B

Motorola J2ME SDK

- 安裝 Motorola J2ME SDK
- Motorola J2ME SDK 目錄結構
- Motorola J2ME SDK 內含的輔助開發工具
- 撰寫並編譯 MIDlet
- 執行 MIDlet
- 對 MIDlet 除錯
- Motorola J2ME SDK 對中文的支援

本附錄將為大家介紹如何使用使用命令列模式與批次檔，加上 Motorola J2ME SDK 來開發 Motorola iDEN、i85s 手機程式。由於台灣在資訊通訊基礎建設方面無法滿足這兩款手機的需求，因此這兩款手機並沒有在台灣發售的打算。但是有鑒於可能會有開發這兩款手機的需求，因此特別將本篇納入，僅供需要的人參考。

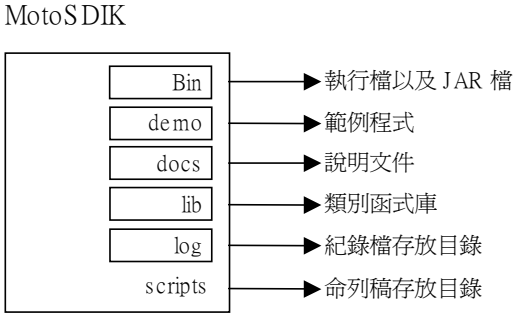
安裝 Motorola J2ME SDK

Motorola J2ME SDK 無法單獨取得，而是附在 CodeWarrior for Java 之中。只要您安裝了 CodeWarrior for Java，就很自然地也安裝了 Motorola J2ME SDK。

Motorola J2ME SDK 位於【CodeWarrior for Java 安裝目錄】\Java_Support\MotoSDK\之下。

Motorola J2ME SDK 目錄結構

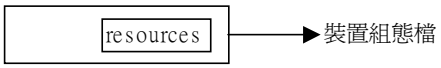
Motorola J2ME SDK 之後，其目錄結構如下圖：



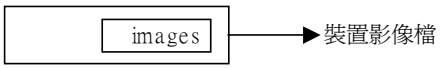
Bin 目錄之下含有使用 Java 所撰寫的組態工具、除錯代理人，以及模擬器。也含有預先審核器及模擬器啟動器。

而 Bin\resources 之中放的都是目的平台的控制檔與影像檔。

Bin

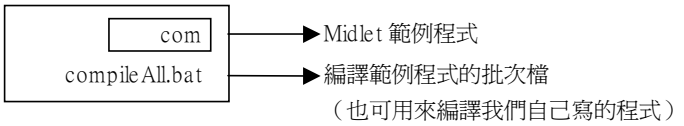


Bin\resources



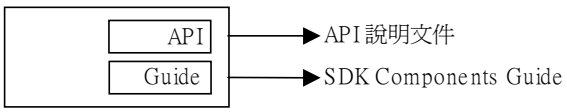
demo 目錄下放置了範例程式的原始碼與其二進位檔。

Demo\midlets

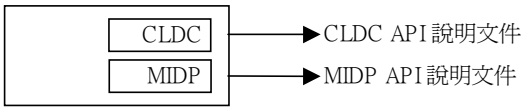


docs 目錄下放置了使用說明書以及類別函式庫參考文件。

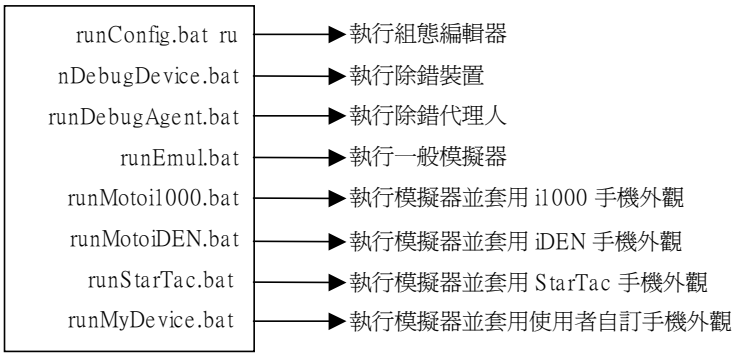
docs



Docs\API



scripts



Motorola J2ME SDK 之中內附許多有趣的範例，它們位於 demo/midlets/目錄底下。有興趣繼續精進的讀者可以藉由這些範例學到更多有關 MIDlet 的深入技巧，由於這些範例的執行畫面過大，所以在本文之中就將他們省略，請讀者們自行參考說明文件。

Motorola J2ME SDK 內含的輔助開發工具

在 Motorola J2ME SDK 之中內含三項輔助開發工具，可以便利我們的程式開發工作，它們分別是:

J2ME 模擬器（J2ME Emulator）

讓您可以在您的 PC 上模擬 Motorola 將來會支援 J2ME 的手機裝置。如此一來就可以在 PC 上直接測試寫好的程式。

Bytecode 驗證器（Bytecode Verifier）

此驗證器用來驗證類別檔（classfile）之中的 bytecode 不會對記憶體做非法的存取。並確認載入虛擬機器的類別檔所做的所有動作皆符合 Java 虛擬機器規格（Java Virtual Machine Specification)。

組態編輯器（Configuration Editor）

讓您能夠建立或修改 device profile。

在 Motorola J2ME SDK 內附的文件之中對這三個工具有詳細的說明，在此就不再贅述。

撰寫並編譯 MIDlet ▼

請先到您的 Motorola J2ME SDK 安裝目錄下的 demo\midlets 目錄底下新增一個名為 HelloMIDlet.java 的檔案，其內容為：

HelloMIDlet.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class HelloMIDlet extends MIDlet
{
   HelloMIDlet()
   {
   }
   protected void startApp() throws MIDletStateChangeException
   {
   }
   protected void pauseApp()
```

```
    {
    }
    protected void destroyApp(boolean unconditional)
            throws MIDletStateChangeException
    {
    }
}
```

在 demo\midlets 目錄之中您會看到一個名為 compileAll.bat 的批次檔，情面有提到，這個批次檔除了可以幫助您編譯所有內附範例程式之外，還可以簡化您自己所撰寫的 MIDlet 之編譯流程。

請在命令列視窗輸入：

```
compileAll HelloMIDlet.java
```

如果成功編譯，螢幕上輸出的結果如下圖所示：

從螢幕上的輸出，您可以發現，要讓 MIDlet 可以在手機上執行，大體上還是要經過兩個動作（與 Spotlet 相似），分別是編譯（compiling）以及預先審核（Preverifying）。

從這個簡短的 MIDlet 製作流程裡頭，相信大家可以發現，比起純粹用 Sun 釋出的 CLDC 撰寫能在 KVM 上執行的 Spotlet 要簡單上許多。原因當於是因為 compileAll.bat 這個批次檔幫我們做掉大部分 dirty work 的緣故。

那麼，如果您寫好的程式並非放在 demo\midlets 目錄之中，是否就無法編譯了？從 compileAll.bat 裡頭，我們可以發現它幫我們完成上述兩項工作的指令。如果我們寫好的程式放在 demo\midlets 目錄以外的地方，您可以執行下面的指令，您仍然可以成功地製作 MIDlet：

（實際上 Motorola J2ME SDK 所在路徑太長，為了方便起見，底下我們假設您的 Motorola J2ME SDK 安裝在 D:\MotoSDK 目錄之中，請到時依您個人的需要修改。我們也假設您自行撰寫的 HelloMIDlet.java 置於 d:\jdk1.3. 1\my 目錄之中。同時我們假設您在 d:\jdk1.3. 1\my 目錄底下執行下面指令。）

編譯

```
javac -O -bootclasspath d:\MotoSDK\lib HelloMIDlet.java
```

【注意】

1．-bootclasspath 指向類別函式庫的所在位置。

預先審核

```
D:\MotoSDK\bin\preverifier -classpath d:\MotoSDK\lib;. -
d . HelloMIDlet
```

【注意】

1. -classpath 指向類別函式庫的所在位置，也要指向我們所撰寫的 MIDlet 所在的位置。

2. -d 指向您希望預先編譯類別檔產生之後所放置的路徑，如果寫“. ”表示本目錄，會覆蓋掉原先未經過預先編譯的類別檔。如果您沒有指定，則預設值為".\output"目錄。

當然，如果您嫌自己手動操作很麻煩，您可以將 compileAll.bat 複製到其他目錄之中，並更改其編譯指令與預先審核指令之中和類別函式庫有關的相關設定即可。

提到 compileAll.bat，順道向各位讀者說明一下，在前面有提到，compileAll.bat 可以幫助您編譯所有位於 demo\midlets 目錄下的範例程式，您只要在命列列下直接輸入：

```
compileAll
```

即可。compileAll.bat 會自動當您編譯的 package 有以下幾項：

```
com.mot.j2me.midlets.bounce
com.mot.j2me.midlets.imagetests
com.mot.j2me.midlets.paddleball
com.mot.j2me.midlets.scribble
com.mot.j2me.midlets.tests
com.mot.j2me.midlets.tutorials
```

如果您希望 compileAll.bat 自動幫您編譯其他 package 底下的程式，請您開啟 compileAll.bat，修改其 COMPILECLASS 環境變數的設定即可。

執行 MIDlet

寫好程式之後，大家最希望的事情當然就是讓它在手機上執行，不過由於目前大家無法取得 Motorola 這些支援 Java 的手機，所以我們只能在 Motorola J2ME SDK 內附的模擬器上執行我們寫好的 MIDlet。相信如果您能拿到手機的話，應該可以順利地在手機上執行才是。底下將告訴您如何使用 Motorola J2ME SDK 內附的模擬器來測試您所撰寫的 MIDlet。

在這之前，由於之前我們所撰寫的範例程式只是簡單的 MIDlet 空殼，我們必須讓它能夠在模擬器上秀出一些訊息才可以，因此請修改上一個程式範例，使它的內容如下：

HelloMIDlet.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class HelloMIDlet extends MIDlet
{
    private Display firstDisplay ;
    private Form firstForm ;
    HelloMIDlet()
    {
       firstDisplay = Display.getDisplay(this) ;
       firstForm = new Form("Hello MIDlet") ;
       StringItem firstStrItem = new
StringItem("Hello","MIDlet") ;
       firstForm.append(firstStrItem) ;
    }
    protected void startApp() throws MIDletStateChangeException
    {
        firstDisplay.setCurrent(firstForm) ;
    }
    protected void pauseApp()
```

```
    {
    }
    protected void destroyApp(boolean unconditional)
            throws MIDletStateChangeException
    {
    }
}
```

請將此檔放置在\demo\midlets 目錄下。編譯完成之後，請將目錄切換到 scripts 子目錄之中，您會在此目錄之裡頭發現一些寫好的批次檔。請在該目錄下執行這些批次檔指令以啟動模擬器。執行的指令與執行結果如下所示：

指令：runEmul HelloMIDlet 輸出結果	指令：runMotoi1000 HelloMIDlet 輸出結果
	

指令：rrunMotoiDEN HelloMIDlet 輸出結果	指令：runStarTac HelloMIDlet 輸出結果
	

指令：runEmul HelloMIDlet 輸出結果	指令：runMotoi1000 HelloMIDlet 輸出結果
	

指令：runMyDevice HelloMIDlet 輸出結果
Error loading property file: C:/properties/mydevice.props (系統找不到指定的路徑。) 會出現錯誤訊息是因為您沒有指定屬於使用者自訂的手機外觀的緣故，後面將會教您如何設定使用者自訂的手機外觀。

如果我們將寫好的程式放在 demo\midlets 目錄以外的地方，您可以執行下面的指令，仍然可以成功地啟動模擬器並執行 MIDlet：

（我們假設您的 Motorola J2ME SDK 置於在 D:\MotoSDK 目錄之中，並將您自行撰寫的 HelloMIDlet.java 置於 d:\jdk1.3.1\my 目錄之中。同時我們假設您在 d:\jdk1.3.1\my 目錄底下執行下面指令。）

執行一般模擬器

```
java -Djava.library.path=d:\MotoSDK\lib
-classpath
d:\MotoSDK\bin\Emulator.jar;d:\MotoSDK\bin\ConfigTool.jar
com.mot.tools.j2me.emulator.Emulator
-classpath.;d:\MotoSDK\lib
javax.microedition.midlet.AppManager
HelloMIDlet
-JSA 1 1
```

注　意

第一個-classpath 設定，-classpath 與路徑名稱之間有空格。

第二個-classpath 設定，-classpath 與路徑名稱之間沒有空格。

執行模擬器並套用 i1000 手機外觀

```
java -Djava.library.path=d:\MotoSDK\lib-classpath
d:\MotoSDK\bin\Emulator.jar;d:\MotoSDK\bin\ConfigTool.jar
com.mot.tools.j2me.emulator.Emulator
```

```
-classpath.;d:\MotoSDK\lib
-deviceFile resources\Motorolai1000.props
javax.microedition.midlet.AppManager
HelloMIDlet
-JSA 1 1
```

注 意

第一個-classpath 設定，-classpath 與路徑名稱之間有空格。

第二個-classpath 設定，-classpath 與路徑名稱之間沒有空格。

執行模擬器並套用 iDEN 手機外觀

```
java -Djava.library.path=d:\MotoSDK\lib
-classpath
d:\MotoSDK\bin\Emulator.jar;d:\MotoSDK\bin\ConfigTool.jar
com.mot.tools.j2me.emulator.Emulator
-classpath.;d:\MotoSDK\lib
-deviceFile resources\MotorolaiDENPlatform.props
javax.microedition.midlet.AppManager
HelloMIDlet
-JSA 1 1
```

注 意

第一個-classpath 設定，-classpath 與路徑名稱之間有空格。

第二個-classpath 設定，-classpath 與路徑名稱之間沒有空格。

執行模擬器並套用 StarTac 手機外觀

```
java -Djava.library.path=d:\MotoSDK\lib
-classpath
d:\MotoSDK\bin\Emulator.jar;d:\MotoSDK\bin\ConfigTool.jar
com.mot.tools.j2me.emulator.Emulator
-classpath.;d:\MotoSDK\lib
-deviceFile resources\StarTac.props
javax.microedition.midlet.AppManager
HelloMIDlet
-JSA 1 1
```

注　意

第一個-classpath 設定，-classpath 與路徑名稱之間有空格。

第二個-classpath 設定，-classpath 與路徑名稱之間沒有空格。

執行模擬器並套用使用者自訂手機外觀

```
java -Djava.library.path=d:\MotoSDK\lib
-classpath
d:\MotoSDK\bin\Emulator.jar;d:\MotoSDK\bin\ConfigTool.jar
com.mot.tools.j2me.emulator.Emulator
-classpath.;d:\MotoSDK\lib
-deviceFile <您的 props 檔所在的絕對路徑>
javax.microedition.midlet.AppManager
```

```
HelloMIDlet
-JSA 1 1
```

注 意

第一個-classpath 設定，-classpath 與路徑名稱之間有空格。

第二個-classpath 設定，-classpath 與路徑名稱之間沒有空格。

如果您將您的 props 檔放在 d:\MotoSDK\bin 的 resources 目錄之下，則上述指令只要改成：-deviceFile resources\<您的 props 檔名>

注 意

當您您直接使用上述指令啟動模擬器，如果出現底下錯誤訊息：Error loading property file: resources/defaultdevice.props（系統找不到指定的路徑。）

這是因為您沒有將 d:\MotoSDK\bin 目錄下的 resources 子目錄複製到 d:\JDK1.3.0_01\my 目錄之下的緣故。

對 MIDlet 除錯 ▼

撰寫 PalmOS 上的 Spotlet 時，我們可以利用 System.out.println()函式幫我們印出一些訊息以幫助除錯，那麼在手機上的 MIDlet 呢? 原則上，我們還是可以利用 System.out.println()函式做一些輸出。當模擬器執行時，就會在命令列上輸出一些訊息。

在 Motorola J2ME SDK 內附的說明文件之中，概略地提到了除錯的問題，裡頭提到，往後如果我們要進行上機除錯（on-device debugging）的話，必須要滿足幾個條件:

1. 機器本身要具備除錯相關功能，並與 KDWP（Kvm Debug Wire Protocol）相容。因為除錯時，除錯工具需要利用 KDWP 和機器上交談以取得除錯資訊。
2. 製造廠商本身要提供下載 MIDlet 到手機上以進行除錯的方法。
3. 提供對 MIDlet 除錯的工具，必須支援手機在利用 KDWP 除錯時所此用的傳輸媒介（例如序列阜或 UDP)。

不過在 Motorola J2ME SDK 所在目錄下，提供了 runDebug Device.bat 與 runDebugAgent.bat。這兩個批次檔會在除錯時對我們有很大的幫助。

Motorola J2ME SDK 對中文的支援 ▼

中文的問題分成兩個部分，一個是在使用者介面上的中文問題，一個是在命令列輸出（利用 System.out.println()函式所做的輸出）上的中文問題，請大家做個小實驗，將前面我們所撰寫的程式改如下：

HelloMIDlet.java

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class HelloMIDlet extends MIDlet
{
    private Display firstDisplay ;
    private Form firstForm ;
    HelloMIDlet()
    {
       firstDisplay = Display.getDisplay(this) ;
       firstForm = new Form("哈囉!MIDlet") ;
       StringItem firstStrItem = new StringItem("哈囉","米德列特
") ;
       firstForm.append(firstStrItem) ;
       System.out.println("MIDlet啓動") ;
    }
    protected void startApp() throws MIDletStateChangeException
    {
        firstDisplay.setCurrent(firstForm) ;
    }
    protected void pauseApp()
    {
    }
    protected void destroyApp(boolean unconditional)
             throws MIDletStateChangeException
    {
    }
}
```

將本 MIDlet 編譯並經過預身審核之後，我們開啟模擬器來執行此 MIDlet，底下為執行結果：

使用者介面輸出	命令列輸出
	D:\MotoSDK\scripts>runEmul HelloMIDlet MIDlet??

我們從結果發現，預設的編譯指令會讓使用者介面正常輸出中文，而命令列無法輸出正確的中文。

接著請將 compileAll.bat 之中原本的指令

```
javac -O -bootclasspath ..\..\lib %COMPILECLASS%
```

修改為

```
javac -encoding ISO8859_1 -O -bootclasspath ..\..\lib
%COMPILECLASS%
```

之後，重新編譯此 MIDlet 執行結果：

使用者介面輸出	命令列輸出
	D:\MotoSDK\scripts>runEmul HelloMIDlet MIDlet啓動

我們從結果發現，預設的編譯指令會讓使用者介面無法正常輸出中文，而命令列卻可以輸出正確的中文。

從這裡我們可以看出，Motorola J2ME SDK 與 MIDP 參考實做所遇到的中文問題完全相同。