Introduction J2ME

Magnus Bladh
Streetmedia 7 AB

Welcome!

About Streetmedia7:

- StreetMedia 7 is a company based in Uppsala.
- Produces games and applications for mobile marketing.
- Offices in Uppsala and Toronto.

About me:

- Studied Computer science and mathematics at Uppsala university. Has previously worked for Vattenfall and RSA Security.
- Contact: magnus.bladh@streetmedia7.com
Why J2ME?

- Mobile phones are limited
  - Processor (~)
  - Memory (Jar max 125 kB, heap 500 kB)
  - Screen size (128x128 pixels)
  - Keys (1 -9 + joystick + left softkey + right softkey)
  - Multimedia

What is J2ME

- J2ME Core concepts
  - Configurations. (CLDC, CDC)
  - Profiles (MIDP1.0, MIDP 2.0)
  - Additional Packages (Bluetooth API, Location API...)
- Virtual machines
  - KVM
  - CVM
A configuration is a complete Java runtime environment, consisting of three things:

- A Java virtual machine (VM)
- Native code to interface to the underlying system.
- A set of core Java runtime classes.

J2ME defines two configurations:

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

CLDC

- CLDC is for very constrained (limited) devices:
  - devices with small amounts of memory and/or slow processors.
  - Basics from the java.lang, java.io and java.util packages, with a few additional classes from the new javax.microedition.io package.
- CLDC 1.0 & 1.1
- JVM fits in 128k
  - No floating point types
  - No object finalization
CDC

- Connected Device Configuration (CDC)
  - CDC includes a full Java VM and a much larger set of core classes, so it requires more memory than the CLDC and a faster processor.
  - CDC is a superset of CLDC

Profiles

- A profile adds domain-specific classes to a configuration to fill in missing functionality and to support specific uses of a device.
- CLDC based profiles:
  - Mobile Information Device Profile (MIDP)
  - Personal Digital Assistant Profile (PDAP)
- CDC based profiles:
  - Foundation Profile (FP)
  - Personal Basis Profile (PBP)
Devices that have the following minimal characteristics:

- Enough memory to run MIDP applications
- A bit addressable display at least 96 pixels wide by 56 pixels high, either monochrome or color.
- A keypad, keyboard, or touch screen.
- Two-way wireless networking capability.
- Comes in two versions MIDP1.0 and MIDP2.0

The MIDP adds APIs in a number of areas to the very basic APIs defined by the CLDC. The new features include:

- Support for application lifecycle management similar to the way applets are defined in Java 2 Standard Edition.
- Persistent storage of data.
- HTTP-based network connectivity based on the CLDC’s Generic Connection Framework.
- Simple user interface support, with enough flexibility to build games or business applications.
Optional packages

- An optional package is a set of APIs in support of additional, common behaviors that don’t really belong in one specific configuration or profile.
- Example of Optional packages:
  - Bluetooth API
  - Location API
  - Multimedia API
  - ...

...and what is a MIDlet?

- A MIDP application is referred to as a MIDlet.
  - MIDP does not support the running of traditional applications that use a static main method as their entry point.
  - Its entry point is a class that extends the `javax.microedition.midlet.MIDlet` class.
  - MIDP defines an application lifecycle model similar to the applet model.
...and what is a MIDlet suite?

- One or more MIDlets are packaged together into what is referred to as a MIDlet suite.
- A MIDlet suite is basically a standard JAR (Java archive) file and a separate file called an application descriptor (JAD).
- All the user-defined classes required by the suite's MIDlets must be in the JAR file, along with any other resources that the MIDlets require.
- The JAR file must also include a manifest with a number of MIDP-specific entries that describe the MIDlets in the suite.

Developing MIDlets

- The reality of MIDP programming today is that the applications you can write are constrained in many ways.
  - Memory is a particularly scarce resource.
  - Limited screen size.
  - Limited processor.
  - There are bugs in the implementations of J2ME on some phones.
  - Operators as the possibility to brand the phones.
Porting applications

- Porting and testing applications is very time consuming since there are differences between different phone models such as:
  - Screen size
  - Memory
  - Processor
  - Bugs
  - Operator problems
  - Specification divergence

Tools

- To solve the problems with restricted devices and porting you'll need some tools.
  - Computer with Windows or Linux.
  - JDK (Java development kit)
  - WTK (wireless toolkit)
  - IDE (Eclipse, Jbuilder, ...)
  - A Java phone
- JDK, WTK and IDE can be downloaded from suns website or www.eclipse.org.
More tools

- There are some additional tools that might be useful to have:
- ANT (To create a build system)
- Antenna (ANT tasks to modify JAD files create package and obfuscate)
- J2ME Polish. A very useful tool to preprocess code. Simplifies porting a lot!!!
- Subversion or CVS. Version control system.
- Bluetooth.

...and more tools

- Emulators. Very useful when testing MIDlets.
- Remember: They are not reliable.
- Emulators
  - Nokia Carbide.
  - Siemens mobility toolkit 3.0.
  - Samsung JSDK.
  - SonyEricsson ME SDK for CLDC.
  - Motorola Launchpad.
Tips and tricks!

- Always think about the phones restrictions!!!
- Do as much as possible on the serverside.
- Build small applications
  - Use obfuscators
  - Minimize graphics, sounds...
- Always think about the phones restrictions!!!
- Help your GC, set used objects to null
  - Only create objects when you need them
  - Release resources as quickly as possible

More tricks

- Always think about performance
  - Optimize at the right place, use the profiler
  - Use local variables.
  - Control your loops
  - Be careful with String concatenation. Take a look at StringBuffer.
- Remember: DON'T trust the emulator.
- Remember: DON't trust the phone.
Useful sites

- www.j2meforums.com/forum // Good forum