J2ME Tutorial

send any comments / suggestions to **michael@uberthings.com**

Introduction

This tutorial assumes that you have some familiarity with general programming concepts and the Java language.

What is J2ME?

J2ME stands for Java 2, Micro Edition. It is a stripped-down version of Java targeted at devices which have limited processing power and storage capabilities and intermittent or fairly low-bandwidth network connections. These include mobile phones, pagers,wireless devices and set-top boxes among others.

A Sample Wireless Stack would consist of:

- Profiles
- Configurations
- Java Virtual Machines
- Host Operating System

What is a J2ME Configuration?

A configuration defines the minimum Java technology that an application developer can expect on a broad range of implementing devices.

J2ME Connected, Limited Device Configuration (CLDC)

- specifies the Java environment for mobile phone, pager and wireless devices
- CLDC devices are usually wireless
- 160 - 512k of memory available for Java
- typically has limited power or battery operated
- network connectivity, often wireless, intermittent, low-bandwidth (9600bps or less)

J2ME Connected Device Configuration (CDC)

- describes the Java environment for digital television set-top boxes, high end wireless devices and automotive telematics systems.
- device is powered by a 32-bit processor
- 2MB or more of total memory available for Java
- network connectivity, often wireless, intermittent, low-bandwidth (9600bps or less)

These two configurations differ only in their respective memory and display capabilities.

What is a J2ME Profile?

A specification layer above the configuration which describes the Java configuration for a specific vertical market or device type.

J2ME Profiles

J2ME Mobile Information Device Profile (MIDP)

- this is the application environment for wireless devices based on the CLDC
- contains classes for user interface, storage and networking

J2ME Foundation Profile, Personal Basis, Personal and RMI profiles

- these are profiles for devices based on the CDC, which are not addressed in this tutorial

Virtual Machines

The CLDC and the CDC each require their own virtual machine because of their different memory and display capabilities. The CLDC virtual machine is far smaller than that required by the CDC and supports less features. The virtual machine for the CLDC is called the Kilo Virtual Machine (KVM) and the virtual machine for the CDC is called the CVM.

[back to top]

Tools

PC | **MacOS X** | **Linux**

First make sure that you have the Java 2 SDK, Standard Edition (J2SE SDK), version 1.4.2 (or later). This is essential for development. If you haven't installed it, download it and install it from here **http://java.sun.com/j2se/downloads/**.
You absolutely MUST have the J2SE SDK installed before you install the Java Wireless Toolkit as you will be needing the tools it contains (such as javac) to compile and run your MIDlets.

Then download the J2ME Wireless Toolkit (WTK) which is available free from Sun here - **http://java.sun.com/products/j2mewtoolkit/**. I'm going to assume that you'll be installing this in the *C:\j2mewtk* directory, if you use another directory, just modify the paths accordingly.

Paths

Java needs to know where all your files are, so we need to add the location of the Java binaries to the system path.

Windows 95/98

Go to Start->Run. Type in *command*. Then type

```
SET PATH=%PATH%;C:\j2mewtk\bin
```

You should also edit your C:\autoexec.bat file to include this line, so you don't have to enter it every single time you restart your computer. After you've done this, you should be able to run the tools included in the Java Wireless Toolkit from any directory on your system.

Windows 2000/XP

- Go to Control Panel -> System.
- Click on the Advanced Tab
- Click on the Environment Variables button
- Double-click the PATH variable in the System variables box
- At the end of the *Variable value* field, add the path to your J2ME WTK installation - for me this is

something like *C:\j2mewtk*
- If you had to install the J2SE SDK too, it's a good idea to add the path for that - for me this is *C:\j2sdk1.4.2_03* and *C:\j2sdk1.4.2_03\bin*. **Here's what my screen looked like.**

A good way to test if this worked is to type the preverify command without any arguments in the command line. You should see something like this on your screen.

```
C:\> preverify
Usage: PREVERIFY.EXE [options] classnames|dirnames ...

where options include:
-classpath
Directories in which to look for classes
-d Directory in which output is written
@ Read command line arguments from a text file.
```

[back to top]

Application Development

MIDlets vs Applets

MIDlets are applets for mobile phones. Just like applets, they run in a protected sandbox - the KVM - but unlike applets, they are extremely limited. MIDP 1.0 is currently found on most Java-capable phones and is fairly restrictive. As an example - the KVM doesn't allow you to process floating point numbers yet and MIDlets written for MIDP 1.0 can't access anything outside of the sandbox without proprietary APIs from phone manufacturers. So, put your dreams of developing the ultimate MIDlet with hooks into every part of your phone OS on the backburner for a while. If you want to find out exactly how limited MIDP 1.0 is, you should probably **read the spec here.** Once you've done that you might want to check out **MIDP 2.0 and see what Sun has fixed** with that spec. For the time being we're going to write our first MIDlet - a full-featured "Hello MIDlet" application.

Simple HelloMIDlet

We're going to use a program called Ktoolbar from the JAVA WTK which we installed earlier.

- Go to Start->Programs->J2ME Wireless Toolkit 2.1->KToolbar.
- Click on the *New Project* button and name your project *HelloProject* and your MIDlet *HelloMidlet*. **You should see something like this.**
- Once you press Create Project, KToolbar will create a bunch of directories for your project in the apps subdirectory. We're going to ignore most of them for the moment and focus on a few important ones

```
C:\j2mewtk\apps\HelloProject - the main directory for your project
C:\j2mewtk\apps\HelloProject\bin - where Ktoolbar stores .jar, .jar and manifest.mf
files
C:\j2mewtk\apps\HelloProject\classes - where the class files are stored
C:\j2mewtk\apps\HelloProject\src - where the source .java files are stored
```

- Now, fire up your favourite text editor - I like **Textpad**- and type in the following code

```
/* Hello Midlet - your first program*/
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class HelloMidlet
extends MIDlet
implements CommandListener {
private Form mMainForm;

public HelloMidlet() {
mMainForm = new Form("HelloMidlet");
```

```
mMainForm.append(new StringItem(null, "Hello, MIDP! \n\nYou and me - we're gonna make
sweet MIDlets together! "));
mMainForm.addCommand(new Command("Exit", Command.EXIT, 0));
mMainForm.setCommandListener(this);
}

public void startApp() {
Display.getDisplay(this).setCurrent(mMainForm);
}

public void pauseApp() {}

public void destroyApp(boolean unconditional) {}

public void commandAction(Command c, Displayable s) {
notifyDestroyed();
}
}
```

- Save this file as HelloMidlet.java in the *C:\j2mewtk\apps\HelloProject\src*
- Go back to KToolBar - then click on Build and then Run - you should see **something like this.**
- Click on the Launch softkey in the emulator to get your MIDlet to say hello. You've just written your first MIDlet!

[back to top]

Provisioning

Okay, now how do I get my code onto my phone?

Once you've created your lovely little MIDlet and ensured that everything worked smoothly in the emulator, the next step is to get it running on an actual device. *Provisioning* is the name given to the process of deploying your application in such a way that it is easily downloaded and installed on the device.

1. Over The Air (OTA) Provisioning

OTA provisioning allows users to download your application wirelessly using the WAP browsers built into their phones. To begin, we need to take a look at the Java Application Descriptor (JAD) file that is created when you package a MIDlet using the J2ME Wireless Toolkit. The JAD file stores information about your application and lets you modify various parameters of the MIDlet suite such as where the icon resource can be found, which MIDlets are included and where you can download the full version of the application. To edit a JAD file using the Wireless Toolkit, open your project, then click on *Settings*. This will open up a new window with a number of tabs - API Selection, Required, Optional, User Defined, MIDlets, Push Registry and Permissions.

API Selection
This is where you choose which version of MIDP your application will use and which optional packages (JSRs) are included. The default is set to JTWI (Java Technology for the Wireless Industry) which allows you to use MIDP 2.0 as well as MMAPI and other exciting things. If you're having any problems with your application on your device try changing this to MIDP 1.0.
Required
This tab includes various options which are essential for packaging a MIDlet suite. The *MIDlet-Jar-URL* attribute is where we will define the location of the packaged JAR file to be downloaded to the device.
Optional
This tab includes optional parameters for your MIDlet - such as the path to the icon for the entire suite, a description and a *MIDlet-Info-URL* parameter.

User Defined

This tab includes user defined variables that your MIDlet can use - such as a common URL that you don't want to hard wire into the source code.

MIDlets

This tab manages all the settings for the MIDlets within your suite. At the very least you need to have one file here. This is also where you set the path to the MIDlet's icon resource.

Push Registry

This lets you configure the Push Registry which allows your MIDlet to listen and act on information received from a remote source. MIDP 2.0 Only.

Permissions

Under MIDP 1.0, applications could only access libraries packaged inside the suite - this was called the *sandbox model*. MIDP 2.0 introduces the concept of *trusted applications* which allow access beyond the sandbox. This section allows you to specify which APIs are accessible.

For our purposes - the most important property is the *MIDlet-Jar-URL* within the Required tab. Here are the steps you need to take:

Create a folder on your web server

Hopefully you have an account with a web provider - login to that account and create a directory for your MIDlets to live and be served from. I created the directory ***http://uberthings.com/mobile/midlets***. Once you've got that, you need to make a few changes to allow your server (assumed to be Apache) to serve JAD and JAR files correctly. Go to the root of your account and edit or create your *.htaccess* file. Add these lines:

```
AddType text/vnd.sun.j2me.app-descriptor jad
AddType application/java-archive jar
```

Save this file. If you're not using Apache, ensure that your MIME types include the above two settings.

Specify the *MIDlet-Jar-URL*

Click on Settings then go to the Required Tab. In the *MIDlet-Jar-URL* field, fill in the absolute URL of your JAR file. This will normally be something like *http://mydomain/mydir/HelloProject.jar*. For my server, this was *http://www.uberthings.com/mobile/midlets/HelloProject.jar*.

Package your MIDlet

Click on Project->Package->Create Package. This will create a .jar and a .jad file in your applications bin folder. For my application - this was *c:\j2mewtk\apps\HelloProject\bin\HelloProject.jar* and *c:\j2mewtk\apps\HelloProject\bin\HelloProject.jad*.

Upload the packaged MIDlet suite

Upload the JAR and JAD files that the packaging operation created to the folder you created earlier.

Test with your device

Open the WAP browser on your phone and point it to the URL of the JAD file. Using my example, this would be *http://uberthings.com/mobile/midlets/HelloProject.jad*. Your device should then prompt you to download and install the MIDlet. Carry it around and show it off to all your friends!

2. Cable / Bluetooth

If you've got a Bluetooth adaptor or a USB cable which connects directly to your phone, you can use

this to quickly test your packaged midlet.

Windows XP/2000: Browse to the bin folder of your project, right click on the .jar file and select Send To->Bluetooth->YOURDEVICE.
MacOS X: Click on the Bluetooth icon in the menu bar, choose Send File. Send it to your device. This should send a message to your phone which will install the MIDlet once opened. This should work on most Nokia Series 60 phones (3650, 6600, N-Gage etc).

[back to top]

Resources

- **Sun J2ME Portal** - They started all this Java stuff, they might just have some useful stuff here
- **Bill Day's J2ME Archive** - Bill Day, a technology evangelist for Sun has a great collection of resources about Java development tools
- **Ben Hui's** site has the best Bluetooth tutorials and an extensive links section
- **Eclipse** - is my favourite Java IDE. It's open source, cross-platform and totally free.
- **Eclipse Plugins** - a great resource for getting all sorts of things to work with Eclipse.

[back to top]
