Guide to getting started in J2ME for the Motorola A780 phone

This guide will take you through setting up a build environment for J2ME in Windows and in writing a few sample applications for the A780 phone. There are some notes at the end if you are on another platform, but we strongly recommend using a Windows environment so that you can make full use of the simulator and debug output.

1. **Setting up Eclipse**

   We recommend using Eclipse for your Java editing. If you're not familiar with Eclipse, you can get more information here: http://www.eclipse.org/

   The latest version of Eclipse can be downloaded here:
   http://www.eclipse.org/downloads/download.php?file=/eclipse/downloads/drops/R-3.1.1-200509290840/eclipse-SDK-3.1.1-win32.zip

2. **Setting up the Motorola J2ME SDK**

   Next, you'll need to set up the Motorola SDK. You will need to create an account on the Motocoder website:
   https://www.motocoder.com/motorola/template.jsp;jsessionid=1efdf:43cd0cee:990f451891647ff?filename=center_registration.jsp

   Then download the latest SDK from:
   http://www.motocoder.com/motorola/download.jsp?FILENAME=downloads/files/SDK542.zip

   Install this SDK to your windows system in the default directory (makes later steps much easier!)

3. **Setting up EclipseME J2ME Plugin**

   Next, you need to set up the J2ME plugin that will tie the Motorola SDK into the Eclipse environment.

   Install EclipseME with these instructions:
   http://eclipseme.org/docs/installEclipseME.html

   Configure EclipseME with these instructions:
   http://eclipseme.org/docs/configuring.html

4. Creating a simple J2ME application

Here's a basic runthrough on how to create a simple J2ME application, basically a J2ME hello world.

1. Create a new J2ME project: http://eclipseme.org/docs/createProject.html
2. Create a new Midlet in the project named "HelloWorld" A midlet is a J2ME application and represents the main UI class for a project. http://eclipseme.org/docs/createMidlet.html
3. You can use the following source for the midlet: Helloworld.java
4. You can debug and install using the installation instructions below.

5. Debugging J2ME applications in the simulator

This section will cover debugging and the use of the M3 simulator in conjunction with Eclipse. It is recommended that you try to run applications in the emulator before trying to run them on the phone.

1. Choose Run... from the Run menu of ecipse.
2. Under Configurations, double click on the header Wireless Tookit Emulator to create a new configuration. The project name should be auto-populated with your current project. Under "Executable: Midlet" enter the name of your Midlet.
3. Under the emulation, platform definition tab, choose "Motorola SDK for J2ME MOJM3"
4. Click on Run to start the Midlet in the emulator.

To debug instead of run, complete the same setup under "debug..."

6. Installing the application on the phone

There are two files needed to install your J2ME application on the phone. One is the jar file which contains all of your compiled java code. The other is the jad file while contains metadata about your application and permissions information. Both of these are generated by Eclipse. To make sure they have been updated properly before loading them onto the phone, right click on your project name in the Package Explorer and select "J2ME" "Create Package" from the context menu. This will force a regeneration of the jar and jad file in the deployed folder for your project.

Make sure you phone is in USB Mass Storage mode by going to the main menu, set up, usb mode and choosing "mass storage." Select "save" to exit out of the settings. Your phone will now mount as a USB hard drive when plugged into your computer.

Plug in your phone and open up a file explorer window with the newly mounted drive. Copy the jad and jar files from the deployed folder of your project to the drive. Wait a few seconds and then unplug your phone. The files are now in the flash file system of the phone.

Navigate to the folder that you saved the files to on your phone by going to the main menu and opening the "files" application. Click once on the jad file (NOT the jar file) to install the application. Say ok to any dialogs that may appear and choose the main menu group that you want the shortcut to your new application to appear in.

You may now launch your application from the main menu by clicking on its icon.

Note: Since you don't have a debugging console when you are running applications on the phone, it is recommended that you catch all exceptions and call the following method with the toString() of your exception:

```
public void displayException(String e) {
    Alert a = new Alert("error:",e,null,AlertType.ERROR);
    a.setTimeout(Alert.FOREVER);
    myDisplay.setCurrent(a,titleScreen);
}
```

7. A word about permissions

When you use various classes in J2ME, you need to make sure you set the permissions in your JAD file appropriately. You can do this by clicking on the JAD file in your main project (not the one in "deployed") and going to the "optional" tab. Under "Midlet Permissions:" add the appropriate permissions comma separated.

The following are common permissions:

- javax.microedition.location.Location
- com.motorola.file.readaccess
- com.motorola.file.writeaccess
- javax.microedition.io.Connector.http
- com.motorola.phone
- javax.wireless.messaging.sms.send
- javax.wireless.messaging.sms.receive

8. Example code:

0. HTTP Download:

```
StringBuffer s = new StringBuffer();
HttpConnection c =
(HttpConnection)Connector.open("http://web.mit.edu/index.ht
ml");
InputStream is = c.openInputStream();
byte b;
while ((b = (byte)is.read()) != -1) {
    s.append((char)b);
}
```

```
is.close();
c.close();
```

Permissions needed: `javax.microedition.io.Connector.http`

1. Writing to a file:

```
FileConnection sc =
(FileConnection)Connector.open("file:///phone/tmp.txt");
OutputStream os = sc.openOutputStream();
os.write(("text to go into the file").getBytes());
os.flush();
os.close();
```

Permissions needed: `com.motorola.file.writeaccess`

2. Reading a file:

```
FileConnection sc =
(FileConnection)Connector.open("file:///phone/tmp.txt");
InputStream is = sc.openInputStream();
StringBuffer sofar = new StringBuffer();
byte c;
while ((c = (byte)is.read()) != -1){
    sofar.append((char)c);
}
is.close();
```

Permissions needed: `com.motorola.file.readaccess`

3. Sending an SMS:

```
sender = (MessageConnection)Connector.open("sms://");
TextMessage t =
(TextMessage)sender.newMessage(MessageConnection.TEXT_MESSA
GE);
t.setPayloadText(message);
t.setAddress("sms://" + contactNumber);
sender.send(t);
```

Permissions needed: `javax.wireless.messaging.sms.send`

4. Listening for an SMS: blah

Permissions needed: `javax.wireless.messaging.sms.receive`

5. Getting GPS location:

```
LocationProvider loc = LocationProvider.getInstance(null);
loc.setLocationListener(ll,0,-1,-1);
Location location = loc.getLocation(60*3);
```

```
String lat =
location.getQualifiedCoordinates().getLatitude();
String lon =
location.getQualifiedCoordinates().getLongitude();
```

Permissions needed: `javax.microedition.location.Location`

6. Getting Cell ID:

```
String cellID = System.getProperty("phone.cid");
```

Permissions needed: `none`

7. Placing a phone call:

```
Dialer dialer = Dialer.getDefaultDialer();
dialer.startCall("6172531000");
```

Permissions needed: `com.motorola.phone`

9. Helper classes

The following helper classes implement common features of standard desktop java that you might find useful to use in J2ME:

- PropertiesFile.java
- StringTokenizer.java
- BufferedInputReader.java
- Timer.java / TimerListener.java

10. J2ME developing on Linux/Mac

While the Motorola emulator only runs on Windows, you should be able to code J2ME apps on other platforms. Soon, we will post the jar file for the a780 sdk that you can link into your project in eclipse on any platform.

11. More complete references

- Motorola a780 developer guide
- J2ME In a Nutshell