MIDlet Technologie

- Ausarbeitung zum Vortrag -

von Gerald Strauch
Matr.-Nr.: 25210
Hochschule für Technik, Wirtschaft und Kultur (HTWK) Leipzig
Studiengang Informatik

Inhaltsverzeichnis

1. MIDlet Technologie – Java goes mobile

"*E pluribus unum*",
also "Aus den Vielen wird Eines" - prangt als große Losung auf jeder Dollarnote. Was hier zuvorderst politischer Anspruch, ist übertragen auf die Welt der Technik genau die Entwicklung, der sich das beginnende 21. Jahrhundert gegenüber sieht.
Gewiss, es wird nicht wirklich alles Eins, aber vieles kommt aufeinander zu. Und je mehr sich der Funktionsumfang verschiedener Geräteklassen vereinheitlicht, desto verschwommener wird auch ihre gegenseitige Abgrenzung.
Der sprichwörtliche Heim-PC verschmilzt unversehens mit dem Fernsehgerät, der Jukebox und dem Spielautomaten. Das Telefon wird zum mobilen Begleiter und vereint Sprache mit Textnachrichten, Video-Messaging, Web-Zugang und Gameboy. Ganze Häuser werden vernetzt und selbst der Kühlschrank geht online.

Diese Bewegung hin zu Integration und Vernetzung erfordert indessen nicht nur immer komplexere Datenverarbeitung, sondern auch gemeinsame Schnittstellen und Kommunikationskanäle. Unterschiedliche Geräte müssen einheitliche Verständigung lernen.
Doch auch über den reinen Datenaustausch hinaus erweisen sich gemeinsame Standards als segensreich. Denn so wünschenswert das Vorhandensein unterschiedlicher konkurrierender Plattformen auch sein mag, gehen damit doch ständige Parallelentwicklungen einher.

Eine zukunftsträchtige Antwort hierauf hat Sun Microsystems bereits in den 90er Jahren des letzten Jahrhunderts gegeben. Und diese Antwort lautet Java.
Anders als übliche Programme, die jeweils für ein Betriebssystem implementiert sind, das die benötigten Ressourcen bereit stellt, setzt ein Java-Programm auf einem abstrakten Standard-System auf. Dieses System, "Virtuelle Maschine" genannt, ist selbst nur ein Programm, das für jedes Betriebssystem separat implementiert ist. Auf diese Art ist es möglich, die unterschiedlichen Plattformen mit einer vereinheitlichten Java-Oberfläche zu versehen. Jedes Java-Programm wird damit auf jedem System mit installierter Virtueller Maschine ausführbar, gleichgültig, welches Betriebssystem darunter gerade seinen Dienst verrichtet.

Dabei vollzog sich der Siegeszug der Java-Plattform längst nicht nur auf dem Gebiet der Computer-Technologie. Zunächst vor allem für die Steuerung von Geräten eingesetzt, breitete sie sich schnell in unterschiedliche Bereiche der Technik aus.
Mit dem Erscheinen der Java-Version 2 kurz nach der Jahrtausendwende schließlich wurde auch der bislang kaum abgedeckte Bereich der Mobilgeräte erschlossen. Dergestalt unterteilt sich das System "Java 2" in drei Editionen. Neben der weitgehend kommerziellen Enterprise Edition (J2EE) und der verbreiteten Standard Edition (J2SE) wurde nun erstmals die Mobile Edition (J2ME) herausgegeben.

1.1 Mit J2ME alles unter einen Hut..?

Dass der Sektor der Mobilgeräte jedoch erheblich heterogener ist als der klassische Computerbereich, stellte die Entwickler dieser Edition vor eine Reihe von Schwierigkeiten. Neben einer Vielzahl von Modellen allein im Bereich der Mobiltelefone erstreckt sich der Anwendungsbereich der Mobile Edition auch noch auf Pager, PDAs, Telematik-Systeme und selbst Settop-Boxen, die von Haus aus kaum unter die Mobilgeräte zu zählen sind.
Ausschlaggebend für eine so breite Aufstellung der besagten Edition war also weniger der Wunsch, alle Mobilgeräte unter einer Plattform zusammenzufassen, sondern in einem Zug die unterschiedlichsten Geräte vergleichbarer Leistung zu versorgen.

Die Portierbarkeit von Java-Programmen ist hier der entscheidende Trumpf. Immerhin ist der Techniksektor, für den das J2ME konzipiert wurde, von enormer Zersplitterung geprägt. Das betrifft nicht allein die Architekturen der mehrheitlich mobilen Geräte, sondern auch die Betriebssysteme. Auf diesem Gebiet sind Standards bislang weitgehend ein Fremdwort, da die einzelnen Hersteller mit ihrer Technik jeweils proprietäre Software ausliefern.

In Abhängigkeit vom jeweiligen Funktionsumfang des Gerätes in Kombination mit den Spezifika dieser Software bestand die Aufgabe zunächst darin, einen geringsten Nenner für die Virtuelle Maschine und die darauf aufsetzenden Ebenen zu finden.
So benötigt die VM 128 KB an persistentem und 32 KB an volatilem Speicher. Des weiteren muss das native System mindestens 10 Threads unterstützen und eine Zeitgenauigkeit von mindestens 40 ms haben. Beim Gerätedisplay werden zumindest 96*54 Bildpunkte und eine Farbtiefe von einem Bit erwartet. Gemeinsam mit der allgemeinen Forderung nach Input- und Speichermöglichkeiten, einem Kernel, einem Timer sowie Netzwerkanbindung bilden diese Voraussetzungen einen technologischen Rahmen, den ein Gerät mindestens bereitstellen muss.
Weitere Grundlagen sind zwar (noch) nicht obligatorisch, haben aber zumindest den Status einer dringenden Empfehlung. Hierzu zählen ein Arbeitsspeicher von 256 KB und ein Display-Format ab 125*125 Pixel bei einer Farbtiefe von 12 Bit. Außerdem erwünscht sind die Unterstützung der Bildformate .png und .jpg sowie des HTTP 1.1-Protokolls und Zugang zum Telefonbuch des Gerätes.
Um zumindest eine Standardisierung dieser Anforderungen zu erreichen, ist es notwendig, eng mit den Herstellern zusammen zu arbeiten. Und in der Tat kam eine lange Liste namhafter Geräte-Anbieter und Mobilfunk-Provider zusammen, die gemeinsam mit Sun Microsystems und der Java-Community die Grundlagen der Mobile Edition schufen und diese auch in der Weiterentwicklung zukünftiger Technik mit berücksichtigen werden.

2. Die innere Struktur des J2ME

Quer durch die Mobile Edition vollzieht sich ein Bruch, und zwar entlang einer etwas unscharf umrissenen Leistungsgrenze, wobei der Aspekt der Mobilität hier unberücksichtigt bleibt.
Für leistungsstarke Geräte, vor allem High-End-PDAs und Settop-Boxen, die unter den Gesichtspunkten Energieversorgung, Internet-Anbindung, Speicherplatz und Rechenleistung über eine recht üppige Ausstattung verfügen, wurde folgerichtig eine eigene Konfiguration unter dem Namen "CDC" bereitgestellt. Dies steht für die "Connected Device Configuration", welcher gegenüber der Standard Edition eine nahezu vollwertige Virtuelle Maschine zugrunde liegt und die ein umfangreiches Paket von Java-Klassen aufweist - mit Einschränkungen in erster Linie im Bereich Bildschirmausgabe.
Mobilgeräte, die sich unter obigen Gesichtspunkten zwischen CDC-befähigter Technik und einer weit tiefer angesiedelten technologischen Untergrenze einordnen lassen, können mit der "Connected Limited Device Configuration" (CLDC) ausgerüstet werden.

Die Java 2 Mobile Edition untergliedert sich also jeweils in eine Konfiguration für Geräte höherer Leistungsfähigkeit (CDC) und solche mit geringerer (CLDC). Beide Konfigurationen unterscheiden sich beträchtlich in der Leistungsfähigkeit ihrer Virtuellen Maschine und im Umfang der enthaltenen Klassen, wobei die hier betrachtete CLDC weitestgehend eine Untermenge der CDC bildet. Sie beinhaltet überwiegend um Basisklassen, zum Großteil aus den Packages java.lang, java.io und java.util.

Auf den Konfigurationen setzen so genannte Profile auf, die gleichfalls Bestandteil der Mobile Edition sind. So ist auf der Basis von CLDC das so genannte "Mobile Information Device Profile" (MIDP) und ein eingenständiges PDA Profile auf dem Markt.

Diese Profile endlich tragen dem eigentlichen Verwendungszweck des jeweiligen Gerätes Rechnung. Für den Fall der CLDC, also der Konfiguration leistungsschwächerer Mobilgeräte, kann somit zumindest unterschieden werden, ob das Gerät vor allem als mobiles Informationsgerät (mit Stärken im Kommunikationsbereich) oder als klassischer PDA gedacht ist.
Die Konfigurationen unterteilen die Zielgeräte also zunächst grob nach ihren Fähigkeiten, die Profile daraufhin nach ihrer Verwendung.

Die Hierarchie des J2ME im Überblick

Wie ihr Name bereits erahnen lässt, leiten sich die hier betrachteten MIDlets von dem MID-Profil ab. Dieses besteht beinahe ausschließlich aus Klassen eines speziellen Pakets, das für Mobilgeräte entwickelt wurde. Es trägt die Bezeichnung "javax.microedition" und untergliedert sich weiter in Unterpakete, die für sich genommen jeweils einen Aspekt der Programmierung von Mobilgeräten abbilden.

Die Auslieferung von MIDlets erfolgt stets in Gestalt so genannter MIDlet Suites. Dahinter verbirgt sich lediglich eine Ordnerstruktur, in der das Haupt-MIDlet, eventuelle weitere MIDlets (die von ersterem aufgerufen werden können), Bild-Dateien und eine "Manifest-Datei" abgelegt sind.
Auf das Gerät übertragen wird die MIDlet Suite als gepacktes .jar-Archiv, aus dem sie später unmittelbar installiert werden kann.

2.1 Application Management Software (AMS)

Bevor ein MIDlet zur Ausführung gelangen kann, müssen aber zunächst seitens des Gerätes die notwendigen Ressourcen zur Verfügung gestellt werden. Dies betrifft zum einen die installierten Java-Komponenten, also Virtuelle Maschine und CLDC, MIDP-Laufzeitumgebung und die Bestandteile der auszuführenden MIDlet Suite (MIDlets, Bilder).

Zum anderen sind Prozessorzyklen, Speicher unf Display-Zugang unabdingbar.
Für all dieses ist eine Geräteschnittstelle zuständig, die als Application Management Software (AMS) bezeichnet wird. Ihr obliegt des weiteren die abschließende Entsorgung beendeter MIDlet-Instanzen sowie ganz grundlegend die Installation und Deinstallation von MIDlets. Dies schließt sogar Fragen von Kompatibilität, Autorisierung und Sicherheit mit ein.
Im Zuge verschiedener Zugriffsrechte auf bestimmte Ressourcen (z.B. Netzzugang) besteht eine Hierarchie von Sicherheits-Domains, in die ein MIDlet eingeordnet wird. Bei Anforderung nicht freigegebener Ressourcen obliegt es der AMS, vom Anwender die entsprechende Erlaubnis einzuholen. Sofern zum Beispiel die Rechte der Netznutzung nicht explizit bei Installation des MIDlets bzw. nachträglich in einem entsprechenden Kontextmenü erteilt wurden, quittiert die AMS den Versuch des Zugangs mit einer (geräteabhängig gestalteten) Anfrage an den Anwender.
Ähnlich wird auch der Versuch der Installation eines MIDlets mit einer Fehlermeldung beantwortet, sofern seine CLDC- bzw. MIDP-Version vom Gerät nicht unterstützt wird. Derzeit (2004/05) sind Geräte mit sowohl CLDC 1.1-, als auch MIDP 2.0-Unterstützung noch nicht sehr verbreitet; derartige Fehlermeldungen sind also bei entsprechenden MIDlets durchaus die Regel.

2.2 CLDC und MIDP

Grundlage dieses Zweiges der Mobile Edition bildet die so genannte "K Virtual Machine". Das "K" steht hierbei für Kilobyte, immerhin belegt die gesamte Laufzeitumgebung des CLDC lediglich 128 KB.
Das CLDC umfasst daher auch nur die grundlegendsten Klassen aus den bereits aus den Rechnerbereich üblichen Java-Packages java.lang, java.util, java.io sowie eine kleine mobile Zugabe unter dem Namen javax.microedition.io.

3. Die Packages und ihre Funktionen

3.1. java.lang: die Grundlage

java.lang beinhaltet die wichtigsten Sprachkonstrukte. Die Laufzeitumgebung ist hier in der Runtime-Klasse implementiert. Zusätzliche Systemfunktionen sind in der Klasse System enthalten, wie etwa Standardausgabe, Fehlerausgabe, Systemzeit, Programmterminierung oder Garbage Collector.
Die in der Laufzeitumgebung ablaufenden Prozesse sind von den Klassen Runnable und Thread abgeleitet. Beide verfügen über die run()-Methode, die eine Ausführung der betreffenden Objekte auslöst.
Grundlegend ist die java.Object-Klasse, von der fast alle anderen Klassen eine Reihe wichtiger Methoden erben; etwa equals(), hashCode() oder toString().
Klassen als solche leiten sich von Class ab und erben ihrerseits einige Zugriffsfunktionen.
Alle wichtigen Variablentypen, also Boolean, String/StringBuffer, Byte, Character, Short, Integer, Long sowie ab CLDC 1.1 auch die Gleitkommazahlen sind ebenfalls Bestandteil von java.lang. Sie enthalten jeweils diverse Methoden zur gegenseitigen Umwandlung bzw. String-Umwandlung. Einige weitere Operationen stellt die Klasse Math bereit.
Schließlich sind noch die allgemeinen Fehlerbehandlungs-Klassen Error und Exception sowie eine größere Zahl davon abgeleiteter, vordefinierter Einzelfälle (u.a. NullPointerException, VirtualMachineError) enthalten.
Gemeinsam ergeben diese Klassen den Kern eines lauffähigen Programms, mit dem alle grundlegenden Berechnungen und Variablenoperationen zu bewerkstelligen sind. Nicht Bestandteil von java.lang sind hingegen Grafik, Nutzerinterfaces und Kommunikationsschnittstellen.

3.2 java.util und java.io: Zusatz- und I/O-Funktionen

Die Basis für mobile Datenkommunikation liefert das Paket java.io. Hier sind In- und Output-Klassen mit ihren grundlegenden Lese und Schreibmethoden, eigene Reader- und Writer-Klassen sowie verschiedene Streams zur Aufnahme von Daten definiert. Die eigentliche Kommunikation erfolgt aber mit Hilfe weiterer Klassen aus anderen Packages, die auf java.io aufbauen.

Als Paket nützlicher Einzelklassen lässt sich java.util umschreiben. Hier finden sich mit den Klassen Random, HashTable, Stack, Vector, Enumeration, Date, Calendar und TimeZone in erster Linie weiterführende mathematische und organisatorische Funktionen sowie Zeit- bzw. Datumsklassen.

Als Faustregel lässt sich festhalten, dass beinahe alle Klassen der gerade beschriebenen Pakete bereits Bestandteil der Konfigurationen sind - im vorliegenden Falle also der CLDC. Innerhalb des MID-Profils kommen lediglich noch Timer und TimerTask bei java.util sowie eine Exception im java.lang-Paket hinzu.

3.3 javax.microedition.io: Generic Connection Framework (CLDC) und höhere Klassen (MIDP)

Dafür beinhaltet das CLDC aus dem Sektor javax.microedition nur einige Klassen des Unterpakets ~.io, die gemeinsam das "Generic Connection Framework" ergeben. Dieses definiert in erster Linie Basisklassen zum Aufbau einer Verbindung (Connector, Connection) und einige grundlegende Verbindungsinterfaces (u.a. InputConnection, OutputConnection, StreamConnection). Die meisten höheren Klassen (z.B. HttpConnection, SocketConnection, SecureConnection) steuern die Profile zu diesem Paket bei.
Außerdem ermöglicht SecurityInfo den Datenzugriff über eine verschlüsselte Verbindung. Eine weitere wichtige Klasse für den mobilen Bereich ist PushRegistry. Sie erlaubt den Start eines MIDlets, sobald von außen Daten an einem Port eintreffen, der für dieses MIDlet reserviert wurde. Auf diese Art ist es zum Beispiel möglich, automatisiert Empfangsbestätigungen abzusetzen oder Nachrichten zu speichern.

Alle übrigen Pakete im Paket javax.microedition sind Bestandteil des Profils MIDP.

3.4 javax.microedition.midlet - das Lifecycle Management

Die Package javax.microedition.midlet bildet die zentrale Komponente einer MIDP-Anwendung. Ihr obliegt die Erzeugung, Steuerung und Entsorgung der Instanzen von MIDlets.
Diese Aufgaben summieren sich in dem Begriff "Lifecycle-Management". Anders als herkömmliche Java-Programme verfügt ein MIDlet nämlich nicht über eine eigene main()-Methode. Statt dessen erfolgt die Ausführung innerhalb des Host-Systems durch direkten Aufruf. Realisiert ist dies alles in der Klasse javax.microedition.midlet.midlet.
Die drei definierten Zustände, in denen sich ein MIDlet befinden kann, werden vermittels der Methodenaufrufe von startApp(), pauseApp() und destroyApp(Boolean) eingeleitet. Dabei können sich startApp() und pauseApp() beliebig abwechseln, wobei jeweils der Großteil der benötigten Ressourcen angefordert bzw. freigegeben werden.
Mit destroyApp(Boolean) werden schließlich alle Ressourcen freigegeben. Der Boolean-Parameter entscheidet darüber, ob das MIDlet eine eventuell zum Zeitpunkt des Methodenaufrufs laufende Operation noch abschließen darf oder ob es bedingungslos terminiert wird.
In diesem Zusammenhang bedeutsam ist notifyDestroyed(). Diese als Convenience-Methode

bezeichnete Funktion benachrichtigt die Application Management Software vom endgültigen Ableben der MIDlet-Instanz und der damit einher gehenden Freigabe der Ressourcen.

3.5 javax.microedition.lcdui und javax.microedition.lcdui.game - die Nutzerschnittstellen

Praktisch keines der Geräte, für die J2ME kreiert wurde, verfügt über einen Bildschirm im klassischen Sinne. Auch Eingabegeräte wie Maus oder herkömmliche Tastatur sucht man hier vergebens.
Statt dessen verfügen Mobiltelefone und auch PDAs über klein dimensionierte LCD-Displays, auf denen die Fensterarchitektur eines AWT aus dem Desktop-Bereich wenig Sinn machen würde.
Infoge dessen wurde im MIDP eine komplett neue Bildschirmgestaltung gewählt.
Sie untergliedert sich in die so genannten High-Level-APIs und die Low-Level-APIs.
Erstere basieren auf der Screeen-Klasse, die einen Rahmen für eine Reihe von Elementen bildet, welche im Display zur Anzeige gebracht werden können. Hierzu zählen u.a. TextBox, List und Choice. Diese fungieren als Menü- und Eingabe-Elemente und können ihrerseits Text und/oder Bilder aufnehmen.
Die Klasse Graphics ermöglicht das vielfältige Zeichnen unterschiedlicher Linien, Bögen und grafischer Primitive sowie Flächenfüllungen.
Der Bereich der Low-Level-APIs ist noch weitaus variabler. Basale Klasse ist hier Canvas, die als eine Art Leinwand fungiert, auf der Elemente angeordnet werden können. Hierauf baut das gesamte Unterpaket javax.microedition.lcdui.game auf. Mit Hilfe seiner Layer- und TiledLayer-Klassen können Bilder auf unterschiedlichen Ebenen angeordnet, bewegt und als periodisch fortgesetzter Hintergrund eingerichtet werden.

3.6 javax.microedition.pki - Public Key Infrastructure

Das Package javax.microedition.pki (Public Key Infrastructure) sorgt für die Sicherheit bei Transfer und Installation von MIDlets mit Hilfe kryptologischer Maßnahmen.
Da Mobilgeräte aufgrund ihrer geringen Ressourcen noch kaum über Sicherheitsfunktionen wie Firewalls oder Virenscanner verfügen, ist sicherer Datentransfer hier von noch deutlich höherem Stellenwert als im Computerbereich. Zudem werden Daten ja üblicherweise in Form von Funkwellen übertragen, was ungebetene Teilnehmer zum Mithören geradezu einlädt. Mit dem PKI-Konzept steht somit die Prävention all dieser Beeinträchtigungen im Vordergrund.
Einzige Klasse des Pakets ist javax.microedition.pki.Certificate mit ihren Methoden, die alle Funktionen sicheren Datentransfers auf der Basis des X.509-Zertifikats realisiert.

3.7 javax.microedition.rms - die Speicherverwaltung

Das RMS-Paket organisiert die Verwaltung des Festspeichers, der dem MIDlet von der AMS zur Verfügung gestellt wird. Intern unterteilt das MIDP diesen Speicher in Blöcke, so genannte Record Stores, die ihrerseits als Array von Records aufzufassen sind. Ein Record als kleinste Speichereinheit besteht aus einem Byte.
Je nach Datentyp kann ein Wert daraus gelesen werden. Die Verwaltung erfolgt über die RecordID, einem individuellen Integerwert.
Für die Persistenz, also die dauernde Verfügbarkeit und Integrität der reservierten Speicherbereiche auch nach Reboots oder Batteriewechseln, ist die native Plattform des Geräts zuständig.
Sofern nicht explizit anders geregelt, bleibt der für eine MIDlet Suite reservierte Speicher für MIDlets anderer Suites unzugänglich - er steht nur den MIDlets innerhalb der Suite zur Verfügung.

3.8 javax.microedition.media und ~.media.control - Mediensteuerung

Vom Package javax.microedition.media und dem zugehörigen javax.microedition.media.control darf man mit Blick auf Multimediafähigkeiten keine Wunderdinge erwarten. Wegen der großen Heterogenität der Zielgeräte wären viele davon mit komplexeren Implementationen schnell überfordert - schon von ihrer Hardware her.
Dergestalt beschränkt sich die Funktion des Pakets weitgehend darauf, ein Player-Objekt zu erzeugen, mit dessen Hilfe unterstützte Medieninhalte wiedergegeben werden können.
Zudem können 128 verschiedene Töne definierbarer Länge und Lautstärke erzeugt werden, aus denen mit entsprechendem Aufwand auch Melodien generiert werden können.

Die folgende Grafik verdeutlicht noch einmal die Einteilung der einzelnen Packages der J2ME zwischen Konfiguration (CLDC) und Profil (MIDP).

Die Aufteilung der Packages zwischen Konfiguration und Profil

4. Hello World!

Ein einfaches MIDlet lässt sich bereits unter Verwendung von javax.microedition.midlet und javax.microedition.lcdui realisieren, wie das folgende Beispiel eines "Hallo World"-MIDlets verdeutlicht:

```
1 import javax.microedition.midlet.*;
2 import javax.microedition.lcdui.*;
3
4 public class HelloWorld extends MIDlet {
5
6 private Display display;
7 private TextBox textBox;
8
9 public HelloWorld() {}
10
11 public void startApp() {
12 display = Display.getDisplay(this);
13 textBox = new TextBox("Erstes MIDlet","Hello World",20,0);
14 display.setCurrent(textBox);
15 }
16
17 public void pauseApp() {}
18
19 public void destroyApp(boolean unconditional) {}
20
21 }
```

5. Wie kommt das MIDlet auf das Handy

Auf das Mobilgerät gelangt ein MIDlet fast immer auf dem Wege des mobilen Downloads. Diese Art der Datenversorgung wird auch als "Over the air provisioning" bezeichnet. Im Idealfall erfolgt der Download über eine sichere Verbindung auf der Basis des PKI-Pakets und unter Einbeziehung der digitalen Signatur des MIDlet-Anbieters. Damit kann dieser zweifelsfrei identifiziert und seine Vertrauenswürdigkeit bei entsprechenden Trust Centers nachgeprüft werden.
Hand in Hand damit geht auch die Versionskontrolle. Wie in den Ausführungen zum AMS bereits berichtet, verweigert dieses die Installation eines MIDlets, sofern es eine CLDC- oder MIDP-Version verwendet, die vom Gerät nicht unterstützt wird.
Ausschlaggebend für diese Prüfung ist der Zugriff auf die so genannte Manifest-Datei, die obligatorischer Bestandteil jeder MIDlet Suite ist. Sie findet sich in Gestalt einer kleinen Text-Datei im Stammverzeichnis der Suite. Im Prozess der Entwicklung des MIDlets wird sie üblicherweise beim Compilieren mit generiert und dort abgelegt.

Beispiel der obligatorischen Angaben im Manifest:

MIDlet-1: HelloWorld, /icons/top.png , HelloWorld
MIDlet-Name: HelloWorld
MIDlet-Description: Muster MIDlet
MIDlet.Vendor: Felix Muster
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEditoin-Profile: MIDP-2.0

Anhand der letzten Zeile würde die AMS eines Gerätes, das nur MIDP 1.0 unterstützt, den Versuch der Installation mit einer Fehlermeldung quittieren.

6. MIDlet-Entwicklung

Die Entwicklung von MIDlets erfolgt unter Verwendung der meisten etablierten Entwicklungsumgebungen wie etwa JBuilder, Sun ONE Studio oder NetBeans. Sofern die betreffenden Funktionen nicht bereits von Haus aus enthalten sind (z.B. im Sun ONE Studio), können sie als Zusatzpaket installiert werden (z.B. NetBeans). Zudem bieten viele der "hauseigenen" IDEs der Hersteller von Mobiltelefonen wie etwa Nokia oder Motorola gleichfalls die Möglichkeit, neben Anwendungen für die native Plattform auch Java-Programme auf der Basis von J2ME zu erstellen.
Zum Test der Programme ist es nicht notwendig, diese jedes Mal auf einem Mobilgerät zu installieren. Statt dessen stehen dafür Emulatoren zur Verfügung, die das Verhalten von eines solchen Gerätes authentisch nachbilden.
Zur Bedienung ist es naturgemäß nicht möglich, Aktionen durch Klicks auf Menüelemente oder in Eingabefelder vorzunehmen. Stattdessen erfolgt die Interaktion ausschließlich über die Tastaturelemente.
Häufig handelt es sich bei diesen Emulatoren um abstrakte Geräte mit verallgemeinerten Merkmalen. Gerade die Gerätehersteller aber bieten in ihren IDEs solche Emulationen, die der realen Technik in Design und Ausstattung weitestgehend entsprechen, sich also in praktisch jeder Hinsicht wie die Original-Technik verhalten. Dadurch werden nicht nur Tests beinahe unter Realbedingungen möglich, sondern es kann auch das Verhalten ganzer Geräte-Serien simuliert werden.

7. Weiterführendes

Über die integralen Bestandteile des CLDC/MIDP hinaus stehen noch eine Reihe von Zusatz-Packages zur optionalen Installation bereit, als da wären:
Mobile Media API
Wireless Messaging API
Java Speech API
Mobile Game API
Mobile Graphics 3D API
Location API
Real Time Specification

Unter Beanspruchung einiger Resourcen gestatten diese Pakete die weitere Konfigurierung der Geräte auf individuelle Bedürfnisse hin. Sofern die jewelige Technik nur mit der notwendigen Ausstattung aufwarten kann, ist das J2ME also auch in beträchtlichem Umfang skalierbar.
In Verbindung mit den gewaltigen Synergieeffekten, die aus der Portierbarkeit seiner Anwendungen resultieren, ist das J2ME als Zukunftstechnologie par excellence zu begreifen. Nicht zufällig hat es sich seit seiner Einführung vor wenigen Jahren bereits heute recht gut etabliert. Und - man darf noch einiges erwarten!

Quellennachweis

- Sun Microsystems, http://java.sun.com
- Sun Microsystems, http://developer.java.sun.com
- Java Community Process, About Java (u.a.), http://jcp.org/aboutJava/
- David Fox, Java 2 Micro Edition and the Mobile Information Device Profile, http://www.developer.com
- Vartan Piroumian, Wireless J2ME™ Platform Programming, Prentice Hall PTR, Palo Alto (California), 2002
- Kim Topley, J2ME in a Nutshell, O`Reilly Press, Sebastopol (California), 2002
- RRZN Hannover, Java (Begleitmaterial zu Vorlesungen / Kursen), Eigenverlag, Hannover, 1997