcen zugeschnittene Java-Umgebung (**Bild 2**).

Bei der Einführung der Java 2 Familie hat sich die Firma Sun entschieden, eine Java 2 Standard Edition (J2SE) und eine Java 2 Enterprise Edition (J2EE) anzubieten. J2SE ist eine Sammlung von Tools und APIs, die Kernbestandteile zur Erstellung von Java Applets und Anwendungen enthält. J2EE enthält weitergehende Unterstützung für Datenbankzugriff, verteilte Systeme und Netzwerkkommunikation, um auch komplexe Anwendungen zu erstellen. J2SE ist (theoretisch) eine Teilmenge von J2EE, wobei J2EE die volle Funktionalität von J2SE besitzt plus spezielle Unterstützungen für den Unternehmensbereich.

Da gerade im Bereich mobiler Endgeräte vielfältige und grundverschiedene Hardwarevoraussetzungen existieren, hat sich Sun für eine weitere Version, der Java 2 Micro Edition (J2ME) entschieden.

Der Einsatz von J2ME ist für die Entwicklung mobiler verteilter Anwendungen besonders sinnvoll, da es noch keinen einheitlichen Standard für mobile Endgeräte gibt und sich die unterschiedlichen Endgeräte auf eine Vielzahl unterschiedlicher Betriebssysteme stützen.

Bild 1: ein Nokia Smart Phone (Quelle Nokia)

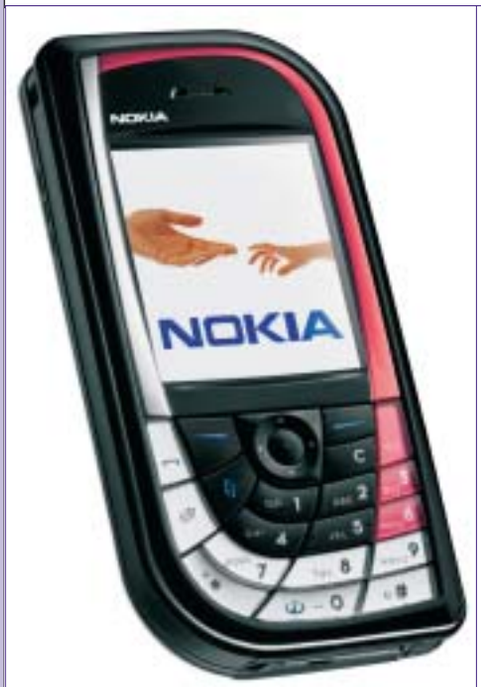
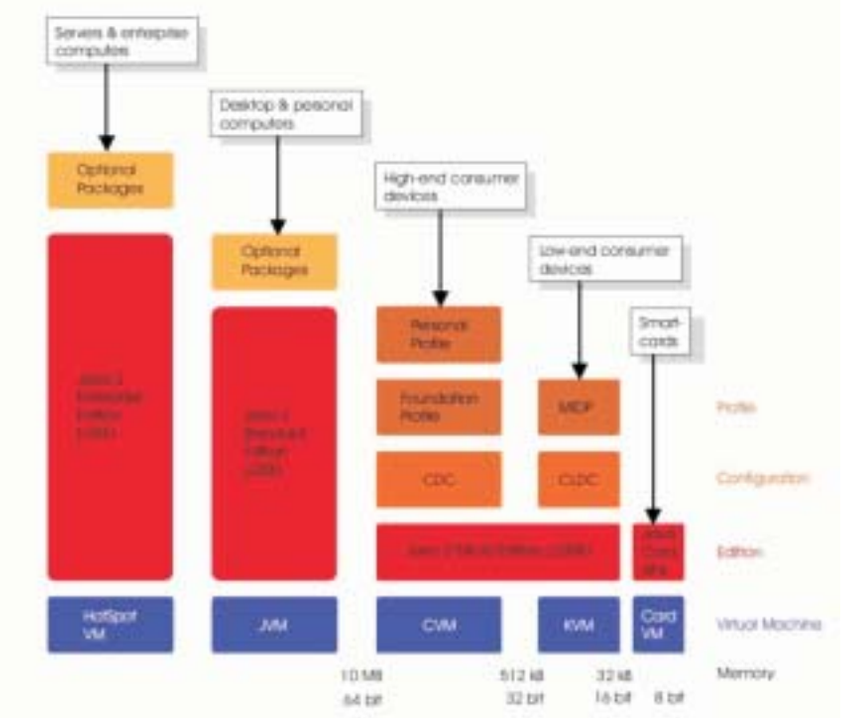


Bild 2: Überblick über die Java Versionen (Quelle Sun)



Daher ist ein Teil der J2ME für alle Geräte festgelegt und ein anderer Teil für eine entsprechende Gerätegruppe spezifisch ausgelegt. Diese spezifischen Auslegungen werden Konfigurationen bzw. Profile genannt.

Daher enthält J2ME allerdings Elemente die weder in der J2SE noch in der J2EE enthalten sind, was zwangsläufig zu einer Einschränkung der Kompatibilität führt.

Wie ebenfalls aus **Bild 2** ersichtlich, gibt es für die *Java 2 Micro Edition* zwei *Virtual Machines* (VMs): die so genannte CVM *(C Virtual Maschine)* und die KVM *(Kilo Virtual Maschine)*. Beide sind entwickelt worden, um Ressourcen zu schonen. Im Aufbau allerdings unterscheiden sich CVM und KVM erheblich, da sie für zwei unterschiedliche Geräte-Zielgruppen entwickelt wurden: Die *High-end consumer devices* (Internet-Bildschirmtelefone, Digitalreceiver usw.) benutzen die CVM, wohingegen die *Low-end consumer devices* (Handys, Handhelds usw.) die KVM benutzen.

Durch die geringeren Ressourcen mobiler Endgeräte (weniger Rechenleistung, Speicher usw.) und sonstiger fehlender Hardware-Unterstützung (z.B. Gleitkommaarithmetik), wurde die KVM nicht kompatibel zur allgemeinen *Java Language Specification* konzipiert. Abweichungen enthalten unter anderem: keine Gleitkomma-Berechnung, eingeschränktes Error-Handling, kein *Java Native Interface* (JNI), kein *finalize()*, keine benuzterdefinierten *Class Loader*, keine *Reflection*, keine *Thread Groups*, keine *Daemon Threads*, keine lose Kopplung u.a.

In J2ME existieren zwei Konfigurationen für unterschiedliche Gerätegruppen: *Connected Device Configuration* (CDC) für *High-end consumer devices* und *Connected Limited Device Configuration* (CLDC) für *Low-end consumer devices*.

Diese Aufteilung entspricht der Aufteilung der virtuellen Maschinen: CDC-Geräte benutzen die CVM und CLDC-Geräte benutzen die KVM.

In **Bild 3** sind die verschiedenen Schichten der J2ME-Spezifikation ersichtlich. Die unterste Ebene ist das Betriebssystem *(Host Operating System)*. Darauf baut die nächste Ebene auf, die aus der JVM (gehört intern noch zum Betriebssystem) und der Konfiguration (gehört intern bereits zur Benutzerschnittstelle). Das Profil entspricht einer Geräteeinteilung nach Funktion wie z.B. Handy, Handheld, Waschmaschine usw. Ein Profil setzt direkt auf einer Konfiguration auf, d.h. es benötigt diese als Grundvoraussetzung. Es bietet eine erweiterte API, die genau auf die Funktionen des Gerätetyps abgestimmt ist, für den es jeweils steht. Durch die Existenz der Profile müssen Programmierer nicht mehr für ein bestimmtes Gerät Software entwickeln, sondern eben nur für ein bestimmtes Profil. Jedes Gerät das dieses Profil unterstützt kann dieses Programm ausführen.

Das traditionelle „Hello World“ soll als Beispielcode dienen:

In der Praxis müssen Programmierer natürlich nicht jeden Schritt „von Hand“ programmieren, es gibt Entwicklungsumgebungen, wie z.B. die *Nokia Developer Suite Java 2 Micro Edition*, die sowohl für Windows als auch für Linux kostenlos erhältlich ist und das *Mobile Information Device Profile* (MIDP) 2 unterstützt. Der MIDP-2.0-Standard bietet eine geschlossene Plattform, die es Programmierern erlaubt, Applikationen zu erstellen, die auf allen MIDP-2.0-konformen Endgeräten laufen. MIDP 2.0 bietet dabei Funktionen wie die Möglichkeit, zusätzliche Kommunikations-Protokolle neben HTTP zu implementieren, darunter Bluetooth und *Short Messaging Service* (SMS). Diese lassen sich aus J2ME-Programmen heraus nutzen. Darüber hinaus unterstützt MIDP 2.0 aber auch eine sichere Kommunikation via HTTPS und es erlaubt darüber hinaus, J2ME-Applikation auf externe Ereignisse wie eingehende Telefonate oder zeitgesteuert reagieren zu lassen. Die Version 2.0 von Nokias Entwicklungswerkzeugen ist zudem mit einem Emulator ausgestattet, der auf MIDP 2.0 basiert (Übersicht siehe: `http://www.mobile2day.de`).

Glossarium

API *Application Programming Interface*. APIs sind Schnittstellen auf die Programme zugreifen können.

AWT steht für *Abstract Window Toolkit*. AWT ist ein in Java integrierter Toolkit zur Erzeugung von Benutzeroberflächen.

CDC ist die Abkürzung für *Connected Device Configuration*. CDC ist eine Konfiguration der J2ME und für leistungsstarke, mobile Geräte wie z.B. Subnotebooks gedacht.

CLDC bedeuted *Connected Limited Device Configuration*. CLDC ist eine Konfiguration der J2ME und für leistungsschwache Geräte wie z.B. Mobiltelefone gedacht.

JAR bedeutet Java Archive. JAR-Dateien beinhalten Java-Klassen sowie zusätzliche Infomationen, wie z.B. den Einstiegspunkt (Klasse, die die "main"-Methode enthält), in einem Manifest-File. JARs ermöglichen die Bündelung von vielen einzelnen Klassen in einer Datei.

JNI ist die Abkürzung für *Java Native Interface*. JNI wird in Java Anwendungen für native Methodenaufrufe benutzt.

JVM bedeutet *Java Virtual Machine*. Eine JVM ist eine VM für ein spezielles Betriebssystem. Die zur Ausführung von Java-Bytecode benötigt wird.

KVM steht für *Kilobyte Virtual Machine*. Eine KVM ist eine VM der J2ME und besonders auf die Bedingungen von ressourcenschwachen, mobilen Geräten zugeschnitten.

MIDP bedeutet *Mobile Information Device Profile*. MIDP ist ein Profil der J2ME und unterstützt vor allem ressourcenschwache Geräte.

MIDlets sind auf MIDP basierende Java-Anwendungen.

PBP ist die Abkürzung für *Personal Basis Profile*. PBP ist ein Profil der J2ME und baut auf dem *Foundation Profile* (FP) auf.

PDA bedeutet *Personal Digital Assistant*. Ein PDA ist ein mobiles, elektronisches Datenverarbeitungsgerät *(Handheld)*.

PP steht für *Personal Profile*. Das PP ist ein Profil der J2ME und baut auf dem PBP auf.

RMI-Profil(e) ist ein Profil der J2ME, das auf dem *Foundation Profile* basiert.

VM steht für *Virtual Machine*. Eine VM abstrahiert ein konkretes System, um ein einheitliches System zu simulieren. Dabei setzt die VM Aufrufe an das einheitliche System in Aufrufe an das konkrete System um.

Literatur

Klaus-Dieter Schmatz (2004):
Java 2 Micro Edition. Heidelberg: dpunkt.

Kim Topley (2002):
J2ME in a Nutshell. Sebastopol: O'Reilly.

Martin DeJode (2004):
Programming Java 2 Micro Edition for Symbian OS. Halsted Press

Eric Giguere (2001):
Java 2 Micro Edition: The Ultimate Guide to Programming Handheld and Embedded Devices. New York: Wiley.

Eric Giguere J2ME pages:
http://www.ericgiguere.com/j2me/index.html

The J2ME Platform:
http://www.javasoft.com/j2me/

MicroJava Network:
http://www.microjava.com

Bild 3: Schichten der J2ME-Spezifikation