J2ME
Część I

mgr inż. P.J. Podsiadły

Trochę historii

J2ME zostało po raz pierwszy zaprezentowane w czerwcu 1999, na konferencji JavaOne, jako młodsze rodzeństwo J2SE i J2EE. Jednakże w tym czasie programiści kładli duży nacisk na programowanie rozproszone i byli bardziej zainteresowani tym co oferuje w tym zakresie J2EE.

Sytuacja uległa zmianie w ciągu następnych 2 lat, dlatego że programiści dostrzegli zalety posiadania małych urządzeń mogacych wykonywać programy napisane jezyku Java.

W 2001 na konferencji JavaOne poświęcono J2ME jedną część tej konferencji. Od tamtej pory następuje ciągły wzrost zainteresowania J2ME
.

Czym jest J2ME

J2SE jest wersją języka Java firmy Sun Microsystems przeznaczoną na rynek konsumencki i systemów wbudowanych obejmujących:

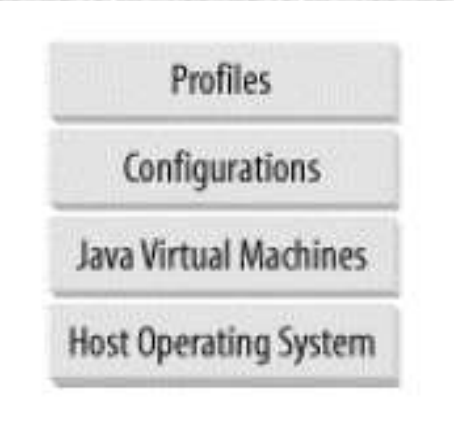
- telefony komórkowe,
- pagery,
- set-top boxy
- PDA
- samochodowe systemy nawigacji.

Dostarcza narzędzi pozwalających bydować aplikacje sieciowe dla urządzeń przenośnych, zapewnia przenośnośc pomiędy różnymi platformami i bezpieczeństwo.

J2ME definiuje następujące składniki

- rodzinę mszyn wirtualnych, do użytku na różnych typach urządzeń (każda z nich posiada inne wymagania sprzętowe),
- grupę bibliotek i API, które można uruchamiać na każdej z maszyn wirtualnych, stanowiących tzw. konfiguracje i profile,
- różnorodne narzędzia programistyczne.

Pierwsze dwa składniki stanowią tzw. Runtime enviroment. Jego struktura przedstawiona jest poniżej.



Konfiguracje

Konfiguracje grupują urządzenia o podobnych zasobach sprzętowych tzn. moc obliczeniowa, ilość pamięci, określa także następujące cechy:
- wspierane cechy języka Java,
- wspierane cechy maszyny wirtualnej,
- wspiercie podstawowych bibliotek i API.

Aktualnie istnieją dwie podstawowe konfiguracje CLDC (Connected Limited Device Configuration) i CDC (Connected Device Configuration).

CDC

Ta konfiguracja jest przeznaczona dla urządzeń o dużej mocy obliczeniowej jak set-top boxy, samochodowe systemy nawigacji, odb. telewizji internetowej). Zawiera „pełnowymiarową” maszynę wirtualną podobną do tej w J2SE, główne różnice dotyczą ilości dostępnej pamięci i możliwości wyświetlania obrazu.

Podstawowe wymagania:
- urządzenie wyposażone w 32 bitowy procesor,
- 2MB całkowitej pamięci dostępnej dla Javy (tzn. RAM + flash, lub ROM),
- wymaga maszyny wirtualnej zgodnej z Java 2 „Blue Book”
- urządzenie posiada dostęp do sieci, często bezprzewodowy, sporadyczny (nie ciągły, przerywany), z ograniczonym pasmem (9600 bps lub mniej),
- może posiadać interface użytkownika lecz nie jest on obowiązkowy

CLDC

Ta konfiguracja przeznaczona jest dla małych urządzeń przenośnych jak telefony komórkowe, PDA, pagery. Jest ona również bardziej rozpowszechniona, choćby ze względu na stale rosnącą liczbe dostępnych urządzeń wspierających technologię Java

Podstawowe wymagania:
- 16 lub 32 bitowy procesor pracujący z częstotliwością co najmniej 25 MHz
- pamięć od 160 do 512kB dostępnych dla platformy Java (RAM +flash, lub ROM),
- urządzenie może mieć ograniczone zasilanie, zwykle bateryjne,
- dostęp do sieci, często bezprzewodowy, sporadyczny z ograniczonym pasmem (jak w CDC).
- może posiadać interface użytkownika z pewnym stopniem wyrafinowania, jednak nie jest on obowiązkowy.

Mszyny wirtualne

Jak wspomniano wcześniej każda z konfiguracji wymaga innej maszyny wirtualnej ze wzgledu na różne zestawy wspieranych funkcji.

Aktualnie używa się 2 maszyn wirtualnych KVM (Kilo Virtual Machine) dla CLDC i CVM dla CDC.

Uwaga!! Początkowo CVM było skrótem od Compact Virtual Machine, jednak z powodu możliwości pomyłek „Compact” z K w KVM obecnie „C” nie ma żadnego znaczenia.

KVM

Jest kompletne środowisko uruchomieniowe (runtime enviroment) dla małych urządzeń. Zgodne z definicją zawarta w Java Virtual Machine Specification + pewne modyfikacje pozwalające na prawidłową pracę na małych urządzeniach (z ograniczonymi zasobami i pamięcią rzędu kilkuset kB).

CVM

Jest przeznaczona dla większych urządzeń, należących do CDC, wspiera wszystkie funkcje maszyny wirtualnej Java 2 Ver. 1.3 w zakresie bezpieczeństwa, słabych referencji, JNI, RMI (Remote Method Invocation).

Referencyja implementacja firmy Sun Microsystems jest dostępna dla systemów Linux i VxWorks

Profile

Profile są zbiorami funkcji API, rezydującymi na konfiguracjach, które zapewniają programom dostęp do własności specyficznych dla danej grupy urządzeń.

Aktualnie dostępne są następujące profile:

- MIDP – do użytku z CLDC, zapewnia zbiór API do współpracy z urządzeniami przenośnymi jak tel. komórkowe i pagery. Zawiera zestaw klas służących do budowania interfejsu użutkownika, połączeń sieciowych (networking) i zapisu danych*. Zawiera również środowisko uruchomieniowe pozwalające na załadowanie nowych programów do urządzenia użytkownika końcowego.

Aplikacje korzystające z profilu MIDP często nazywane sa MIDletami.

* zapis danych (presistent storage) polega na zapisie informacji w rekordach w sposób podobny do bazy danych, głównie dlatego, że małe urządzenia przenośnie nie implementują systemu plików

- PDA – przeznaczony dla CLDC zawiera API do budowania UI (oparte na AWT) i do zapisywania danych.

- Foundation – przeznaczony dla CDC, nie zawiera żadnych API do budowy UI, stanowi raczej podstawę dla profili Personal i RMI.

- Personal – jest rozszerzeniem profilu Foundation o GUI pozwalające na uruchamianie Java Web Applets. Jest kompatybilny z Personal Java 1.1 i 1.2,

- RMI – jest przeznaczony dla CDC i rozszerza profil Foundation o RMI, jest kompatybilny z J2SE RMI API 1.2 lub wyższym

Wymagane oprogramowanie

Do pisania aplikacji w J2ME potrzebne są:

- J2SDK
- J2ME Wireless Toolkit
- prosty edytor tekstu (np. Notatnik) lub środowisko zintegorwane (Forte For Java, SunOne Studio, Netbeans, Eclipse),
- przeglądarka internetowa do czytania dokumentacji.

CLDC w szczegółach

CLDC zapewnia:
- pewien podzbiór cech języka Java i funkcjonalności maszyny wirtualnej
- podzbiór podstawowych bibliotek (java.lang i java.util)
- podstawowe we/wy (java.io)
- podstawowe funkcje sieciowe (javax.microedition.io)
- bezpieczeństwo.

Nie zapewnia:
- obsługi cyklu życia aplikacji,
- obsługi UI,
- obsługi zdarzeń,
- interakcji użytkownik – aplikacja

Te funkcje (właściwości) są zapewniane przez profil MIDP.

Różnice w VM:

- brak arytmetyki zmiennoprzecinkowej – nie istnieją typy danych float i double, głównie ze względu na ograniczenia sprzętu na którym CLDC działa,
- brak finalizacji – nie można wykonać Object.finalize()
- ograniczona obsługa błędów – w CLDC istnieją tylko trzy klasy błędów:
    - java.lang.Error,
    - java.lang.OutOfMemoryError,
    - java.lang.VirtualMachineError
- brak Java Native Interface (JNI) – ze względów bezpieczeństwa i zasobożerności,
- brak możliwości definiowania class-loaderów przez użytkownika
- brak grup wątków – ale w zamian można użyc kolekcji obiektów do przechowywania kilku wątków,

Bezpieczeństwo

Model bezpieczeństwa w CLDC jest podzielony na dwa obszary:

- bezpieczeństwo na poziomie maszyny wirtualnej:

  aplikacja wykonywana przez KVM nie może być w stanie uszkodzić urządzenia na którym działa. Jest to zapewnione prze weryfikator klas, który sprawdza czy bytecode nie zawiera odwołań do niewłaściwych adresów w pamięci, zapewnia również to, że klasa załadowana do pamięci nie może się wykonać w sposób niezgodny ze specyfikacja maszyny wirtualnej.

- bezpieczeństwo na poziomie aplikacji:

  VM wsperająca CLDC zapewnia prosty model bezpieczeństwa polegający na tym, że aplikacja działa w zamkniętym środowisku i może odwoływać sie tylko do klas wspieranych przez dane urządzenie

Klasy dziedziczone z J2SE

Package	Classes
java.lang	Boolean, Byte, Character, Class, Integer, Long, Math, Object, Runnable, Runtime, Short, String, StringBuffer, System, Thread, Throwable
java.io	ByteArrayInputStream, ByteArrayOutputStream, DataInput, DataOutput, DataInputStream, DataOutputStream, InputStream, OutputStream, InputStreamReader, OutputStreamWriter, PrintStream, Reader, Writer
java.util	Calendar, Date, Enumeration, Hashtable, Random, Stack, TimeZone, Vector

Inherited exception and error classes

Package	Class
java.lang	ArithmeticException, ArrayIndexOutOfBoundsException, ArrayStoreException, ClassCastException, ClassNotFoundException, Error, Exception, IllegalAccessException, IllegalArgumentException, IllegalMonitorStateException, IllegalThreadStateException, IndexOutOfBoundsException, InstantiationException, InterruptedException, OutOfMemoryException, NegativeArraySizeException, NumberFormatException, NullPointerException, RuntimeException, SecurityException, StringIndexOutOfBoundsException, VirtualMachineError
java.io	EOFException, IOException, InterruptedIOException, UnsupportedEncodingException, UTFDataFormatException
java.util	EmptyStackException, NoSuchElementException

Metody usunięte z java.lang.String

public void valueOf(float f)
public void valueOf(double d)
public int compareToIgnoreCase(String str)
public boolean equalsIgnoreCase(String anotherStr)
public static copyValueOf(char[] data)
public static String copyValueOf(char[] data, int offset, int count)
public String intern()
public int lastIndexOf(String str)
public int lastIndexOf(String str, int fromIndex)
public boolean regionMatches(int toffset, String other,
int ooffset, int len)
public String toLowerCase(java.util.Locale locale)
public String toUpperCase(java.util.Locale locale)

Metody usunięte z java.lang.StringBuffer

public StringBuffer append(float f)
public StringBuffer append(double d)
public StringBuffer insert(int offset, float f)
public StringBuffer insert(int offset, double d)
public StringBuffer insert(int index, char[] str, int offset, int len)
public StringBuffer replace(int start, int end, String str)
public String substring(int start)
public String substring(int start, int end)

java.lang.Runtime

```
public void exit(int status);
public native long freeMemory();
public native void gc();
public static Runtime getRuntime();
public native long totalMemory();
```

java.lang.System

```
public static final PrintStream err;
public static final PrintStream out;

public static native void arraycopy(Object src,
int src_position, Object dst, int dst_position,
int length);

public static native long currentTimeMillis();
public static void exit(int status);
public static void gc();
public static String
getProperty(String key);

public static native int identityHashCode
       (Object x);
```

java.lang.Math

```
public static int abs(int a);
public static long abs(long a);
public static int max(int a, int b);
public static long max(long a, long b);
public static int min(int a, int b);
public static long min(long a, long b);
```