

Java in der Telekommunikation

S. Rupp

Objektorientierte Systembeschreibungen

Die sogenannte Objektorientierung ist mittlerweile zu einem gängigen Schlagwort geworden. Dahinter steckt ein relativ einfaches Konzept. Der Begriff eines Objektes orientiert sich eng an unserer Wahrnehmung und an unserer Art, sich mit unserer Umgebung auseinanderzusetzen. Bei Objekten des täglichen Gebrauches denkt man zum Beispiel an Telefone, Fernbedienungen, CD-Player, Digitalkameras, Digitaluhren, Set Top Boxen, Mediaplayer am PC, oder Chipkarten für Geldautomaten und Kasse.

Während zur Zeit der analogen Medien der Plattenspieler und der Telefonapparat wenig gemeinsam hatten, haben im Zeitalter der digitalen Medien die oben genannten Geräte durchaus Gemeinsamkeiten. Betrachtet man z.B. oben genannte Sammlung von Geräten, so fällt auf, dass viele der Objekte wiederum gemeinsame Unterobjekte besitzen, z.B. Displays, Eingabemöglichkeiten durch Knöpfe, Möglichkeiten zum Abspielen von Medien, bzw. Schnittstellen zur Vernetzung mit anderen Geräten.

Eine weitere Gemeinsamkeit der oben genannten Objekte besteht darin, dass es sich um digitale Geräte handelt, deren Funktionen als Software einprogrammiert wurden. Die moderne Alltagselektronik unterscheidet sich darin nicht mehr von den professionellen Applikationen und Servern in den Telekommunikationsnetzen.

Moderne objektorientierte Programmiersprachen beschreiben ein Objekt mit seinen Eigenschaften und den Methoden, die auf das Objekt angewendet werden können. Für ein Objekt "Medienplayer" kämen z.B. als Eigenschaft das Medienformat (Audio bzw. Video) in Frage, und als Methoden z.B. Start, Stop, Vorlauf, Rücklauf. Objekte können anderen Objekten ihre Eigenschaften und Methoden vererben. So lässt sich auf eine bereits vorhandene Basis zurückgreifen, indem man ein Objekt definiert, das ein geeignetes, bereits fertiges Objekt beinhaltet und erweitert. So kann man aus einem vorgefertigten Medienplayer einen eigenen Medienplayer generieren. Erweitert man einen Medienplayer um Objekte, die eine Vernetzung ermöglichen, und die nach Abfrage eines Telefonverzeichnisses eine Verbindung herstellen können, so erhält man ein multimediales Terminal.

Java als objektorientierte Sprache

Das besondere an Java als objektorientierter Programmiersprache liegt in der Offenheit des Standards und in der grossen Zahl bereits fertig spezifizierter Objekte. Software-Entwickler, die mit der Methode vertraut sind, können auf diesen Fundus zurückgreifen. Anfang 2002 wurde die Zahl der Java Entwickler weltweit auf über 2.5 Millionen geschätzt. In den kommenden 3 bis 5 Jahren geht man davon, dass sich diese Zahl nochmals verdoppeln wird. Auch wenn man nicht zum Programmierer werden will, lohnt sich eine überschlägige Auseinandersetzung mit der Terminologie von Java als objektorientierter Programmiersprache. Die hier verwendete Terminologie lehnt sich an den deutschen Sprachgebrauch an, wie in [1] beschrieben.

Objektklasse, Exemplar, Eigenschaften und Methoden

Basisbegriff eines objektorientierten Konzeptes ist das Objekt selber. Den Objekttyp bezeichnet man auch als eine Klasse. Jedes individuelle Objekt ist ein Exemplar seiner Objektklasse. Im englischen Sprachgebrauch verwendet man für den Objekttyp die Bezeichnung class, die gleichzeitig ein Sprachelement in Java ist und für das Exemplar der Klasse die Bezeichnung Instance. Wollte man zum Beispiel als Objekttyp einen mathematischen Punkt beschreiben, so würde man eine Klasse Punkt definieren. Aus der Klasse Punkt lassen sich dann im Programmtext Exemplare erzeugen, beispielsweise, um Punkte an bestimmten Koordinaten in einer Ebene anzuordnen. Zur Laufzeit eines Programmes finden sich die Exemplare eines Objekttyps im Arbeitsspeicher des Rechners wieder.

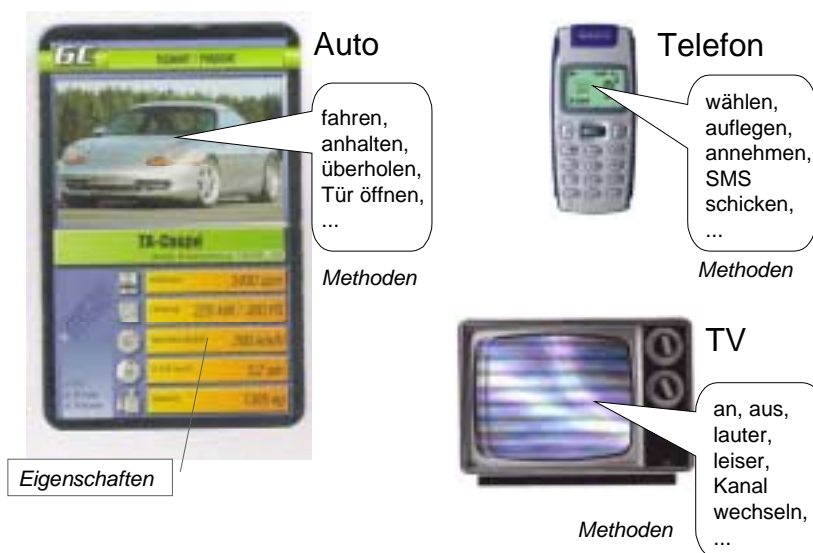


Abb. 1 Objekte, Eigenschaften und Methoden

Zur Beschreibung des Objekttypes gehört allerdings mehr als die Definition eines Namens. Ein Objekt hat in der Regel bestimmte Eigenschaften oder Attribute. Bei der Klasse "Punkt" wären die Eigenschaften beispielsweise die kartesischen Koordinaten x und y . Ein Beispiel aus dem täglichen Leben wäre beispielsweise ein Objekt vom Typ "Auto". Unabhängig vom jeweiligen Modell erwartet man von einem Auto grundsätzliche Eigenschaften, z.B. die Anzahl der Türen oder die Geschwindigkeit.

Ausserdem enthält der Objekttyp auch die auf dem Objekt ausführbaren Operationen, bzw. anders ausgedrückt die auf das Objekt anwendbaren Methoden. Methoden für den Objekttyp Punkt wären beispielsweise die Erzeugung eines Punktes an der Stelle (x,y) bzw. die Verschiebung

des Punktes um (dx, dy). Passende Bezeichnungen für die Methoden in einem Programmtext wären dann beispielsweise "teleport Punkt (x,y)" , bzw. "move Punkt (dx,dy)". Das Alltagsobjekt "Auto" verhält sich ähnlich. Geeignete Methoden wären hier "fahren", "anhalten" usw. Während beim Autoquartett bzw. für Statusbewusste beim Auto die Eigenschaften im Vordergrund stehen, beschäftigen uns im alltäglichen Gebrauch eher die Methoden. Das gilt auch für andere Gebrauchsgegenstände wie das Telefon oder den Fernsehapparat (wie war das noch gleich mit der neuen Fernbedienung?).

Bei traditionellen, prozedurbezogenen Programmiersprachen greifen Prozeduren und Funktionen auf einen Datenbestand zu und verarbeiten ihn. Im Unterschied zur prozedurbezogenen Sprache sind bei einer objektorientierten Sprache die Daten und Prozeduren im Objekt eingebettet. Die Objekttypen enthalten ihre Datenelemente (d.h. die Eigenschaften des Objekttyps) und die auf das Objekt anwendbaren Operationen (d.h. die Methoden des Objekttyps). Dadurch wird es sehr einfach, auf bereits vorhandene Programme anderer Entwickler zurückzugreifen und Programmtexte miteinander auszutauschen.

Eine weitere Eigenschaft objektorientierter Sprachen ist die sogenannte Vererbung. Darunter ist zu verstehen, dass man aus existierenden Objekttypen weitere Objekttypen generieren kann, die zusätzliche Eigenschaften und Methoden besitzen. Aus dem Objekttyp "Punkt" lässt sich beispielsweise ein Objekttyp "Kleck" generieren. Ein passender Pseudocode dazu wäre "class Kleck extends Punkt". Die zusätzlichen Eigenschaften wären die Form und die Farbe des Klecks. Die vererbten Methoden "teleport()" und "move()" bleiben für den neuen Objekttyp Kleck erhalten. Als weitere Methoden kämen Operationen zur grafischen Darstellung in Frage. Bei Autos gibt es das Konzept der Vererbung ebenfalls. Zwar nicht im alltäglichen Gebrauch, aber immerhin im Prozess der Herstellung. Aus einem Fahrwerk lassen sich in Kombination mit unterschiedlichen Motoren, unterschiedlicher Karosserie und unterschiedlicher Ausstattung andere Modelle erzeugen. Für ein Fahrzeug mit Klimaanlage gibt es für die neuen Eigenschaften dann auch neue Methoden für die Bedienung.

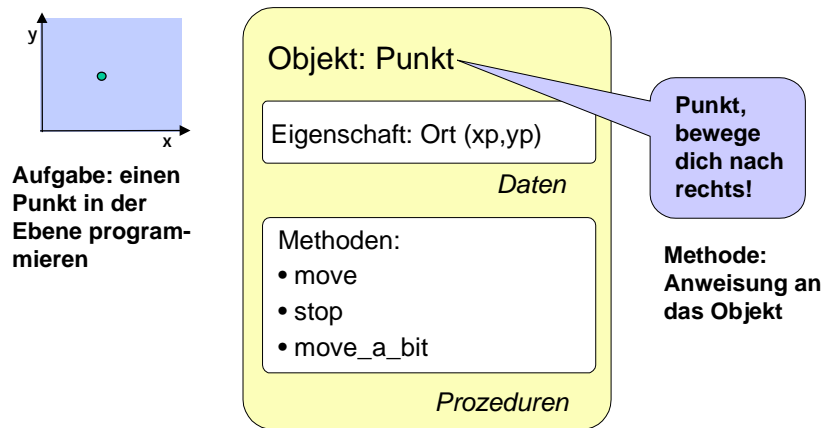


Abb. 2 Objektorientierte Programmierung

Um die Verwendung vorgefertigter Objekte und der Vererbung weiter zu unterstützen, gibt es bei Java ein weiteres Sprachelement, nämlich das sogenannte Interface. Ein Interface ist kein fertig implementierter Objekttyp, sondern eine Vorform, die eine abstrakte Struktur und sinnvolle Bezeichnungen für Methoden enthält, allerdings keine ausgeführte Implementierung. Aus einem Interface wird ein Objekttyp durch die Anweisung "class Objekttyp implements Interface".

Sourcecode und Bytecode

Weitere Begriffe aus dem Sprachgebrauch von Java sind Sourcecode und Bytecode. Unter dem Sourcecode versteht man im allgemeinen Sprachgebrauch den Quelltext der Programme, der sich mit konventionellen Texteditoren schreiben lässt. Sourcecode findet sich in Dateien mit der Endung ".java". Durch Aufruf eines JAVA Compilers wird der Quelltext umgewandelt in den sogenannten Bytecode und findet sich dann in einem Dateiformat mit der Endung ".class".

Der Bytecode wird von einer sogenannten Java Virtual Machine ausgeführt. Java Virtual Machines werden für unterschiedliche Betriebssysteme und Prozessortypen angeboten. Durch die Ausführung innerhalb einer Virtual Machine werden Java Anwendungen auf allen Systemen portabel, für die eine Virtual Machine angeboten wird. Eine Übersicht über die unterschiedlichen Java-Releases und ihren Funktionsumfang findet sich im folgenden Abschnitt.

Vorab jedoch noch einige weitere Begriffe aus dem Sprachgebrauch von Java. Ein Wort, das üblicherweise als erster im Zusammenhang mit Java assoziiert wird, ist das sogenannte Applet. Ein Applet ist eine Java Anwendung, die von einem Browser gesteuert wird. Letzten Endes wurde Java gerade durch die interaktiven Möglichkeiten populär, um die es Webseiten im Internet bereichert hat. Der Applet-Bytecode wird vom Browser geladen und ausgeführt. Der Browser besitzt dazu eigene Laufzeitumgebung für Applets. Wegen der Einbindung in die Umgebung des Browsers und weil der Browser viele Steuerfunktionen der Applets übernimmt, weicht die Programmierung von Applets ein wenig von der Programmierung von Applikationen ab.

Für Applets steht in HTML ein sogenanntes "Applet Tag" zur Verfügung, d.h. eine Markierung im HTML-Text für den Aufruf eines Applets. Mit Hilfe eines URL kann der Ort angegeben werden, von dem der Browser das Applet laden soll, sofern sich das Applet nicht auf dem gleichen Verzeichnis befindet wie das HTML-Dokument. Die Laufzeitumgebung (Run Time Environment) wird durch den Browser bereitgestellt, so dass der Server mit der Ausführung des Applets nicht belastet wird.

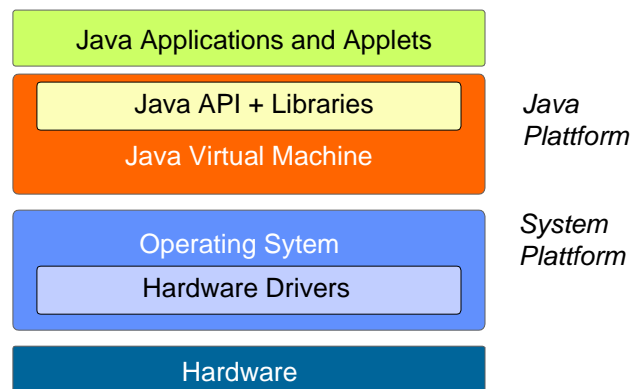


Abb. 3 Java Architektur für portable Anwendungen

Eine weitere Besonderheit von Applets ist die, dass sie in einer sicherheitstechnisch eingeschränkten Umgebung ablaufen. Einem nicht näher bekannten Applet ist es nicht gestattet, Dateien zu lesen, zu schreiben, zu erzeugen oder zu löschen. Ein Applet darf ausserdem keine Programme im Zielsystem aufrufen und keine Verbindung aufnehmen zu weiteren Maschinen im Internet.

Java unterstützt die Parallelisierung von Vorgängen durch den Anwendungsprogrammierer. Aufgaben wie beispielsweise die Abfrage einer Tastatureingabe, eine Bildschirmausgabe oder die Kommunikation mit einem Server im Netz lassen sich in voneinander getrennten Abläufen durchführen, den sogenannten "Threads". Einzelnen Abläufe können un-

abhängig voneinander auf Ereignisse warten, bzw. Ausnahmesituationen behandeln (die sogenannten Exceptions, wenn beispielsweise eine von der Anwendung gesuchte Datei nicht gefunden wird oder nicht geöffnet werden darf).

Programmierschnittstellen (API) und wie sie entstehen

API - Unter einem API (Application Program Interface) versteht man einen Satz von Spezifikationen über Objekttypen und Interfaces für einen bestimmten Anwendungszweck. Im Grunde genommen lässt sich jedes Objekt in unserer Umgebung in objektorientierter Weise beschreiben. Anwendungsentwickler können auf diese Spezifikationen zurückgreifen. Zu den Spezifikationen gehören ausser der Dokumentation des API in der Regel White Papers und technische Vereinbarungen. Eine API Spezifikation ist ein recht umfangreiches Dokument und am besten als HTML-Text mit Hilfe eines Browsers mit mehreren Frames (Fenstern innerhalb des Browser-Fensters) zu lesen. Verknüpfungen durch Hyperlinks innerhalb des Dokumentes erleichtern dabei das Nachschlagen und die Orientierung.

Java Community Process - Der Java Community Process ist der Standardisierungsprozess für objektorientierte Spezifikationen. Wie bei anderen Standardisierungsgremien üblich, werden zu unterschiedlichen Themen Vorschläge eingebracht, kommentiert und weiter entwickelt. Das Internet selbst liefert die Mittel für eine solche kollaborative Arbeitsweise. Der Java Community Prozess mit dem aktuellen Stand seiner Spezifikation lässt sich im Web einsehen. Eine URL findet sich im Literaturverzeichnis unter [2].

Referenz-Implementierung - Eine Referenzimplementierung ist fertiger Quellcode für eine gegebene Spezifikation und den zugehörigen Anwendungen.

Runtime Environment (Laufzeitumgebung) - Um Java Programme auf einer bestimmten Plattform mit gegebenem Betriebssystem und Prozessortyp abzuspielen, benötigt man ein sogenanntes Run Time Environment. Das Runtime Environment enthält eine passende Virtual Machine, einen Compiler und einige zusätzliche Hilfsmittel wie z.B. einen Applet Viewer zum abspielen von Applets. Die Virtual Machine stellt auch die systemtypischen Funktionen bereit, wie z.B. die Verwaltung des Arbeits-

speichers, Absicherung gegen fehlerhaften oder bösartigen Bytecode, sowie die Verwaltung der Threads für die Parallelisierung von Prozessen.

Development Toolkits - Hersteller von Java kompatiblen Produkten stellen weitere Entwicklungswerkzeuge für ihre Produkte bereit, z.B. Emulatoren für mobile Endgeräte passend zu Java Toolkits bzw. komplexere Entwicklungsumgebungen wie z.B. JBuilder oder Forte.

Java Releases - vom Server bis zum Haushaltsgerät

J2EE - Java 2 Platform, Enterprise Edition

Die Enterprise Edition wendet sich an die Infrastruktur in Unternehmen. Sie bietet Unterstützung für Intranet Server mittels Java Server Pages, Java Servlets und Enterprise Java Beans. Ausserdem unterstützt werden objektorientierte Schnittstellen (CORBA), XML und Sicherheitsfunktionen. Ein Vorteil der professionellen Java Edition ist die Skalierbarkeit auf die Grösse der jeweils benötigten Infrastruktur. J2EE unterstützt zu diesem Zweck mehrstufige Client-Server Architekturen. Ein Beispiel für eine dreistufige Architektur wären einfache Clients zur Ausführung der Benutzerschnittstellen (die sogenannten Netzcomputer), leistungsfähige Server zum Abspielen der Applikationen und als dritte Stufe Server für Datenbankzugriffe. Nachfolgend einige kurze Erläuterungen zu den oben genannten Begriffen aus der professionellen Java Edition.

Java Server Pages (JSP) – Wie der Name bereits andeutet, wird Java benutzt, um auf dem Server Web-Seiten zu generieren, also um HTML-Inhalte zu produzieren oder zu ergänzen. Im HTML-Text wird dazu eine Java Methode aufgerufen. Ein praktisches Beispiel wäre die Einblendung von Datum und Uhrzeit in eine HTML-Seite.

Java Servlets – Der Name deutet bereits an, dass Servlets so etwas ähnliches sind wie Applets. Der Unterschied besteht darin, dass Servlets auf dem Server ausgeführt werden und nicht auf dem Client. Servlets sind also vergleichbar mit CGI-Scripts. Gegenüber CGI-Scripts haben Servlets allerdings den Vorteil, dass alle aktivierten Servlets innerhalb eines einzigen Prozesses auf dem Server laufen, während für jedes aufgerufene CGI-Script jeweils ein neuer Prozess im Betriebssystem des Servers gestartet wird.

Enterprise Java Beans – Für Java Beans bietet J2EE ein eigenes Development Kit (das sogenannte Bean Development Kit BDK). Java Beans sind Objektklassen mit Standard Schnittstellen, die sich miteinander kombinieren lassen und deren Programmierung durch ein visuelles Tool vereinfacht wird. Die Kommunikation über Standardschnittstellen soll im Netz verteilte Anwendungen unterstützen. Während reguläre Java Beans direkt auf der Virtual Machine laufen können, benötigen Enterprise Java Beans (EJB) eine spezielle Umgebung, nämlich einen sogenannten Container. Der Container übernimmt die generell benötigten Systemfunktionen des Servers, z.B. für den Zugriff auf Datenbanken, die Abwicklung von Transaktionen, Zugangskontrolle, CORBA-Schnittstellen, Load Balance usw. Der Programmierer von EJBs braucht sich um diese Funktionen nicht mehr zu kümmern und kann sich auf die Programmierung der Anwendungen konzentrieren.

J2SE - Java 2 Platform, Standard Edition

Die Standard Edition ist für die typische Desktop-Umgebung geschaffen. Das Java 2 Service Development Kit der Standard Edition enthält einen Java Compiler für die gängigen Desktop Plattformen (Windows, Linux, Solaris), eine Java Virtual Machine, Class Libraries, Applet Viewer zum Abspielen von Applets ausserhalb eines Web-Browsers, einen Debugger, Tools und Dokumentation.

Eine Übersicht über die APIs der Standard Edition findet sich in [1], bzw. auf den Web-Seiten von Sun Microsystems. Typische Anwendungen sind die Programmierung von Applets für Web-Seiten, grafische Benutzeroberflächen in einer Desktop-Umgebung, bzw. Kommunikation in vernetzten Umgebungen. Ein weiteres Paket, Java Media Framework, erlaubt die Programmierung interaktiver multimedialer Anwendungen.

J2ME - Java 2 Platform, Micro Edition

Das für Anwendungsentwickler von Telekommunikationsdiensten interessanteste Release ist das Java 2 Micro Environment. Während die J2SE auf Geräte der Leistungsklasse von PCs und Notebooks zielt, bietet das J2ME Unterstützung für Geräte mittlerer und geringer Leistung, wie digitale Set-Top-Boxen, Organizer, Mobiltelefone und Kartenleser. Gegenüber der J2SE hat J2ME für einen Anwendungsentwickler übrigens den

Vorteil des vereinfachten Funktionsumfangs und der vereinfachten API. Die vom Micro Environment unterstützten Geräte sind typischerweise kleiner und einfacher zu bedienen als ein PC. Solche Geräte besitzen beispielsweise keinen bzw. keinen vollständig ausgestatteten Browser, ein wesentlich kleineres Display, keine Fenster-Oberfläche und keine Maus, wenig Arbeitsspeicher, wenig Prozessorleistung. Für portable Geräte muss immer Rücksicht genommen werden auf die Betriebsdauer der Akkuladung.

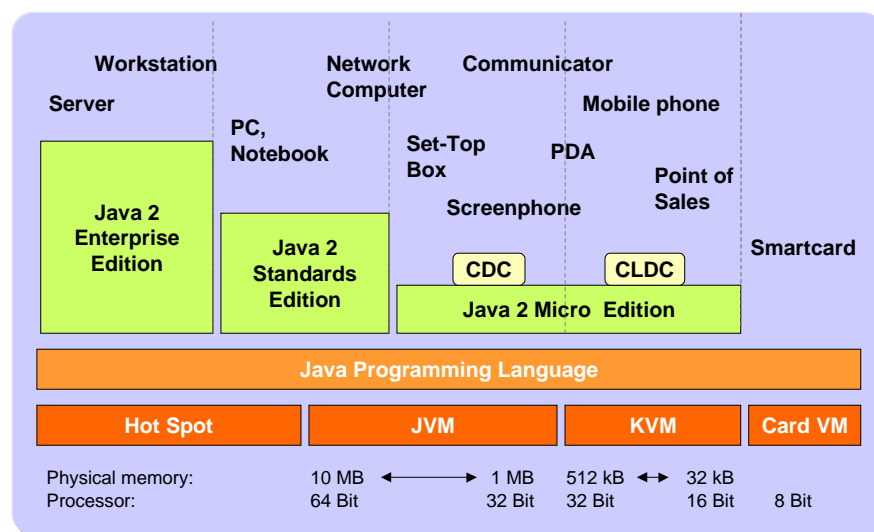


Abb. 4 Java Releases und Zielsysteme

Die Minimalausstattung und Funktionalität für neue Geräteklassen oder neue Klassen von Anwendungen werden in sogenannten Konfigurationen und Profilen spezifiziert. Neben der Portabilität von Anwendungen erlau-

ben die so geschaffenen Standards die Erstellung von Applikationen unabhängig vom Hersteller des Endgerätes und unabhängig vom Netzbetreiber.

J2ME Configuration - Hierunter versteht man eine für eine vorgegebene Geräteklasse universell verfügbaren, minimalen Funktionsumfang. Die Configuration spezifiziert die von der Geräteklasse unterstützten Sprach-elemente, die passende Java Virtual Machine, API und die zugehörigen Programm-bibliotheken (Class Libraries).

J2ME Profile - Ein Profil spezifiziert weitere Eigenschaften einer Geräteklasse oder eines bestimmten Types von Anwendungen für eine vorgegebene J2ME Configuration. Das Profil enthält weitere API mit den zugehörigen Objektklassen.

Eine Besonderheit des J2ME ist, dass nicht nur Bytecode auf die Geräte geladen werden kann, sondern auch Programm-bibliotheken nachgeladen werden können. Dadurch werden Endgeräte dynamisch konfigurierbar. Ein weiterer Gesichtspunkt sind Sicherheitsaspekte. Auf das Endgerät geladener fremder Bytecode wird vor der Ausführung nach syntaktischen und semantischen Regeln überprüft. Kritische Systemfunktionen sind geschützt bzw. stehen dem Programmierer nicht frei zur Verfügung.

CDC - Connected Device Configuration

Ein Connected Device ist ein Gerät der Leistungsfähigkeit einer digitalen Set-Top-Box oder eines PDA (Personal Digital Assistant, ein leistungsfähigerer Organisier), bzw. Elektronik in Fahrzeugen. Solche Geräte besitzen typischerweise einen 32-Bit Prozessor oder 64-Bit Prozessor und über 2 MBytes Arbeitsspeicher. Allerdings fehlt meistens eine vollständige Desktop-Benutzeroberfläche mit Fenstern und Mausbedienung. Stattdessen gibt es eine konventionelle Menüführung beispielsweise mit Stift bzw. mit einigen Softkeys.

Die Connected Device Configuration enthält eine voll ausgestattete Java Virtual Machine mit der Bezeichnung CVM. Ein passendes Profil zur Connected Device Configuration ist das sogenannte Foundation Profile. Das Foundation Profile unterstützt fast den gesamten Funktionsumfang von J2SE. Allerdings gibt es einige Anpassungen und gerätebedingte Einschränkungen beispielsweise bei den Benutzeroberflächen.

CLDC - Connected Limited Device Configuration

Ein Connected Limited Device ist typischerweise ein Mobiltelefon. Die Limitierung besteht hier in der stark eingeschränkten Displayfläche, einem kleinen Arbeitsspeicher zwischen 128 und 512 kBytes, und eingeschränkter Prozessorleistung der 16 Bit oder 32 Bit CPU. Persönliche, mobile Endgeräte mit Akkubetrieb sind die Regel. Ausser Mobiltelefonen und elektronischem Spielzeug kommen grundsätzlich aber auch Haushaltsgeräte in Frage, z.B. für Telemetrieapplikationen wie die Meldung des Systemzustandes für die Störungsdiagnose an den Kundendienst. Ein etwas exotischeres Beispiel wäre die Waschmaschine mit Fernwartung und ladbarem Waschprogramm.

Für Connected Limited Devices wurde eine eigene minimale Virtual Machine geschaffen, die sogenannte KVM. Der Buchstabe K in KVM deutet auf die Einheit Kilo hin. Damit soll angedeutet werden, dass der erforderliche Arbeitsspeicher für die Virtual Machine in Kilobytes gemessen wird, und nicht in Megabytes. Für die Virtual Machine selbst werden ca. 40 bis 80 kBytes benötigt. Ein typisches CLDC verfügt über insgesamt ca. 256 kBytes Arbeitsspeicher. Um eine solch kleine Virtual Machine möglich zu machen, mussten die Sicherheitsprozeduren für die Überprüfung von geladenem Bytecode in einer Weise vereinfacht werden, dass zusätzlich eine Prüfung vor dem Laden des Bytecodes erfolgt (sogenannte Prä-Verifizierung des Programmcodes als weiterer Schritt zwischen Compiler und Laufzeitumgebung).

Der für den Anwendungsentwickler wichtigste Teil einer CLDC-Umgebung ist das Mobile Interconnected Device Profile, oder kurz MIDP. Das Mobile Interconnected Device Profil beschreibt typischerweise ein Mobiltelefon, auf dem Informationsdienste, interaktive Dienste oder Spiele angeboten werden. Die grösste Einschränkung beim MIDP ist die Benutzeroberfläche. Für die Darstellung fordert MIDP eine Displaygrösse von mindestens 96x54 Bildpunkten mit 1 Bit Farbtiefe (also einem monochromen Bildschirm). Auf solch einem Display erfolgt die Darstellung bildschirmorientiert ohne eine Fenster-Umgebung. Scrollen erfolgt grundsätzlich nur vertikal. Die Benutzerführung erfolgt durch eine Menüsteuerung. Das MIDP API stellt dafür Menülisten, Textboxen, Alarmboxen oder Formulare für Kombinationen von Graphiken, Text, editierbaren Text oder Auswahllisten mit Radio-Buttons und Checklisten zur Verfügung. Für mit

üppiger ausgestattete Displays lassen sich so vor allen Dinge Spiele realisieren. Wer es kennt, erinnert sich bei diesem Funktionsumfang an die "Space Invaders" und die sonstigen Spielhallenklassiker aus den 80-er Jahren. Allerdings kommt beim Mobiltelefon die Möglichkeit zum Laden von Spielen und die Möglichkeit zur Vernetzung mit Freunden und Bekannten dazu.

Als Eingabemedium akzeptiert das MIDP entweder eine Standard Telefontastatur (12 Tasten mit Zahlen von 0 bis 10 plus * und #) bzw. eine Schreibmaschinentastatur, wie sie beispielsweise zu einigen PDAs oder von grösseren Mobiltelefonen angeboten wird. Je nach Hersteller stellt das spezifische Endgerät zusätzliche programmierbare Tasten, Rädchen oder Joysticks zur Verfügung. Das MIDP API gestattet es, solche gegebenenfalls vorhandenen programmierbare Tasten in die Benutzerführung zu übernehmen. Dazu stellt das API sogenannte Abstract Commands zur Verfügung für Funktionen wie "weiter", "zurück", "clear", "cancel", "Hilfe" usw. Diese Funktionen werden dann herstellerspezifisch den vorhandenen Bedienelementen zugeordnet. Übersteigt der Funktionsumfang der Applikation die Möglichkeiten des Endgerätes, erfolgt eine Umsetzung in eine Menüauswahl mit den Bezeichnungen (Labels) der gewünschten Funktionen.

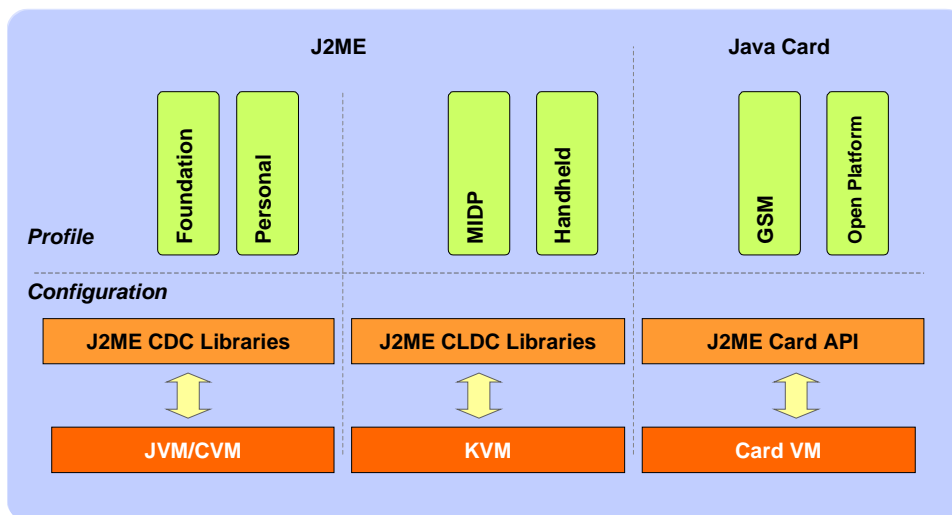


Abb. 5 J2ME Profile und Konfigurationen

J2ME Wireless Toolkit

Ein interessantes Werkzeug für den praktischen Umgang mit Java Applikationen auf mobilen Endgeräten ist das Java Wireless Toolkit. Das Toolkit findet sich zum Download auf den Java-Seiten von Sun Microsystems und erlaubt die Emulation von Midlets auf unterschiedlich ausgestatteten Mobiltelefonen und PDAs, beispielsweise Mobiltelefone mit monochromen Display, Farbdisplay, verallgemeinerten PDAs, aber auch Emulatoren für ein Java-fähiges Mobiltelefon von Motorola und den Palm in unterschiedlichen Ausprägungen. Das Wireless Toolkit enthält einige

Beispiele für Applikationen, z.B. Spiele, die man auf den verschiedenen Emulatoren ablaufen lassen kann.



Abb. 6 Java Wireless Toolkit

Download von Midlets

Bereits heute werden eine Menge an Midlets zum Download angeboten, vor allem Spiele und Bildschirmschoner. Die Möglichkeiten gehen sicherlich aber noch weiter als das Laden oder Versenden von Spielen oder Grusskarten. Bei mobilen Endgeräten besteht grundsätzlich die Möglichkeit, spontan mit Bekannten und Freunden zu spielen, die ebenfalls im Netz sind. Sei es Schach oder Dame, bzw. die moderneren Varianten von Strategie und Action. Kombiniert mit einem Dienst, der zeigt,

wer aus dem Bekanntenkreis bzw. wer aus der näheren Umgebung gerade online verfügbar sind, ergeben sich viele interessante Kandidaten für Anwendungen. Man darf gespannt sein, was unsere Lebensgewohnheiten für mobile interaktive Anwendungen noch alles hergeben werden. Ob die Sprachverbindung dabei über die Datenverbindung geht, oder bei realistischer Betrachtung über eine längere Zeit im etablierten Weg über GSM verbleibt, ist für neue Anwendungen relativ unerheblich. Bereits mit GPRS lassen sich Interaktionen über den Datenkanal abwickeln, und Telefongespräche bei Bedarf über GSM dazu schalten. Wegen der auf absehbare Zeit mangelnden Fächendeckung wird das auch bei UMTS eine ganze Weile so bleiben.

Im Zusammenhang mit den in den vorausgegangenen Abschnitten beschriebenen Methoden gibt es zwei Möglichkeiten, Midlets auf das Mobiltelefon zu laden bzw. für den Download verfügbar zu machen. In beiden Fällen wird das Midlet auf einem Web-Server zur Verfügung gestellt. Die reguläre Variante benutzt dazu HTML-Seiten mit einem Download-Bereich. Das Midlet lässt sich nun über einen regulären Web-Browser in einen PC laden. Dort kann es innerhalb eines Wireless Toolkits in einer Emulationsumgebung abgespielt werden. Besitzt man ein Java-fähiges Mobiltelefon mit Anbindung an den PC, kann man die Applikation auch vom PC aus via Datenverbindung über Infrarot, Bluetooth oder serielles Kabel auf das Mobiltelefon spielen. Die spannendere Variante ist das Laden von Anwendungen von unterwegs direkt über die Luftschnittstelle. Der Ablauf ist grundsätzlich der Gleiche. Das Midlet wird auf einer WML-Seite bereitgestellt (WML entspricht HTML bei WAP). Auf der WML-Seite wählt man die gewünschte Anwendung aus und startet den Ladevorgang.

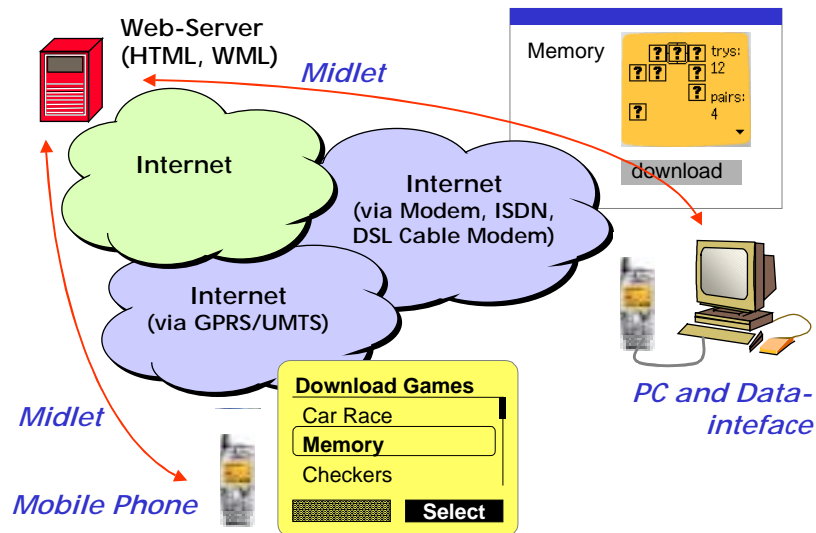


Abb. 7 Download Prozedur für Midlets

Java TV und MHP - Anwendungen für digitale Set-Top-Boxen

Zur Programmierung von Anwendungen wie interaktives Fernsehen, Video-on-Demand, elektronischen Programmführern gibt es ebenfalls bereits verfügbare API. Die Java TV API stellt zusätzliche Funktionen bereit für Audio und Video Streaming, Entschlüsselung der Zugangskontrolle (den sogenannten Conditional Access), Zugang zu Datenkanälen innerhalb und ausserhalb der Nutzkanäle, Zugang zu Informationen über die angebotenen Dienste, die Steuerung des Tuners für den Kanalwechsel (das sogenannte Zappen), der Steuerung der grafischen Darstellung am Bildschirm, sowie die Synchronisation von Audio und Video mit interakti-

ven Menüs. Java als Basis für Set-Top-Boxen hat sich in verschiedenen Ausprägungen etabliert. Die Cable Labs benutzen Java als Basis ihrer Open Cable Application Platform (OCAP). Die DVB Fraktion (Digital Video Braodcast) definieren Java als Basis ihrer Multimedia Home Platform (MHP).

Java TV stellt für die auf die Set-Top-Box ladbaren Applikationen eine spezielle Umgebung bereit stellt. Applikationen, die diese Umgebung nutzen, bezeichnet man als Xlets. Die Umgebung selber bezeichnet man als Xlet Life Cycle Manager. Xlets in der Set-Top-Box können unterschiedliche Zustände besitzen, nämlich gerade geladen worden sein, aktiv sein, Pause machen oder gelöscht sein. Unter dem Xlet Life Cycle Manager versteht man eine Finite State Machine, die die Zustände der Xlets verwaltet und eine Kommunikation dieser Zustände mit der Umgebung des Xlets herstellt.

Mit der fortschreitenden Digitalisierung der Medien und der Übertragungswege für das Fernsehen darf man von einer wachsenden Verbreitung digitaler Set-Top-Boxen ausgehen. Die Digitalisierung der Übertragungswege hat den entscheidenden Vorteil, dass innerhalb des Kanalrasters für einen analogen Fernsehkanal 6 digitale Fernsehkanäle übertragen werden können. Digitale Satellitentransponder bzw. das digitale terrestrische Fernsehen (DVB-T) bieten also wesentlich mehr Kapazität bzw. ein wesentlich wirtschaftlicheres Transportmedium. Das gilt ebenso für die Fernseekabelnetze (CaTV), die zusätzlich Datendienste (schnelles Internet) und interaktive Dienste bieten können, wenn man sie mit einem Rückkanal ausstattet. Zu den Vorteilen des digitalen Mediums selber gehören die bessere Bildqualität und bessere Tonqualität (z.B. Dolby Digital), wie sie von der DVD bereits bestens bekannt ist.

JAIN: Objekte für SIP und sonstige Protokolle der Telekommunikation

JAIN ist ein Paket, das speziell für die Programmierung von Diensten in Telekommunikation geschaffen wurde. Ursprünglich stand der Begriff für Java Architecture for Intelligent Networks. Die Architektur Intelligenter Netze stand auch Modell für die Architektur von JAIN, ebenso wie die Architektur von GSM, die selber eine Spezialisierung Intelligenter Netze für mobile Teilnehmer darstellt. Intelligente Netze ermöglichen die Real-

sierung von neuen Diensten in existierender Netzinfrastruktur, ohne das in den vielen Knoten des Netzes für jeden neuen Dienst ein Software-Update erforderlich wäre. Dazu wird im Netz ein spezielles Kommunikationsprotokoll implementiert (das sogenannte Intelligent Network Application Protocol INAP). GSM-Netze verwenden ebenfalls ein spezielles Kommunikationsprotokoll zur Unterstützung der Teilnehmermobilität (das sogenannte Mobile Application Protocol MAP). Mit der aufkommenden multimedialen Kommunikation über Datennetze entstehen weitere Protokolle, wie beispielsweise SIP [6].

JAIN bildet einen Überbau über alle diese Kommunikationsprotokolle, inklusive weiterer klassischer und neuer Protokolle (wie beispielsweise TCAP, ISUP, H.323 und MGCP). Ein Überblick über die Netzarchitektur zur Realisierung von Diensten finden sich in Kapitel 3 dieses Buches. Eine Übersicht über die Software Architektur von JAIN findet sich in [4] bzw. in [7]. In der Terminologie des JAIN API unterscheidet man grundsätzlich folgende Komponenten: den Protokollstack (INAP, MAP, SIP etc.), den Stack (als Abstraktion der Funktionen des Protokollstacks), den Provider (eine für Anwendungen benötigte Abbildung des Protokollstacks), sowie den Listener (als Basis für die Implementierung portabler Applikationen auf Stack und Provider). Listener und Provider kommunizieren über Ereignisse (Events) miteinander. Den Ereignissen sind Benutzertransaktionen zugeordnet (z.B. Signalling Point Codes und Subsystem Nummern für TCAP Transaktionen).

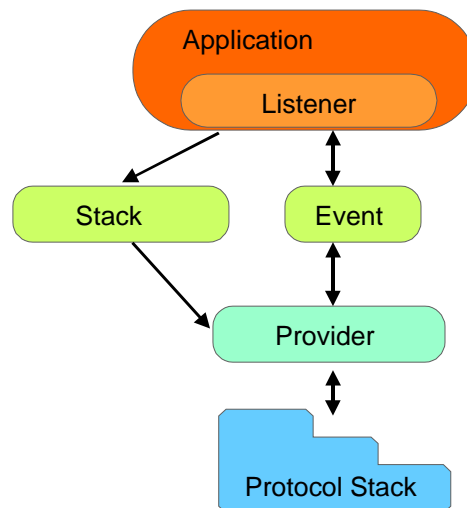


Abb. 8 JAIN Software Architektur

Zur Unterstützung SIP basierender Anwendungen finden sich in der JANI API ein Interface als Basis für herstellerspezifische Implementierungen des SIP Stacks (JainSipStack), ein Interface zur Implementierung der Provider Komponente (JainSipProvider), sowie ein Interface zur Implementierung SIP basierender Anwendungen (JainSipListener). Weiterhin finden sich zur Unterstützung einer Implementierung Standardfunktionen für SIP messages und header.

Anwendungsbeispiel: Mobile Controlled Media

Was kann man mit Midlets anstellen, ausser Spiele und Bildschirm-schoner zu laden? Man könnte beispielsweise mit dem Mobiltelefon Vide-

os oder Webseiten zum Anschauen auf dem Fernsehbildschirm auswählen. Solch eine Anwendung erscheint ungewöhnlich, da das Mobiltelefon quasi als Fernbedienung eingesetzt wird. Möglicherweise wären solche Anwendungen aber durchaus interessant, beispielsweise für Pay-Per-View Angebote. Das Mobiltelefon ermöglicht eine starke Authentisierung des Teilnehmers und kann grundsätzlich das Inkasso übernehmen, die Bedienung vereinfachen und unter Umständen den Jugendschutz verbessern. Für interaktive Programme bietet es eine gewohnte Umgebung für die Kommunikation per SMS, Browser und Sprache.

Ein anderer Aspekt wäre ganz einfach die Ergänzung des Mobiltelefons um ein hochwertiges Display. Interessante Angebote im Web, die man unterwegs identifiziert, könnte man sich zu Hause am Fernseher oder PC näher anschauen. Dazu braucht dann keine URL abgelesen und an einer Tastatur eingegeben werden, die Bedienung erfolgt gleich per Mobiltelefon. Unter Umständen lassen sich auf diese Art auch Mobiltelefone für vernetzte Spiele ohne Spielkonsole nutzen. Voraussetzung wären natürlich preisgünstige Angebote für eine mobile Vernetzung, beispielsweise durch einen günstigen Tarif für geringe Datenvolumen über GPRS bzw. die Verfügbarkeit von Bluetooth im Mobiltelefon als Brücke zu einem LAN (über DSL oder Kabelmodem) bzw. zu einem ISDN NT (über den D-Kanal).

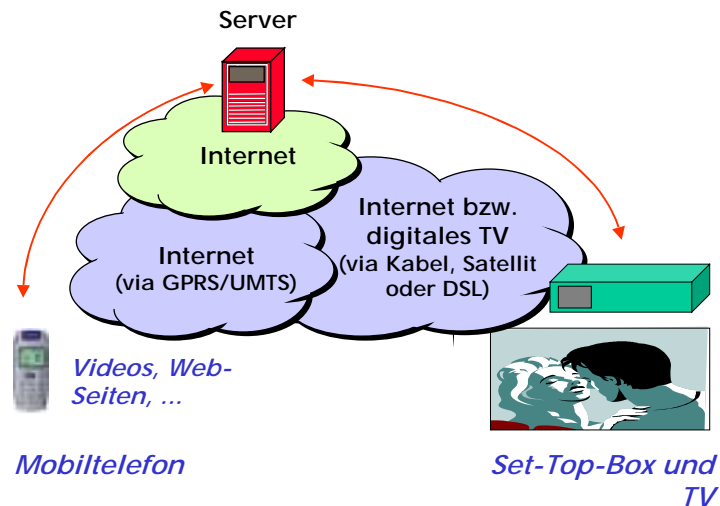


Abb. 9 Mobile Controlled Media

Unabhängig davon, ob solche Anwendungen tatsächlich einmal genutzt werden, sollen an einem einfachen Beispiel die technischen Möglichkeiten aufgezeigt werden. Voraussetzung dabei sind Java fähige Endgeräte. Im Beispiel sind die Aufgaben auf folgende Weise verteilt. Auf das Mobiltelefon lässt sich ein Midlet laden, das die Auswahl von Videos ermöglicht. Das Midlet kommuniziert mit einem Server, der den ausgewählten Inhalt bereitstellt. Das ausgewählte Video wird mit Hilfe einer digitalen Set-Top-Box abgespielt. Mit Hilfe des Mobiltelefons lässt sich der Film neu starten bzw. auf einen anderen Film umschalten.

Die Kommunikation zwischen Mobiltelefon und Server erfolgt im einfachsten Fall über HTTP mit Hilfe von Servlets, die im Server ablaufen.

Die Spezifikation des MIDP definiert HTTP1.1 als Minimalanforderung für die Vernetzung, wobei als Basis für HTTP entweder TCP/IP oder WAP in Frage kommen. Im einfachsten Fall ist die Set-Top-Box ebenfalls Java fähig, besitzt einen Browser und kann also mit dem Server ebenfalls per HTTP kommunizieren. Diese Voraussetzung gilt natürlich streng genommen nur, wenn für die Set-Top-Box ein Rückkanal vorhanden ist (über das Telefonnetz, bzw. über DSL oder ein interaktives Kabelnetz). Man kann sich aber auch vorstellen, dass man den interaktiven Anteil der Kommunikation im Netz terminiert und nur den ausgewählten Inhalt über eine Videokodierung in einem digitalen Broadcastkanal sendet. In diesem Falle wäre dann kein Rückkanal erforderlich.

Bei der Kommunikation über HTTP ist die völlige Transparenz der beteiligten Netze bemerkenswert. Das Mobiltelefon ist über einen Mobilnetzbetreiber angebunden. Der Server kann wiederum ein Gateway oder Media Control Point eines Internet Service Providers sein, der seinerseits an weitere Content Server angebunden ist. Das Abspielen der ausgewählten Information kann über ein digitales Fernsehnetz über Satellit, CaTV oder DSL über ein Telefonnetz erfolgen. Alle diese Abhängigkeiten sind oberhalb des IP-Protokolles bzw. ab HTTP nicht sichtbar.

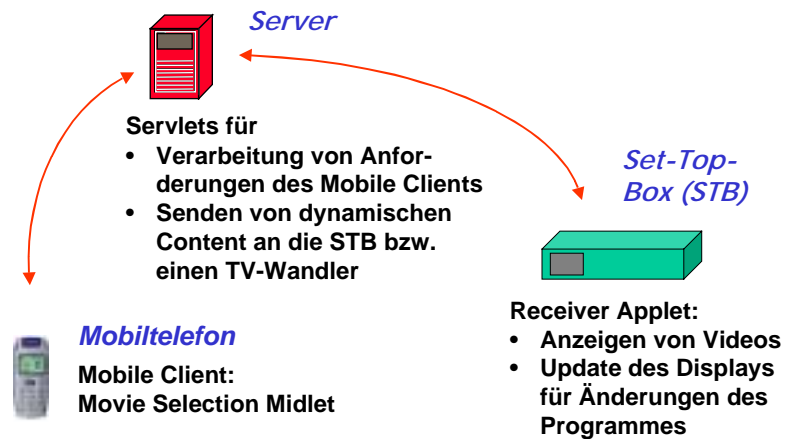


Abb. 10 Java basierende Realisierung

Im Anwendungsbeispiel ist die Kommunikation basierend auf HTTP. Dazu einige Erläuterungen im Zusammenhang mit MIDP und Servlets. Zur Kommunikation besitzt das MIDP ein Interface namens `HttpConnection` (das seinerseits eine Erweiterung des Typen `Stream Connection` und `Content Connection` des `Generic Connection Frameworks` ist). `HttpConnection` beinhaltet alle benötigten Komponenten für den Verbindungsaufbau zum Server, den Austausch von Informationen während der Verbindung, sowie für den Abbau der Verbindung. Eine Verbindung wird durch Angabe einer URL in einer Anweisung `"Connector.open"` eröffnet, die eine `HttpConnection` erzeugt. Für die Kommunikation stehen `InputStreams` und `OutputStreams` zur Verfügung, die gezielte Abfrage von Parametern via

"get"-Anweisungen (getURL, getHost, getPort, getFile, ...), sowie die aus HTML bekannte Übergabe von Parametern an Umgebungsvariable mit Hilfe der Methoden GET und POST.

Zur Kommunikation mit dem Midlet über HTTP stellt der Server ein HTTP-Gateway bereit. Im Beispiel werden dazu Servlets verwendet, d.h. Java Applikationen, die auf dem Server ablaufen und auf die Kommunikation mit Clients spezialisiert sind. Servlets sind Exemplare von Klassen, die das Interface `javax.servlet.*` implementieren. Obwohl Servlets grundsätzlich nicht an ein spezielles Kommunikationsprotokoll gebunden sind, werden sie gewöhnlich im Zusammenhang mit HTTP verwendet im Sinne von HTTP-Servlets. Nachdem das Servlet initialisiert ist, werden seine Dienste mit Hilfe der Anweisungen `ServletRequest` und `ServletResponse` beansprucht. Zur Anforderung einer URL mit Hilfe eines Browsers wird die Methode GET verwendet, bzw. beim Verschicken von HTML-Formularen die bekannten Methoden GET oder POST. Nach diesen grundsätzlichen Betrachtungen über die für das Anwendungsbeispiel verwendeten Komponenten folgen nun einige Erläuterungen über die Funktionen der Applikationen im Server und in der Set-Top-Box.

Wie realisiert man nun die Anwendung "Mobile Controlled Media" auf dem Server? Im einfachsten Fall durch zwei Servlets, von denen eines die Anfragen des Mobiltelefones bedient, und das andere die gewünschten Programme für die Anwendung auf der Set-Top-Box bereitstellt. Das erste Servlet, in Abbildung 11 als `MIDServlet` bezeichnet, wählt auf Anfrage des mobilen Clients den gewünschten Film aus, spielt ihn auf dem Server ab und gibt die Anfrage an das zweite Servlet weiter.

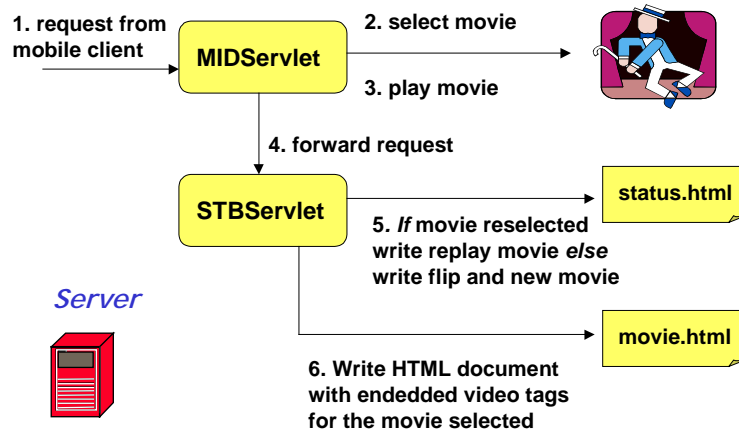


Abb. 11 Funktionen der Servlets

Da wir in diesem einfachen Beispiel auf der Set-Top-Box ein Applet vorausgesetzt haben, erfolgt die Kommunikation des STBServlets mit der Set-Top-Box als dynamischer HTML-Inhalt. Die folgenden HTML-Inhalte werden übertragen. Die Information über den Status der Anfrage erfolgt über eine Seite (status.html), die entweder ein Kommando zum Umschalten des Programmes enthält (flip), bzw. ein Kommando zum erneuten Abspielen des Programmes (replay). Die Koordinaten des Programmes werden in einer zweiten Seite übertragen (movie.html), und zwar einfach als eingebettetes Video Tag (`<HTML> ... <EMBED movie1 ...> ...</HTML>`).

Die übermittelten Informationen muss nun das Receiver Applet auf der Set-Top-Box noch richtig interpretieren. Da vom Server jeweils immer nur ein gültiges Programm übergeben wird, besteht die Funktion des Receiver Applets im wesentlichen darin, als Reaktion auf eine geänderte Statusinformation entweder die Wiedergabe neu zu starten oder auf ein neues Programm umzuschalten. Bei der Implementierung als Applet wird für die Wiedergabe der unterschiedlichen Medien auf den Browser zurückgegriffen, der diese Funktionen des Browsers für das Surfen im Web bereitstellt. Auf diese Weise bleibt die Anwendung in unserem Beispiel sehr einfach.

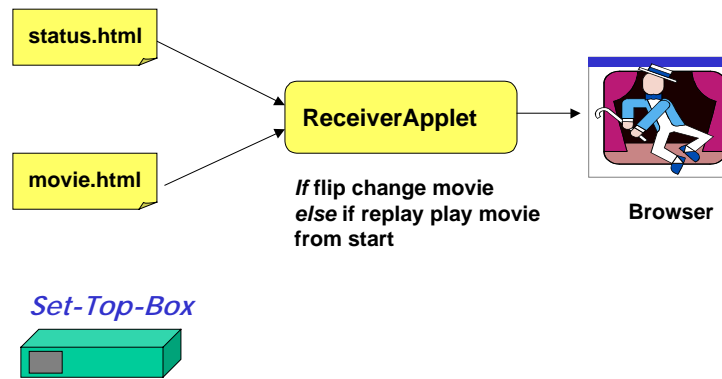


Abb. 12 Funktion des Receiver-Applets

Das Beispiel demonstriert den minimal erforderlichen Aufwand für die Kommunikation zwischen ganz unterschiedlichen Endgeräten und Netzen.

Für eine reale Anwendung gibt es natürlich noch viele Möglichkeiten zur weiteren Ausgestaltung. Während bisher das Mobiltelefon an das Mobilnetz gebunden war, und die Set-Top-Box an ein TV-Netz, ergibt eine Vernetzung mit sonstigen verfügbaren Anwendungen ein völlig neues Spiel. Die Verbindung des Mobiltelefones mit Web-Angeboten und multimedialen Anwendungen ist noch überschaubar einfach. Das Beispiel verwendet auch noch keine multimedialen Kommunikationssessions, die mit Hilfe des SIP-Protokolles initiiert werden. Man kann sich aber leicht vorstellen, dass sich Kommunikationsdienste leicht in interaktive Angebote für digitales Fernsehen oder Spiele einbauen lassen. Die Methode zur Realisierung ist grundsätzlich gleich.

Ausblick

Java eignet sich als Baukasten und Methode für universelle Systembeschreibungen hervorragend für die Anwendungsprogrammierung neuer multimedialer Kommunikationsdienste. Das Konzept geht jedoch über den Baukasten und über die Methode hinaus. Als Bestandteil einer Architektur zur Realisierung in den Netzen verteilter Anwendungen gewinnt Java bzw. der objektorientierte Ansatz eine wesentliche Bedeutung für künftige Telekommunikationsdienste und die dafür benötigte Infrastruktur. Als Beispiel für Netzarchitekturen sei die Initiative Sun One (Open Network Environment) von Sun Microsystems bzw. die .NET Architektur von Microsoft genannt. Als Verallgemeinerung von HTML gewinnen dabei speziell XML-basierende Systemschnittstellen an Bedeutung, die sich wiederum objektorientiert beschreiben und realisieren lassen.

Die mit Hilfe dieser neuen Systemschnittstellen gewonnene Flexibilität führt mittelfristig zu einem Redesign der Funktionen der Server und Systeme im Netz. Davon betroffen sind auch die klassischen Systeme der Telekommunikation, die z.B. durch die Auslagerung und Zentralisierung der teilnehmerbezogener Daten einerseits an Flexibilität für neue Anwendungen gewinnen andererseits an Komplexität und den damit verbundenen Kosten verlieren.

Auch der klassische Dienst der Telekommunikation, nämlich der Telefondienst, wird sich verändern. Der grundsätzliche Bedarf bleibt hier sicherlich unverändert: Nichts ist zielstrebiges als der individuelle Kontakt mit dem jeweils gewünschten Ansprechpartner. Dieser Kontakt muss auf

die einfachste und zuverlässigste Weise hergestellt werden. Für die technische Realisierung bieten sich allerdings neue und zielstrebigere Methoden. Basierend auf dem Signalisierungssystem SIP bieten neue Endgeräte und die bereits vorhandenen PCs raffiniertere Methoden für die individuelle und zielstrebige Ansprache und eine bessere Integration der Telefondienstes in die Anwendungen für die jeweiligen Alltagssituation. Man denke an die Verfügbarkeit eines Telefonverzeichnisses als Alternative zum Nachschlagen im Telefonbuch oder den Weg zur Telefonzelle, bzw. eine kurze Nachricht mit Empfangsbestätigung als Alternative zum laut-hals geführten Telefongespräch im Zug. Die Methoden zur Realisierung solcher Anwendungen entsprechen den hier beschriebenen Methoden.

Literaturhinweise

- [1] Java: Programmierhandbuch und Referenz für die Java 2 Plattform; Einführung und Kernpakete, Stefan Middendorf und Reiner Singer, dpunkt.Verlag, Heidelberg, 1999, ISBN 3-920993-82-9
- [2] Java Community Process, <http://jcp.org>
- [3] Java 2 Plattform, Micro Environment, <http://java.sun.com/j2me>
- [4] Referenz zur JAIN SIP API Spezifikation: <http://java.sun.com/aboutJava/cummuntyprocess/review/jsr032/index.html>
- [5] R. Riggs, A. Taivalsaari, M. VandenBrinck, Programming Wireless Devices with the Java 2 Platform, Micro Edition, Addison - Wesley Pub Co, 2001, ISBN 0-201-74627-1
- [6] S. Rupp, G. Siegmund, W. Lautenschlager, SIP - Session Initiation Protocol, dPunkt Verlag, Heidelberg, 2002, ISBN 3-89864-167-8
- [7] T.C. Jepsen (Hrsg.), Java in Telecommunications: Solutions for Next Generation Networks, Wiley, 2001, ISBN 0-471-49826-2