1

La piattaforma Java ME

Introduzione

Java Micro Edition (di seguito Java ME o JME) è la specializzazione di Java (linguaggio, virtual machine e librerie) per lo sviluppo di applicazioni per telefoni cellulari, computer palmari, sistemi di intrattenimento ed elettrodomestici evoluti. Questi oggetti sono talvolta chiamati "sistemi embedded" oppure "mobile devices" o, alcune volte, "Internet appliances". Si tratta di minuscoli calcolatori specializzati per un piccolo insieme di funzioni e realizzati con componentistica a basso consumo e capace di prestazioni modeste (se paragonate ai PC domestici). È un mercato vivace e in continua crescita, in cui si moltiplicano i processori, i sistemi operativi, le interfacce di comunicazione e le condizioni operative. Uno scenario ben diverso da quello PC, in cui le architetture hardware sono ormai consolidate e i sistemi operativi più diffusi (Windows, Mac OS X e Linux) rappresentano una base certa su cui sviluppare le applicazioni. Per questi sistemi sono disponibili Java Enterprise Edition (Java EE o JEE) e Java Standard Edition (Java SE o JSE), orientate rispettivamente alle soluzioni server-side ad alte prestazioni e alle applicazioni per PC desktop. Java EE e Java SE possono essere considerate piattaforme monolitiche: a meno di particolari API opzionali, esse forniscono tutti gli strumenti di cui lo sviluppatore ha bisogno. Inoltre, con il succedersi delle versioni, entrambe hanno arricchito la propria libreria standard dotandosi di funzionalità in precedenza delegate a pacchetti separati. Due piattaforme monolitiche, appunto, poiché monolitico, in un certo senso, e prevedibile è l'ambiente di esecuzione per il quale sono progettate. Nel settore dei sistemi embedded massima flessibilità e capacità di adattamento al singolo prodotto sono requisiti fondamentali per il contenimento dei costi. Tuttavia tale flessibilità è attuabile solo in fase di progettazione: una volta che il prodotto è stato realizzato e installato (o, più semplicemente, venduto) l'eventuale aggiornamento, o la modifica di un qualsiasi componente hardware o software, avrebbe costi superiori ad un prodotto nuovo funzionalmente equivalente. Non più, dunque, un approccio monolitico ma un paniere di componenti, API, librerie specializzate da cui attingere per costruire un prodotto specifico. Un modello di cellulare potrebbe avere il modulo Bluetooth ma non supporto per il multimedia, mentre un altro potrebbe avere un accelerometro e un lettore di RFID ma essere privo di interfaccia grafica.

Si è dunque reso necessario progettare una nuova piattaforma che fosse modulare, flessibile, portabile ed economica da realizzare. Java ME, appunto, è piattaforma disegnata ex novo e in maniera indipendente, per sottolinearne il ruolo fondamentale per lo sviluppo e l'esecuzione delle applicazioni per sistemi mobili: non una semplice "versione ridotta" della Standard Edition, bensì una architettura appositamente progettata per risolvere le problematiche di questo particolarissimo settore applicativo. La piattaforma Java ME è organizzata in virtual machine (indispensabile per l'esecuzione del bytecode Java), configurazioni, profili ed API opzionali, che opportunamente combinate consentono di realizzare una vastissima gamma di prodotti e servizi applicativi. Prima di entrare nel vivo di questa affascinante tecnologia, è opportuno ricordare che la tecnologia Java include anche la piattaforma JavaCard, specializzata per la

scrittura di applicazioni eseguite all'interno dei microprocessori delle smart card. JavaCard è la massima espressione della portabilità di Java, su processori a 8bit e memorie di pochi KB.

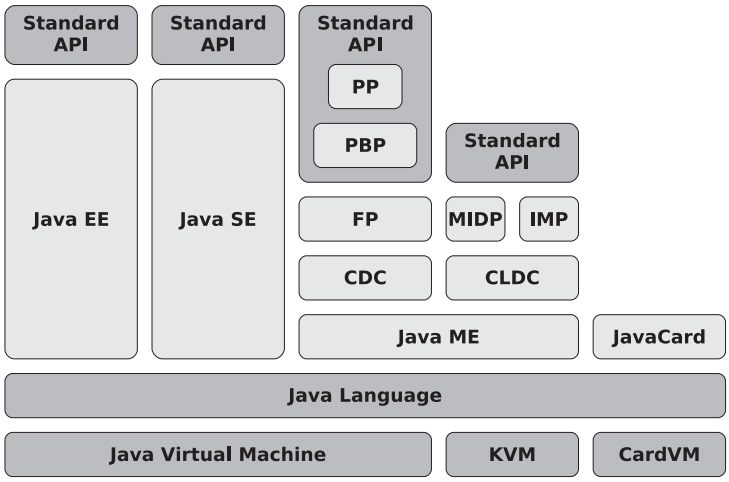

Figura 1.1 - La piattaforma Java suddivisa in Enterprise Edition, Standard Edition, Micro Edition e JavaCard.

Il Java Community Process

La suddivisione della piattaforma Java in edizioni (Enterprise, Standard e Micro) ha segnato la nascita di un processo partecipativo di definizione e pubblicazione delle specifiche del linguaggio e delle API standard. Il Java Community Process (JCP) è, dunque, il luogo privilegiato in cui aziende ed esperti, riuniti in sessioni di lavoro chiamati Expert Group, valutano la necessità di standardizzare un certo contesto applicativo e provvedono alla redazione di una specifica, la Java Specification Request (JSR). Nel mondo Java ME, le JSR seguono progressivamente l'evoluzione dei dispositivi e, dunque, soddisfano l'esigenza di disporre di opportune API per consentire alle applicazioni Java di accedere alle nuove funzionalità dell'hardware. Per ogni Expert Group un Specification Leader coordina l'attività di specifica; come detto, il gruppo è composto da aziende ed esperti del settore, scelti per l'interesse specifico nel dominio applicativo oggetto della specifica. Sun Microsystems, i maggiori produttori di terminali telefonici e gli operatori delle reti di telefonia mobile partecipano a buona parte degli Expert Group. Esistono JSR dedicate alla piattaforma vera e propria (configurazioni e profili), altre destinate ad API opzionali (ad esempio la Wireless Messaging API) oppure di armonizzazione delle API già rilasciate (ad esempio, la Java Technology for Wireless Industry). Quando, dunque, si nomina una JSR ci si riferisce al documento di specifica formalizzato e rilasciato ufficialmente dal JCP. Tali documenti sono descrivono **API standard** (cioè standardizzate, frutto di un lavoro partecipativo e condiviso), in contrapposizione a quelle **proprietarie**, definite arbitrariamente da un produttore o un operatore. Le API rilasciate dal JCP vengono chiamate sia "API standard" sia "API opzionali": con la prima accezione si intende sottolineare che si tratta di specifiche condivise, accettate collegialmente; con la seconda si intende sottolineare che esse descrivono funzionalità opzionali, non necessariamente presenti nel runtime Java di un dispositivo. In questo libro i termini "API standard" e "API opzionali" saranno utilizzati sempre per indicare specifiche del JCP (con le due accezioni appena descritte), mentre con il termine "API proprietarie" saranno indicate le soluzioni dei singoli produttori.

Virtual Machine, Configurazioni e Profili

L'architettura modulare di Java ME consiste nella suddivisione della piattaforma in tre componenti essenziali: virtual machine, configurazione e profilo. Qualsiasi implementazione Java ME possiede questi tre componenti, scelti tra diverse versioni disponibili. La virtual machine è il componente software che esegue le applicazioni Java ME, traducendone il bytecode nelle corrispondenti istruzioni del processore del dispositivo (esistono diverse tipologie di "traduzione", dalla semplice esecuzione passo dopo passo di ogni singola istruzione, alla traduzione completa del bytecode in codice nativo). La virtual machine è responsabile del caricamento codice, dell'isolamento del runtime Java dal resto del sistema operativo, della gestione della memoria, dei thread e delle altre risorse fondamentali per l'esecuzione delle applicazioni. A seconda della potenza dell'hardware disponibile, le risorse e le prestazioni fornite dalla virtual machine variano notevolmente (ad esempio nel numero massimo di thread gestibili oppure nella efficienza del garbage collector responsabile della periodica rimozione di oggetti in memoria non più utilizzati). La configurazione fornisce le funzionalità di base garantite dalla virtual machine e stabilisce il sottoinsieme del linguaggio supportato dall'ambiente di esecuzione. In altre parole, la configurazione costituisce l'astrazione "in linguaggio Java" delle funzionalità di base fornite dal microprocessore e dai sistemi di interfacciamento verso le periferiche. Ad esempio, la configurazione determina la presenza del supporto all'aritmetica in virgola mobile, il modello di multithreading, l'eventuale framework di connettività e di accesso alle funzioni di I/O di basso livello. Essa non costituisce un ambiente di esecuzione completo, poiché non fornisce il modello applicativo (cioè i meccanismi con cui le applicazioni sono eseguite, terminate o sospese durante l'esecuzione) né l'interfaccia utente. In altri termini, si può affermare che la configurazione "mostra" alle applicazioni Java ciò che la virtual machine sottostante può fare; di fatto virtual machine e configurazione possono considerarsi inscindibili. Il profilo completa la configurazione definendo il modello applicativo, le classi per l'interfaccia utente (non necessariamente basata su componenti grafici) e le altre funzionalità che rendono l'ambiente Java ME pronto per l'esecuzione di una applicazione. Una medesima configurazione può avere più profili: ad esempio, per la Connected, Limited Device Configuration (CLDC) sono stati definiti il Mobile Infomation Device Profile (MIDP), presente nella quasi totalità dei telefoni cellulari, e l'Information Module Profile (IMP), privo di interfaccia grafica e destinato al controllo remoto di apparecchiature come macchine per la distribuzione delle bibite o servocomandi. Pur con una forte semplificazione, virtual machine, configurazioni e profili possono essere associati a tre fasi successive della progettazione di un dispositivo. La virtual machine è fortemente legata al tipo di processore utilizzato: processori di fascia bassa forniranno prestazioni ridotte e, di conseguenza, la virtual machine sarà piuttosto semplice. La configurazione può essere pensata come la modellazione della motherboard del dispositivo: essa, infatti, stabilisce quali sono le funzionalità di basso livello e, mantenendo questa metafora, quali sono i componenti essenziali che circondano il processore. Il profilo, infine, definisce con buona approssimazione il dispositivo vero e proprio, ovvero la specializzazione di una determinata board (la configurazione) basata su un certo processore (la virtual machine), per diventare un prodotto ben definito. È il profilo, dunque, che determina se una scheda diventerà la base di un navigatore satellitare oppure un PDA. Come si vedrà in seguito, il primo prodotto potrebbe essere facilmente realizzato con la configurazione CDC e il Personal Basis Profile, mentre il secondo, mantenendo la stessa configurazione, potrebbe essere implementato con il Personal Profile: stessa configurazione ma profilo differente, dunque due prodotti sostanzialmente diversi. Come detto nel paragrafo di apertura, virtual machine, configurazione e profilo provengono da una architettura squisitamente modulare ma, una volta definiti, costituiscono un elemento monolitico: attualmente lo sviluppatore non ha la possibilità di scegliere la combinazione dei tre componenti separatamente, ma deve adattarsi a ciò che è fornito (o disponibile) per il dispositivo che intende utilizzare. Nella telefonia cellulare, il runtime Java ME è fornito come parte integrante del dispositivo: non è possibile rimuoverlo né modificarne la configurazione. Sui PDA si ha maggiore flessibilità, poiché l'utente ha la possibilità di scegliere quale runtime installare (si vedano le sezioni dedicate a Windows Mobile e Palm OS nel capitolo 3); per contro, però, raramente un runtime generico (pensato, cioè, per essere installato su dispositivi simili ma non identici, ad esempio tutti i palmari dotati di Microsoft Windows Mobile 5.0) riesce a sfruttare al massimo le funzio-

nalità dell'hardware e del sistema operativo. Attualmente sono disponibili due configurazioni destinate rispettivamente a telefoni cellulari consumer (e altri dispositivi con caratteristiche hardware limitate) e a smartphone, palmari o altri apparati con prestazioni superiori:

- Connected, Limited Device Configuration (CLDC)
- Connected Device Configuration (CDC)

Per ciascuna configurazione sono disponibili più profili applicativi:

- CLDC
  - Mobile Infomation Device Profile (MIDP)
  - Information Module Profile (IMP)
- CDC
  - Foundation Profile (FP)
  - Personal Basis Profile (PBP)
  - Personal Profile (PP)

Come si è già visto, la combinazione configurazione più profilo definisce un contesto applicativo preciso e fornisce l'ambiente al quale il programmatore deve riferirsi durante la progettazione e lo sviluppo di una applicazione. Se in ambiente desktop la prima scelta determinante (e talvolta vincolante) è il sistema operativo, in ambito mobile è il profilo a costituire il riferimento (e dunque il vincolo) durante la realizzazione di un progetto. La piattaforma Java è intrinsecamente multipiattaforma, ma l'insieme di librerie e funzionalità definite da un profilo potrebbero essere completamente assenti in un altro; ad esempio, effettuare il porting di una applicazione per telefono cellulare su un set-top box per la televisione interattiva può richiedere la totale riprogettazione e implementazione del codice scritto.

Dispositivi entry-level e consumer

Il primo rilascio ufficiale della piattaforma Java ME è avvenuto nel 2001 e da allora questa tecnologia è cresciuta costantemente diventando leader assoluto del mercato della telefonia mobile. L'80% dei nuovi telefoni è equipaggiato con un runtime Java ME e il numero di dispositivi equipaggiati con Java si attesta intorno ai 2 miliardi. Sono numeri importanti che dimostrano quanto fosse strategico proporre in quegli anni una tecnologia portabile, semplice ed flessibile nel mercato della telefonia mobile. Il successo di Java ME è dipeso proprio dall'enorme diffusione dei telefoni cellulari, in particolare di quelli di fascia entry-level, cioè economici e dunque accessibili all'utenza non professionale. Il libro è interamente dedicato a CLDC e MIDP, perciò nei paragrafi seguenti sarà offerta solo una panoramica di questa coppia configurazione/profilo.

Connected, Limited Device Configuration

La Connected, Limited Device Configuration (CLDC, JSR 30) è la configurazione più semplice della piattaforma Java ME. È stata progettata per equipaggiare dispositivi con caratteristiche hardware molto modeste; i requisiti minimi sono infatti processori a 16 o 32bit, memoria complessiva inferiore ai 512KB, connettività ridotta. La virtual machine su cui è basata questa configurazione non è conforme alla specifica standard, ma è stata opportunamente semplificata per consentirne l'esecuzione anche su processori con limitate capacità di elaborazione. In particolare, il processo di verifica del bytecode, che nella piattaforma standard è eseguito dalla virtual machine prima dell'esecuzione dell'applicazione, è anticipato nella fase di sviluppo ed è immediatamente successiva alla compilazione: il codice per il dispositivo,

dunque, è "già verificato" e include marcatori che segnalano alla virtual machine l'avvenuto controllo. La virtual machine su cui si basa CLDC ha ulteriori limitazioni: non supporta la Java Native Interface (JNI), né la *reflection* e la *serializzazione* degli oggetti. CLDC contiene un sottoinsieme minimale della libreria Java standard, allineata alla versione 1.1. In particolare, sono inclusi i package `java.lang`, `java.io` e `java.util`, le cui classi sono solo una parte molto modesta di quelle disponibili nella piattaforma, mentre altre, ad esempio la classe Thread, sono state notevolmente semplificate. Priva del supporto all'aritmetica in virgola mobile, CLDC 1.0 ha introdotto il *Generic Connection Framework* (GCF), un efficace infrastruttura per la gestione dei servizi di connettività locale e remota in grado di supportare numerosi protocolli con una interfaccia unificata (il GCF è descritto dettagliatamente nel capitolo 5). La versione 1.1 di questa configurazione ha introdotto il supporto all'aritmetica in virgola mobile attraverso i tipi primitivi `float` e `double`, oltre alle relative classi wrapper `Float` e `Double` e opportune estensioni alle classi di utilità. L'implementazione CLDC HI (Hotspot Implementation) non ha aggiunto funzionalità alla configurazione ma ne ha incrementato le prestazioni in maniera significativa al fine di trarre beneficio dai nuovi processori dei dispositivi mobili, in particolare dalle architetture a 32bit e dalla maggiore quantità di memoria a disposizione. CLDC è alla base del MIDP e IMP, due profili applicativi per telefoni cellulari e sistemi di telecontrollo.

TINI e altri moduli embedded

Nel 2000 Dallas Semiconductor realizzò TINI (TIny Network Interface), un modulo programmabile in Java delle dimensioni di un banco di memoria SIMM a 72 pin. TINI è stato il primo esempio di utilizzo della tecnologia Java per i sistemi embedded e, grazie ad una serie di *demoboard* dotate di numerose possibilità di interfacciamento (seriali, Ethernet, 1-Wire), ha consentito la realizzazione di diversi progetti studio su robotica e *ubiquitous computing*. Dopo la pubblicazione della specifica Java ME e il coinvolgimento di vari produttori indipendenti, la piattaforma TINI si è evoluta in termini di prestazioni e di varietà dell'offerta di moduli. Oggi aziende quali MAXIM, Imsys e Systronix producono e distribuiscono moduli programmabili basati su CLDC e su classi proprietarie per l'accesso alle periferiche e alle linee di I/O. TSTIK, JStik TINI, SNAP, JStamp sono alcuni dei prodotti più interessanti, che offrono a sviluppatori e hobbisti la possibilità di sperimentare l'uso di Java su sistemi embedded e realizzare prototipi di "oggetti intelligenti" programmabili, sfruttando esperienza, documentazione ed esempi di codice del mondo Java ME.

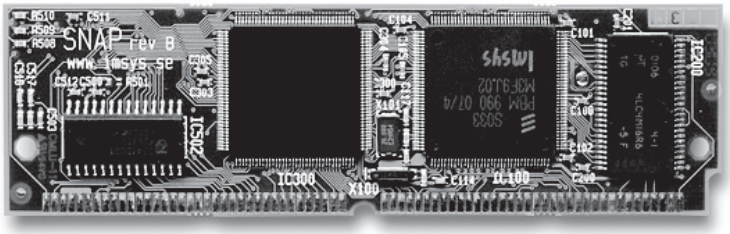

Figura 1.2 – Il modulo SNAP, prodotto dalla svedese Imsys.

Mobile Information Device Profile

MIDP è il Profilo applicativo di Java ME più diffuso nel mercato dei telefoni cellulari programmabili e dei PDA. I primi cellulari compatibili furono i Motorola iDEN i50sx e i80, destinati al mercato americano e commercializzati nell'aprile del 2001. In Italia il primo cellulare compatibile fu il Nokia 7650, che introdusse per la prima volta la fotocamera e la possibilità di inviare immagini e foto via MMS (funzionalità, però, precluse al runtime Java ME!). MIDP ha introdotto il modello applicativo basato su MIDlet (simile alle Applet), nel quale le applicazioni sono gestite da un Application Manager che ne governa l'installazione, l'esecuzione e l'eventuale rimozione. Questo profilo estende il GCF con il supporto alla

connettività via HTTP e fornisce un servizio elementare di persistenza dei dati chiamato *Record Management System* (RMS). Parte fondamentale di MIDP è la libreria LCDUI, che comprende un insieme di componenti grafici per la creazione di interfacce utente per dispositivi con piccoli display ed interazione via softbutton o touchscreen. La versione 1.0 (JSR 37) era molto semplice e priva di meccanismi di sicurezza particolari. La versione 2.0 (JSR 118), attualmente in uso, dispone invece di un sistema di gestione della sicurezza basato su permessi utente, certificati e firma digitale delle MIDlet, nonché della possibilità di restringere l'accesso alle funzionalità sensibili (ad esempio la rubrica) o soggetti a tariffazione (i servizi di rete) solo alle applicazioni autorizzate. MIDP 2.0 include anche un servizio di avvio automatico delle applicazioni installate al verificarsi di eventi particolari, come la ricezione di un SMS o lo scadere di un timer. La versione 3.0 (JSR 271) è attesa entro il 2007 e porterà significative migliorie a questo profilo, in particolare nella gestione delle interfacce utente, della comunicazione tra MIDlet e nell'integrazione con i servizi di base del dispositivo.

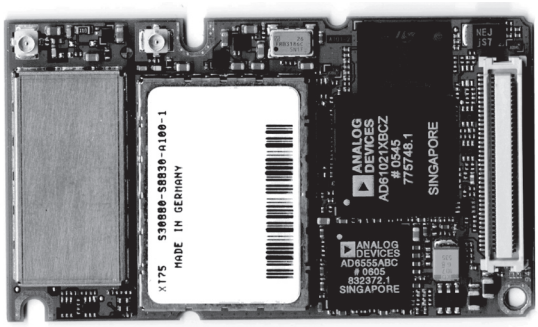

Figura 1.3 – Il modulo Siemens XT75 dotato di runtime Java ME, profilo IMP-NG e Location API.

Information Module Profile

La configurazione CLDC si presta bene ad equipaggiare anche apparati diversi dai telefoni cellulari, quali sistemi di controllo remoto, piccoli robot, macchine per la vendita di bibite e snack, sistemi di automazione industriale. Parte integrante di MIDP è la libreria grafica LCDUI: questo profilo, dunque, è espressamente disegnato per una interfaccia a display con la quale l'utente interagisce. Questa libreria, però, è del tutto inutile in un sistema di telecontrollo, eventualmente non presidiato. L'Information Module Profile (IMP, JSR 195) è definito "headless" (letteramente "decapitato", "senza testa") poiché non dispone di una interfaccia utente. IMP è progettato a partire da MIDP e le sue applicazioni, benché dette IMlet, sono tecnicamente vere e proprie MIDlet. Questo profilo eredita anche il GCF e il RMS, ed esiste in due versioni, la più recente è chiamata IMP-NG, Infomation Module Profile Next Generation (JSR 228). Oltre alle eventuali API standard, i produttori dei moduli embedded aggiungono solitamente estensioni proprietarie per accedere a funzionalità dedicate, come ingressi e uscite analogiche e digitali, accesso alle routine di bootstrap e gestione delle applicazioni. Siemens e Motorola producono moduli dotati di connettività GPRS (o superiore) e runtime Java ME integrato. La casa tedesca dispone del catalogo più ricco; tra i moduli prodotti è particolarmente interessante il modello denominato XT75, visibile in Figura 1.3. Questo modulo è dotato di connettività EDGE e di ricevitore GPS; l'ambiente di esecuzione Java è basato su CLDC 1.1, IMP-NG e include la Location API (JSR 179) per l'accesso alle funzionalità di localizzazione.

Sun SPOT

Una variante di IMP è installata sui moduli Sun SPOT (Small Programmable Object Technology), sviluppati da Sun Microsystems per la realizzazione di reti wireless di sensori. La tecnologia Sun SPOT consente la creazione di un vero e proprio laboratorio programmabile, in cui una "nuvola" di sensori di

diverso tipo possono scambiarsi informazioni, inviare dati ad un PC, controllare attuatori di vario tipo (motori, relè...) oppure semplicemente restare dormienti sino al verificarsi di un determinato evento.

Figura 1.4 – Un free-range Sun SPOT, dotato di sensor board e batteria.

Il runtime degli SPOT si basa sulla Squawk VM, virtual machine particolarmente ottimizzata in termini di prestazioni e basso consumo energetico, e su un sottoinsieme di MIDP arricchito da una vasta libreria di classi per il controllo del modulo radio, dei sensori e del sistema di I/O. La dotazione di sensori e la possibilità di interfacciamento con qualsiasi componente elettronico aggiuntivo è l'aspetto più interessante di questa tecnologia. Ogni SPOT, infatti, può essere dotato di una *daughterboard* specializzata. In particolare, nel kit di sviluppo è presente una sensor board dotata di un accelerometro su tre assi, sensori di luminosità e temperatura, una serie di LED tricolori e due switch, una serie di ingressi e uscite analogiche e digitali. I Sun SPOT sono equipaggiati con un modulo radio conforme allo standard IEEE 802.15.4, che opera in uno spettro di frequenza compreso tra 2.4GHz e 2.4835GHz. In questo senso, Sun SPOT condivide parte della specifica di ZigBee, che però comprende uno stack più sofisticato e completo. La codifica è di tipo DSSS (Direct Sequence Spread Spectrum), la stessa utilizzata per lo standard WiFi. Dal punto di vista hardware, infine, ciascun modulo è più piccolo di un terzo rispetto ad un comune cellulare, è alimentato da una piccola batteria agli ioni di litio ed è dotato di una porta mini-USB per la connessione al PC che provvede anche alla ricarica della batteria.

Dispositivi hi-end

CLDC, MIDP e IMP costituiscono la porzione della piattaforma Java ME per i telefoni cellulari e computer palmari di fascia economica. Oltre a questo tipo di dispositivi, il mercato offre architetture hardware e sistemi operativi ben più potenti, progettati per equipaggiare terminali mobili ed embedded di alto profilo (hi-end), quali computer palmari evoluti, set-top box per la TV interattiva, chioschi informativi, sistemi di home-theater e multimedia, navigatori satellitari per auto. La specializzazione Java per il mercato embedded è iniziata proprio su dispositivi di questo tipo, in particolare con la tecnologia PersonalJava che fu progettata per i primi PDA Compaq iPAQ (basati su Windows CE) e Sharp Zaurus (basati su Linux). PersonalJava era essenzialmente una rivisitazione di Java 1.1 con un meccanismo di sicurezza più robusto mediato da Java 1.2. La questa tecnologia non prese piede (probabilmente per una immaturità del mercato in sé), mentre divenne elemento fondante di JavaTV e, successivamente, della Multimedia Home Platform (MHP), destinata ad equipaggiare i set-top box per la televisione interattiva satellitare e terrestre. PersonalJava, però, era ancora una volta un componente monolitico e difficilmente

poteva adattarsi a diversi contesti applicativi. L'avvento di Java ME ha portato ad una architettura modulare e ciò che inizialmente era PersonalJava è stato suddiviso in una configurazione e tre profili. Meno diffusi della coppia CLDC/MIDP, questi componenti costituiscono un ambiente di esecuzione affidabile e flessibile per PDA di fascia alta, smartphone e altri dispositivi con potenza di calcolo sufficientemente elevata da eseguire una virtual machine standard.

Connected Device Configuration

La Connected Device Configuration (CDC, JSR 36 per la versione 1.0 e JSR 139 per la versione 1.1) è stata progettata per equipaggiare dispositivi embedded hi-end, come quelli appena elencati. Requisiti minimi per la CDC definiti al momento del rilascio della specifica erano: 512KB ROM, 256KB RAM, connettività cablata o wireless, implementazione di una Java Virtual Machine conforme alla specifica standard (cioè la stessa della piattaforma desktop). Le quantità di memoria richieste dalla specifica iniziale sono ormai ampiamente superate da qualsiasi telefono cellulare di recente progettazione, anche di fascia economica. L'introduzione di funzionalità multimediali, come fotocamera e player audio mp3, ha inevitabilmente portato ad un incremento delle prestazioni dell'hardware dei dispositivi consumer; occorre tenere presente che la configurazione CDC non è mirata al settore della telefonia cellulare, ha uno spettro di applicazione molto ampio. Come si vedrà in seguito, per la CDC sono stati definiti tre profili, che rappresentano livelli crescenti di complessità e ricchezza della libreria. In ambito CLDC si è visto che MIDP e IMP sono mutuamente esclusivi: la sovrapposizione avviene tra una configurazione ed un singolo profilo. Nel mondo basato su CDC, invece, ogni profilo può essere ulteriormente specializzato da livelli successivi che introducono nuove funzionalità. È il caso del Personal Profile (PP), che estende il Personal Basis Profile (PBP); questo, a sua volta, si basa interamente sul Foundation Profile (FP) che rappresenta il setup funzionale più semplice. Questo modello "a scatole cinesi" permette a ciascun profilo di ereditare le caratteristiche di quelli sottostanti e ai produttori di runtime Java ME di gestire in modo più efficiente il rilascio di dispositivi di complessità crescenti. I tre profili rendono disponibili tre modelli applicativi, ovvero tre modalità diverse di avvio e interruzione delle applicazioni e della relativa interazione con il runtime Java. La configurazione CDC fornisce il sottoinsieme minimale della piattaforma standard e in particolare i package `java.lang`, `java.lang.ref` e `java.lang.reflect`, `java.io`, `java.net`, `java.security` e `java.security.cert`, `java.text`, `java.util`, `java.util.jar` e `java.util.zip`. CDC si caratterizza come appartenente alla piattaforma Java ME grazie alla presenza del Generic Connection Framework; è incluso, infatti, il package `javax.microedition.io` che comprende la classe Connector ed alcune interfacce `Connection` del GCF. La specifica sottolinea il fatto che, benché si tratti di un design omologo al framework definito in ambito CLDC, occorre valutare (e testare) con cautela il porting si porzioni di codice scritte per il GCF di CLDC su CDC poiché non è assicurato che il comportamento sia identico tra le due configurazioni. Come già accennato, CDC rispetta le specifiche della virtual machine standard ed include perciò il supporto JNI, che consente di estendere le funzionalità del runtime Java attraverso moduli nativi a caricamento dinamico (DLL in ambiente Windows, Share Objects in ambiente Linux) scritti in C/C++. Rispetto a CLDC, questa possibilità amplia notevolmente le possibilità di sviluppo e di integrazione con l'hardware del dispositivo. Nella piattaforma entry-level, infatti, se una funzionalità nativa (Bluetooth, messaging, multimedia) non è implementata dal runtime in dotazione al dispositivo non c'è alcuna possibilità di estenderlo. Su CDC, viceversa, qualsiasi API fornita dal sistema operativo ed eventualmente non implementata dal runtime Java può essere interfacciata attraverso JNI. Su Windows Mobile, ad esempio, per il runtime WEME di IBM sono state sviluppate estensioni per la libreria grafica SWT e l'accesso alle funzioni multimedia e database del sistema operativo.

Prima di conoscere la dotazione di ciascun profilo, è importante sottolineare che CDC, FP, PBP e PP sono stati rilasciati in due versioni, 1.0 e 1.1, che forniscono rispettivamente compatibilità con le versioni 1.3 e 1.4 della piattaforma standard. Rispetto alla coppia CLDC/MIDP, in cui la scelta delle versioni di configurazione e profilo può essere fatta in maniera indipendente (ad esempio, MIDP 2.0 può essere associato a CLDC 1.0 e a CLDC 1.1), nel caso di CDC la versione attribuita alla configurazione vincola la corrispondente versione del profilo. In altre parole, scegliendo, ad esempio, di utilizzare CDC 1.0

sarà necessario adottare la medesima versione per qualsiasi profilo superiore venga integrato con la configurazione. Tra gli smartphone attualmente dotati di CDC vi sono i Nokia 9300, 9300i e 9500 Communicator, i Sony-Ericsson P990, M600 e P1. CDC è disponibile come software opzionale per palmari equipaggiati con Microsoft Windows Mobile 2003 SE e 5.0.

Foundation Profile

Si è già detto più volte che il profilo specializza la configurazione per un ben preciso contesto applicativo. Il profilo aggiunge il modello di esecuzione delle applicazioni, l'interfaccia grafica e numerose utility di supporto. È, dunque, un componente piuttosto corposo, denso, ricco di funzionalità. Il JCP ha definito un profilo particolare, che potrebbe essere definito "sottile". Il Foundation Profile (FP, JSR 46 e JSR 219) è il primo strato software che rende la configurazione CDC immediatamente utilizzabile in un prodotto commerciale (o quasi…). Come l'IMP visto in precedenza, anche il Foundation Profile è detto "headless", poiché non include una libreria grafica, fornita invece dai profili superiori. Il Foundation Profile completa il set di classi di CDC ottenendo una copertura pressoché completa della piattaforma standard: sono fornite tutte le classi della gerarchia `java.security`, le classi per la scrittura di file compressi in formato ZIP e GZIP (mentre CDC supporta la lettura di file ZIP); vi sono, inoltre, numerose classi dedicate all'input/output e alla gestione delle connessioni di rete. Se si esclude la libreria di componenti grafici (assente, come detto), questo profilo è sostanzialmente completo per l'esecuzione di applicazioni standard. Il Foundation Profile definisce il modello applicativo "unmanaged", non gestito, cosiddetto perché le applicazioni sono eseguite a partire dall'invocazione del metodo statico `main()` e terminano all'uscita di quest'ultimo, se non vi sono altri thread attivi. Applicazioni di questo tipo sono dette anche stand-alone, cioè isolate, "per conto proprio", volendo indicare con questa espressione il fatto che esse, una volta avviate, gestiscono autonomamente la propria esecuzione, che termina senza possibilità di sospensione e successivo riavvio o negoziazione di risorse con la virtual machine. Come si vedrà nel seguito, Personal Basis Profile e Personal Profile definiscono modelli applicativi più complessi. Il Foundation Profile rappresenta un ottimo banco di prova per sperimentare il porting di un runtime Java ME verso nuove architetture e sistemi operativi. Ad esempio, non esistendo una implementazione CDC commerciale per la piattaforma Internet Tablet di Nokia, la community "Mobile and Embedded" lo ha scelto per iniziare il porting di phoneME per Nokia 770 (il progetto phoneME è descritto nel seguito di questo capitolo). L'interfaccia grafica è senza dubbio tra le parti più onerose da integrare; per questo che spesso l'attività di porting ne esclude inizialmente l'adattamento, al fine di svincolare lo sviluppo della virtual machine vera e propria (esecuzione bytecode, gestione della memoria, multithreading e accesso alle risorse del sistema operativo) dall'interfacciamento al motore grafico.

Personal Basis Profile

Il Personal Basis Profile (PBP, JSR 129 e JSR 217) completa il Foundation Profile con le classi essenziali di AWT per la realizzazione di componenti *lightweight* (letteralmente "leggeri"). La piattaforma Java ha introdotto due tipologie di componenti: i cosiddetti componenti *heavyweight* ("pesanti") e quelli *lightweight*, presenti in questo profilo. I primi sono componenti grafici direttamente mappati sui componenti nativi del sistema operativo (detti *peer*). Ad esempio, quando l'applicazione istanzia un pulsante o un campo di testo, questi vengono richiesti al sistema operativo, il quale li istanzia a sua volta attingendo dalla propria libreria grafica. I componenti , essendo gestiti direttamente dal sistema operativo, sono coerenti con l'aspetto grafico delle applicazioni native e, come queste ultime, hanno elevate prestazioni *rendering* (di disegno) della scena grafica. Viceversa, i componenti lightweight non dipendono dai componenti nativi e sono disegnati direttamente dall'applicazione Java, invocando le primitive grafiche della classe `Graphics`. Il codice Java, dunque, è indipendente dai componenti grafici del sistema operativo (che potrebbero essere del tutto assenti) e l'aspetto dell'applicazione non rispecchia il tema in uso. Lo svantaggio di questo tipo di componenti è nelle ridotte prestazioni durante la fase di disegno,

poiché il tempo necessario a disegnare tutta la scena grafica è superiore rispetto a quello richiesto dai componenti nativi. Ogni singola istruzione di disegno deve essere infatti tradotta dal bytecode Java alla corrispondente istruzione nativa, con inevitabile dilatazione dei tempi di esecuzione. L'uso del Personal Basis Profile è strategico in quei contesti in cui implementare una libreria grafica nativa sarebbe troppo oneroso rispetto al vantaggio della portabilità del codice Java, che consente di riciclare codice scritto per diverse piattaforma hardware senza necessità di ricompilazione (necessaria, invece, per eventuali librerie native). Inoltre, l'uso di componenti lightweight consente il massimo di personalizzazione dell'aspetto grafico in base ai requisiti del client. Questo profilo, infatti, è stato definito dopo il Foundation Profile e il Personal Profile: in pratica, il JCP ha constatato che un profilo intermedio tra i due avrebbe potuto semplificare l'adozione di CDC in particolari contesti applicativi. Il Personal Basis Profile introduce i package `java.awt`, `java.awt.event`, `java.awt.image` e `java.awt.color`; in particolare sono presenti `Component`, `Container`, `Graphics` e le altre classi fondamentali per la creazione di componenti grafici elementari. Il Personal Basis Profile è fondamentale nell'economia della piattaforma Java ME, poiché in esso è definito il modello applicativo basato su `Xlet`. Queste, inizialmente introdotte dall'API JavaTV, vengono gestite attraverso un `XletManager` (simile all'Application Manager già visto per il profilo MIDP). Ogni `Xlet` implementa l'interfaccia omonima ed è eseguita all'interno di un `XletContext`, che costituisce il punto di accesso alle risorse del sistema, come il `Container` grafico, il `ClassLoader` e la notifica del cambio di stato dell'applicazione. Le `Xlet` eseguite contemporaneamente sul dispositivo possono comunicare tra loro attraverso IXC, l'Inter-Xlet Communication Protocol.

Personal Profile

Il Personal Profile (PP, JSR 62 e JSR) realizza l'ambiente applicativo CDC più completo: oltre a quanto fornito da Foundation Profile e Personal Basis Profile, completa l'intero set di classi di AWT (compresi, dunque, i componenti *heavyweight*). Il Personal Profile include anche il supporto per Applet e accesso alla clipboard, consentendo il trasferimento dati tra le applicazioni Java ed altre native. È importante notare che il Personal Profile potrebbe **non includere** il Personal Basis Profile, ma possedere le classi di quest'ultimo (necessarie per AWT) senza garantire supporto al modello applicativo `Xlet`. I Nokia Series 80, ad esempio, offrono un runtime compatibile CDC e PP, ma modello applicativo stand-alone.

Convergenza CLDC e CDC

La maggior parte dei terminali in commercio supporta una sola configurazione (CLDC o CDC) e non è possibile sostituire il runtime Java preinstallato; eventualmente, l'unico aggiornamento consentito è fornito dal produttore del dispositivo e generalmente si tratta di versioni di consolidamento piuttosto che di integrazione di nuove funzionalità. Come già detto, CLDC è la configurazione più diffusa in assoluto e, dunque, MIDP è il profilo presente nella maggioranza dei telefoni cellulari e palmari. I modelli di fascia alta (smartphone o PDA-phone) in genere includono sia CLDC e CDC, rispettivamente con MIDP e PP. Esempi di questo tipo sono i Nokia 9300, 9300i e 9500 Communicator, i Sony-Ericsson P990, M600 e P1, già visti in precedenza. Sorprende il fatto che su questi dispositivi le due configurazioni facciano capo a runtime separati. Ne è prova il fatto che le API standard associate a CLDC non sono disponibili per CDC e viceversa. Questo crea talvolta non pochi problemi in fase di progettazione di una applicazione, poiché "sulla carta" un certo dispositivo supporta un insieme di API standard, mentre nella realtà questi risultano inaccessibili ad una delle due configurazioni. Unica eccezione in questo scenario fu il Jasper S20, il primo telefono dotato del sistema operativo Savaje OS, interamente basato su Java. Savaje OS disponeva di un unico runtime in grado di eseguire applicazioni CLDC e CDC, rispettivamente basate sui profili MIDP e PBP, con la possibilità per entrambi di accedere alle API opzionali. A fine 2006, Savaje ha dovuto fronteggiare gravi difficoltà finanziarie e nella primavera dell'anno successivo è stata acquisita da Sun Microsystems. Savaje OS è diventata la base per un progetto più ambizioso, chiamato JavaFX Mobile. Si tratta di un intero ambiente di esecuzione basato su un kernel Linux e equipaggiato

con un runtime Java ME compatibile CLDC/CDC. La novità rispetto a Savaje OS è appunto JavaFX, un motore di scripting che consente l'implementazione rapida di applicazioni e *widget*. L'intera interfaccia utente sarà personalizzabile in base alle esigenze degli operatori, ai quali la nuova piattaforma di Sun Microsystems si rivolge.

I progetti phoneME e phoneME Advanced

Nell'estate 2006 Sun Microsystems ha annunciato il rilascio di buona parte della piattaforma Java sotto licenza GPL. L'adozione della licenza opensource (più precisamente free software) ha progressivamente coinvolto diversi parti della piattaforma, inclusi gli ambienti di sviluppo, gli application server e, più recentemente, i runtime Java ME. I progetti phoneME e phoneME Advanced hanno come obiettivo fornire rispettivamente implementazioni di qualità di CLDC e CDC e relativi profili, con particolare attenzione ai requisiti di prestazioni, portabilità e compatibilità con le API standard. Come detto, si tratta di progetti opensource: chiunque ha la possibilità di scaricare il codice sorgente e intraprendere il porting per una specifica piattaforma. La progressiva diffusione di Linux come sistema operativo open per telefoni cellulari e il consolidamento di phoneME e phoneME Advanced porterà in breve tempo a disporre di ambienti per il mobile computing totalmente aperti e configurabili, di fatto svincolati dalle scelte (e, dunque, dai limiti) dei produttori di terminali.

Le API opzionali

Configurazioni e profili forniscono le funzionalità di base di un ambiente Java ME: essi permettono la realizzazione di applicazioni semplici, senza però avvantaggiarsi delle caratteristiche più interessanti fornite dal sistema operativo e dall'hardware. Ad esempio, la specifica del profilo MIDP 2.0 prevede il supporto obbligatorio ai protocolli HTTP e HTTPS, ma non permette l'invio e la ricezione di messaggi (SMS e MMS), la riproduzione di contenuti audio e video, l'accesso a periferiche Bluetooth; tutte funzionalità, quelle appena elencate, ormai presenti anche nei terminali consumer. Il JCP ha dunque provveduto a definire e pubblicare numerose JSR, ciascuna dedicata ad un particolare ambito applicativo. Vi sono API dedicate alle funzionalità di networking (Bluetooth, Web Services, SMS e MMS, RFID, SIP), altre dedicate al multimedia (registrazione e riproduzione di contenuti audio e video), grafica (2D, 3D), alla localizzazione e alla sicurezza. Il capitolo 11 offre una panoramica delle API opzionali già implementate nei terminali in commercio, con un breve cenno a quelle di futuro rilascio.

Device fragmentation

La disponibilità di diverse versioni di configurazioni e profili e di numerose API opzionali ha portato in breve tempo a disporre di una grande varietà di dispositivi, ciascuno caratterizzato da un ambiente Java ME "personalizzato". Non è raro trovare nel catalogo dei maggiori produttori dispositivi apparentemente molto simili (anche nell'aspetto esteriore) dotati di runtime Java ME leggermente differenti; talvolta si tratta di una intera API presente in un modello ed assente in un altro, mentre altre volte di tratta di porzioni di una stessa API che risultano mancanti. Esempi concreti sono la Bluetooth API (JSR 82) o la Web Services API (JSR 172). La prima è costituita da due moduli: un primo generico, fondamentale, costituito dalle classi essenziali per l'accesso allo stack Bluetooth (discovery dei dispositivi limitrofi e gestione delle connessioni), e un secondo modulo per il supporto del protocollo OBEX. I primi terminali Nokia equipaggiati con la JSR 82 supportavano solo il primo modulo (ad esempio, i best-seller 6600, 6630 e 9300). Con l'introduzione della Series 60 2nd Edition, Feature Pack 3 (cioè dal N70 in poi) la JSR 82 è stata implementata interamente. Lo scenario si complica ulteriormente allorché la libertà di implementazione non è a livello di modulo ma di sue singole funzionalità. Il caso più eclatante è rappresentato dalla Mobile Media API (MMAPI, JSR 135), che fornisce le classi per la riproduzione e l'acquisizione di contenuti audio e video: la capacità di decodificare certi formati, il supporto a risorse in streaming, la possibilità di riproduzione parziale dei contenuti (*progressive download*) non sono garantiti e la effettiva

compatibilità con la funzione richiesta deve essere verificata per ogni singolo dispositivo. Spesso, infatti, la documentazione fornita dal produttore non riporta tutti i dettagli sull'implementazione di ogni API. Il risultato di questa abbondanza di specifiche e della libertà dei produttori di implementarle a piacimento è l'impossibilità di prevedere il comportamento del runtime Java ME di un dispositivo e, allo stesso tempo, la necessità di predisporre versioni delle applicazioni adattate al singolo dispositivo. Questa condizione è detta *device fragmentation* (letteralmente "frammentazione dei dispositivi", intesa come entropia delle possibili combinazioni di API e implementazioni particolari di queste). Per ovviare a questo problema, gli ambienti di sviluppo (capitolo 3) offrono sistemi di gestione automatica di diverse versioni di codice specializzate per differenti famiglie di dispositivi; è evidente, però, che si tratta di soluzioni "artigianali" e la questione necessita di essere affrontata a livello di specifica e non di applicativi. Il JCP ha dunque definito tre specifiche che hanno l'obiettivo di armonizzare l'offerta degli ambienti Java ME dei dispositivi mobili, indicando un sottoinsieme minimo di API garantite e, per ciascuna di esse, il comportamento atteso dal runtime. La prima specifica è stata la Java Technology for Wireless Industry (JTWI, JSR 185), che considerava come requisiti fondamentali CLDC 1.0, MIDP 2.0, la Wireless Messaging API 1.0 (WMA, JSR 120) e la MMAPI. Il valore aggiunto della JTWI rispetto alla semplice enumerazione delle API è l'indicazione **esatta** del comportamento intrinseco di queste. Ad esempio:

- CLDC **deve** consentire almeno 10 thread simultanei;
- MIDP 2.0 **deve** supportare immagini in formato JPG (la specifica originale richiede solamente PNG) e il RMS **deve** poter gestire almeno 5 RecordStore;
- la WMA **deve** consentire l'avvio dell'applicazione alla ricezione di un SMS;
- la MMAPI **deve** supportare l'acquisizione di immagini in JPG e la possibilità di accedere a contenuti multimediali attraverso HTTP.

La specifica contiene numerosi altri parametri che limitano l'arbitrio di chi implementa il runtime Java ME. La JTWI è ormai datata e per risolvere il problema della device fragmentation al crescere delle API opzionali, il JCP ha rilasciato le specifiche Mobile Service Architecture e Mobile Service Architecture Advanced (MSA, rispettivamente JSR 248 e JSR 249 per CLDC e CDC) che estendono la JTWI includendo numerose API, alcune delle quali ben poco diffuse tra i terminali in commercio: paradossalmente, con la MSA le soluzioni arrivano prima che i problemi si manifestino. La MSA per CLDC, ad esempio, include ben 14 API opzionali, mentre un suo sottoinsieme (MSA Subset) ne include solo 6, ma di importanza strategica come la Bluetooth API o il supporto alla gestione di file SVG attraverso la JSR 226. Nell'autunno 2007 saranno finalmente disponibili i primi telefoni cellulari conformi alla JSR 248, tra cui il Nokia 6500, appartenente alla Series 40 5th Edition, e il Sony-Ericsson W910i; con il progressivo rinnovamento della gamma di modelli dei maggiori produttori, è auspicabile che entro 18-24 mesi tutti i telefoni di nuova commercializzazione siano basati sulla Mobile Service Architecture e che, dunque, si rafforzi la compatibilità delle applicazioni su runtime Java ME di produttori diversi.

Riferimenti

Java Micro Edition: http://java.sun.com/javame

Java Community Process: http://www.jcp.org

Maxim TINI: http://www.maxim-ic.com/products/microcontrollers/tini/

Systronix: http://www.systronix.com

Imsys: http://www.imsys.se

Progetto Sun SPOT: http://www.sunspotworld.com

Progetto phoneME: https://phoneme.dev.java.net/