COMPUTERWORLD

J2ME i forhold til J2SE og J2EE

I denne artikel vil jeg uddybe teknologien bag J2ME og sammenligne med J2SE og J2EE.

Skrevet den **07. Feb 2009** af **jonasbc** I kategorien **Programmering / Java** | ★★★☆☆

Generelt om Java
Bygger på mange områder på C++. Objektorienteret.
Oprindeligt udviklet til at styre forbruger-elektronik som fjernsyn, videoer og køleskabe.
Har udviklet sig til et moderne, platforms-uafhængigt højniveau programmeringssprog.
Fik flere og flere funktioner med i standard-bibliotekerne, og blev ved udgivelsen af Java 2 opdelt i forskellige udgaver, hver især målrettet til en specifik type opgaver. Er i øjeblikket opdelt i tre udgaver: J2SE, J2EE og senest J2ME. (Faktisk nævner Sun selv nogle steder i white papers og datasheets Java Cardä med den tilhørende Card VM som en fjerde udgave, men ifølge deres hjemmeside hører den endnu ikke for sig selv)

J2ME - Definition og funktionalitet
Sun tog initiativ til J2ME i 1998, og det blev første gang præsenteret til den årlige JavaOne konvention i juni 1999. De tilhørende API'er defineres løbende af en række virksomheder, der anvender teknologien, i samarbejde med Sun i den såkaldte Java Community Process (JCP).
Anvendelsesområdet er lidt diffust, men afgrænses i [kilde 1] til området af apparater mere avanceret end et chip kort og mindre avanceret end en Pc. Det vil sige stort set alt lige fra personsøgere og mobiltelefoner til avancerede set top bokse, der har næsten samme funktionalitet som en Pc.
Formålet med J2ME er, at det på en given enhed skal være muligt at anvende applikationer udviklet af andre and hardware leverandøren. Dette kan f.eks. være spil, kalendersystemer, software til en specifik brance som eksempelvis lagerstyring og meget, meget mere. Det gør det desuden også muligt at opgradere softwaren på en given enhed.

Krav
J2ME skal kunne køre på mange forskellige typer enheder, der er ofte meget begrænsede. I nogle tilfælde måles pladsen, der er til rådighed til applikationen, klasser og den virtuelle maskine i antal hundrede kb! Disse enheder kan være både faste og trådløse og har ofte en begrænset display størrelse samt begrænset mulighed for bruger input.
Configurations
Da pladsen som nævnt ofte er trang på de små enheder, er det ikke hensigtsmæssigt, at have en hel række standard-biblioteker liggende, som ikke bruges eller måske endda ikke understøttes af den enkelte enhed. Derfor har Sun valgt at opdele J2ME i forskellige configurations, der anvendes til enheder af samme "familie". En sådan configuration kan siges at være den "laveste fællesnævner" for en række enheder. Altså den funktionalitet, som en programmør kan være sikker på eksisterer. I øjeblikket findes to configurations: CDLC (Connected Limited Device Configuration) og CDC (Connected Device Configuration). Førstnævnte er, som navnet antyder, den mindste, og er designet til apparater med langsomme CPU'er og begrænset hukommelse - ofte mellem 128 og 512 kb. Dette kan f.eks. være mobiltelefoner, personsøgere og PDA'er.
CDC er tiltænkt mere avancerede apparater med mere processorkraft og hukommelse som f.eks. set top bokse, underholdnings-systemet til bilen og avancerede PDA'er. Disse enheder har oftest en 32 bit CPU og minimum 2 mb hukommelse til rådighed til Java platformen og applikationer. Disse enheder nærmer sig altså mere en normal Pc, og CDC indeholder en større del af J2SE end CLDC.

Profiles

For yderligere at tilpasse J2ME til en enkelt type af apparater, tilføjes ud over CDC eller CLDC yderligere funktionalitet i såkaldte profiles. Her defineres yderligere elementer som brugergrænseflade og adgang til egenskaber for en specifik type enhed.

Eksempel på en profil
Profilen til mobile enheder: Mobile Information Device Profile (MIDP)
Målrettet til:

- mindst 160 kb tilgængelig hukommelse til Java
- processor på mellem 8 og 32 MHz
- batteridrevet
- begrænset netværksforbindelse (9,6 kbps)
- begrænsede input muligheder (ITU-T keypad)
- begrænset brugergrænseflade (96x54 pixels)

Sammen med CLDC udgør MIDP et komplet JRE til bl.a. mobiltelefoner og PDA'er.

Den virtuelle maskine
CDLC
Til disse på flere områder begrænsede enheder har Sun udviklet den såkaldte K Virtual Machine (KVM). K'et står for "kilo", hvilket antyder, at hukommelsen på dette niveau måles i kilobytes og ikke megabytes som i Pc'ere.

I øjeblikket hænger KVM og CLDC meget tæt sammen, eftersom CLDC kun kan køre på KVM og KVM kun kan afvikle CLDC teknologi. I fremtiden er det dog meningen, at de to skal kunne henholdsvis afvikle og afvikles på andre lignende teknologier.

Design-målene for KVM var at lave den mindst mulige "komplette" Java virtuelle maskine, der på den ene side understøttede alle de centrale aspekter af Java sproget og på den anden side kunne afvikles på enheder med meget lidt hukommelse til rådighed. KVM er implementeret i C, hvilket gør det muligt at anvende den til en lang række forskellige enheder, da C stadig er det mest anvendte sprog i denne henseende. Sun har lavet reference implementationer til 3 forskellige operativsystemer: Solaris, Windows og Palm OS. Andre firmaer har lavet mere end 25 forskellige implementationer af KVM på andre platforme.

Den seneste tids udvikling f.eks. indenfor mobiltelefoner og PDA'er med farvedisplays og en række netværks- og multimedie-funktioner har medført, at der stilles endnu større krav til performance. Derfor er Sun i gang med at udvikle teknologien CLDC HotSpot (enkelte steder nævnt under kodenavnet monty), der performance-mæssigt giver en række fordele i forhold til KVM. I den traditionelle VM allokeres separat plads til objekter, data og "code cache" hvilket resulterer i unødig fragmentering. I CLDC HotSpot implementationen gemmes alt i objekt heap'en, hvorved fragmentering undgås og håndteringen af hukommelsen kan simplificeres.

CDC
Til de mere avancerede apparater anvendes ofte JVM, Java's Standard virtuelle maskine. For nylig er teknologien CLDC HotSpot, der performance-mæssigt giver en række fordele, dog vundet frem.

Sammenligning med J2SE og J2EE
J2SE
- standard edition

J2EE
- enterprise edition
- J2SE + forskellige muligheder på server-siden (Servlets, JEB)
- J2SE's compiler, men ofte med "-server parametren". Giver lidt længere startup tid, men grundet mere memory forbrug bedre performance, også via en mere grundig runtime analysator

J2ME - CLDC

Mål

At definere en standard Java platform for små, ressource-begrænsede enheder.

Omfatter ud over sproget, VM og Core Java Libraries (java.lang. og java.util.*)*

Input-output, netværk, sikkerhed, internationalization.Omfatter IKKE user interface, event handling og generelt interaktion mellem applikation og bruger. Dette varetages af én eller flere overliggende profiles.

Sikkerhed

Java klasser gennemgår en "verification". Dette foregår meget hurtigt, da hver klasse på kompilerings-tidspunktet får tilføjet en "stackmap" attribut. Forskellige applikationer er adskilt fra hinanden vha. et lukket "sandbox" miljø. Klasser i protected packages kan ikke overskrives af applikationer.

Garbage Collection

Garbage Collection eksisterer på de fleste implementationer. Den afvikles ikke automatisk, men med et kald til System.gc(). Den er kun i stand til at frigøre hukommelse fra ubrugte objekter (mark and sweep algoritme) - ikke at defragmentere. Derfor kan man let ende med at have for lidt hukommelse til rådighed, hvis man ikke tænker sig om.

Fravigelser fra Java Language Specification

Ingen understøttelse af datatyper flydende tal. Ingen "finalization" af klasse instanser. Begrænset fejl-håndtering.

Specifikt for denne udgave

Alle connections håndteres af javax.microedition.Connector og en række interfaces i same package.

Øvrigt

Når en applikation distribueres officielt, skal alle klasser indeholde "stackmap" attributten og være indkapslet i en JAR fil.

J2ME - CDC

Mål

At definere en standard Java platform for mellemstore enheder. Dvs. enheder "mellem" J2ME CLDC og J2SE/EE.

Omfatter ud over sproget, VM og Core Java Libraries (java.lang. og java.util.*)*

Forskellige packages fra J2SE: java.net.*, java.io.*, java.text.* og java.security.*. API'er er desuden "renset" - depricated metoder er fjernet. CDC er altså ikke fuldt bagud kompatibelt! Understøtter desuden alle J2SE 1.3 VM funktioner som sikkerhed, JNI , RMI og JVMDI .

Sikkerhed

Understøtter samme sikkerhedsfunktionalitet som J2SE.

Garbage Collection

Både garbage collection og synchonization håndtering som i J2SE. Desuden er interfaces mellem de forskellige komponenter som garbage collector og oversætter mere veldefinerede og dokumenterede end i forrige udgaver. Dette gør det lettere at udvikle nye komponenter.

Fravigelser fra Java Language Specification

Som nævnt er depricated elementer fjernet og enkelte steder er det foretaget ændringer i klasser, der før havde referencer til java.awt.*. Denne package er jo ikke nødvendigvis understøttet af en given profile.

Specifikt for denne udgave

Kræver ca. 40% mindre hukommelse til VM på grund af muligheden for at hente klasser fra ROM og fjernelsen af unødvendige funktioner.

Øvrigt
CDC er i stand til at udføre bytecode direkte fra ROM, hvilket betyder en bedre opstarts-tid og mindre fragmentering.

J2SE
Mål
At skabe er moderne, simpelt, objektorienteret og platforms-uafhængigt programmeringssprog, der kan anvendes til såvel simple applikationer som avancerede klient-server systemer.

Omfatter ud over sproget, VM og Core Java Libraries (java.lang. og java.util.*)*
Standard Java pakken med almindelige funktionaliteter som serialization, JNI, sikkerhed og grafisk brugergrænseflade.

Garbage Collection
Er efterhånden ved at være "voksen". I tidligere versioner var GC ikke tilstrækkelig, og risikoen for en out-of-memory-error var derfor større. Med nye teknologer som HotSpot er dette ved at være løst.

Øvrigt
fra ROM, hvilket betyder en bedre opstarts-tid og mindre fragmentering. Java er på trods af udviklingen til det bedre stadig et meget hukommelses-krævende sprog på grund af de mange runtime tjek. Nye teknologier er ikke altid bedre på det område. Eksempelvis er HotSpot til servere meget hukommelseskrævende.

J2EE
Mål
At skabe en overbygning til J2SE, der gør det muligt at udvikle stabile, hurtige og sikre server-centrerede applikationer.

Omfatter ud over sproget, VM og Core Java Libraries (java.lang. og java.util.*)*
Ud over den sædvanlige J2SE funktionalitet understøttelse af server-relevante funktioner som f.eks. JavaServer Pages (JSP), Servlets, Enterprise JavaBeans (EJB), J2EE Deployment API, JNDI , CORBA, og JDBC data access API.

Sikkerhed
Ud over de normale sikkerhedsrutiner som f.eks. class verification fokuseres i J2EE meget på adgangstilladelser. Når en klient eksempelvis ønsker at anvende en service på serveren, bliver klienten identificeret og får tildelt en credential - en nøgle, som giver adgang til forskellige dele af serveren. Dette giver f.eks. mulighed for at give en klient adgang alene til JSP sider og ikke til EJBs.

Garbage Collection
Anvender samme garbage collection som J2SE.

Øvrigt
J2SE's compiler anvendes, men ofte med "-server parametren". Giver lidt længere startup tid, men grundet mere memory forbrug bedre performance, også via en mere grundig runtime analysator

Konklusion
J2ME ser ud til at være vel gennemtænkt. Det virker fremtidsorienteret med en stabil arkitektur. Det, at man har bibeholdt den grundlæggende java-del gør det let at lære for eksisterende Java-programmører. Mange interessenter involveret i udviklingen af standarder. Sikrer at teknologien vil blive brugt fremover. Fornuftigt opdelt - de forskellige udgaver supplerer hinanden. Fantastisk hvor meget ydelse der kan presses ud af meget små enheder. En sammenligning med eksempelvis J2SE er svær, da der er tale om to forskellige målgrupper.

Kilder

James White & David Hemphill - *Java 2 Micro Edition, Java in small things, Manning 2002*
John Lewis & William Loftus - *Java Software Solutions, Addison-Wesley 2001*
Deitel, Deitel, Nieto & Steinbuhler - *Wireless Internet & Mobile Business, Prentice Hall 2002*
J2ME Data Sheet - http://java.sun.com/j2me/j2me-ds.pdf
CLDC HotSpot Implementation White Paper - http://java.sun.com/products/cldc/wp/CLDC_HI_WhitePaper.pdf
Connected Limited Device Configuration 1.1 - http://jcp.org/aboutJava/communityprocess/final/jsr139/index.html
KVM Data Sheet - http://java.sun.com/products/cldc/ds/
KVM White Paper - http://java.sun.com/products/cldc/wp/KVMwp.pdf
CDC Technical White paper - http://java.sun.com/products/cdc/wp/CDCwp.pdf
CDC Specification 1.0 - http://jcp.org/aboutJava/communityprocess/first/jsr036/index.html
The CVM Virtual Machine - http://java.sun.com/products/cdc/cvm/
Java HotSpot White Paper - http://java.sun.com/products/hotspot/docs/whitepaper/Java_Hotspot_v1.4.1/JHS_141_WP_d2a.pdf
J2EE 1.4 Specifications - http://java.sun.com/j2ee/1.4/download.html#platformspec
Java 2 Platform, Standard Edition Data Sheet - http://java.sun.com/j2se/1.4/datasheet.1_4.html
MicroJava.com FAQ - http://www.microjava.com/developer/faq
Java Virtual Machine Debug Interface Reference - http://java.sun.com/products/jdk/1.2/docs/guide/jvmdi/jvmdi.html
Project Monty White Paper - http://java.sun.com/products/cldc/wp/ProjectMontyWhitePaper.pdf

Kommentar af simonvalter d. 17. Apr 2004 | 1

en artikel der holder hvad den lover. Udemærket beskrevet, jeg ville nu godt have haft noget mere uddybende omkring f.eks hvorfor klasser gennemgår en "verification" før deployment noget man nok ikke får meget ud af medmindre man læser nærmere på emnet.

Kommentar af ioni d. 04. Jul 2004 | 2

God artikel.

Kommentar af conrad d. 13. Apr 2004 | 3

Kommentar af avlund d. 03. Jun 2004 | 4

Udmærket artikel. Godt opbygget og struktureret, relevant læsestof, og i øvrigt rart med de mange kildehenvisninger, så man ved hvor man kan finde mere info.

Kommentar af thundergod d. 17. May 2004 | 5

Kommentar af arne_v d. 12. Apr 2004 | 6

God artikel om et emne som der mangler information om. De fleste ved at man kan køre Java på mobil telefoner og at det har noget med J2ME at gøre, men de færreste har sat sig ind CDLC, KVM etc.. Beskrivelsen af J2SE & J2EE er lidt overfladisk, men det er ikke noget problem, da de er beskrevet mange andre steder.

Kommentar af aniels21 d. 21. Apr 2004 | 7