基本自訂指南

Sun Java™ Wireless Toolkit for CLDC

版本 2.5.2

Sun Microsystems, Inc.
www.sun.com

2007 年 9 月

目錄

前言

「Sun Java™ Wireless Toolkit for CLDC 基本自訂指南」提供用於自訂 Sun Java Wireless Toolkit for CLDC 的技術性詳細資訊。

本書適用對象

本指南適用於爲個人使用目的 (例如爲了能夠適應新設備，或爲了修改原始碼以適應模擬器的新 API) 而需自訂 Sun Java Wireless Toolkit for CLDC 的開發者。

本書架構

本指南包括以下各章及附錄：

第 1 章概略介紹各種自訂的可能方式。

第 2 章說明如何建立新的模擬器面板。

第 3 章說明如何建立混淆器外掛程式。

路徑與作業系統指令

本文件可能未包含有關基本 UNIX®、Linux 或 Microsoft Windows 系統指令和程序的資訊，如開啓終端機視窗、變更目錄以及設定環境變數。如需此類資訊，請參閱系統隨附的軟體文件。

本文件中的範例適用於 Windows 平台。在 Linux 系統中，請用正斜線 (/) 取代反斜線路徑分隔符號 (\)。例如：

Windows	*toolkit*\wtklib\devices\DefaultColorPhone\netIndicatorOn.png
Linux	*toolkit*/wtklib/devices/DefaultColorPhone/netIndicatorOn.png

在本書中，*toolkit* 固定指安裝 Sun Java Wireless Toolkit for CLDC 的目錄。*workdir* 指使用者的工作目錄。*workdir* 的預設位置通常是以下位置之一：

Windows:	C:\Documents and Settings*User*\j2mewtk\2.5.1 (其中 *User* 是您的帳號名稱)
Linux:	~/j2mewtk/2.5.1 (其中 ~ 是您的主目錄)

印刷排版慣例

表格 P-1　印刷排版慣例

字體	意義	範例
AaBbCc123 (固定間距)	API 和語言元素、HTML 標記、網站 URL、指令名稱、檔案名稱、目錄路徑名稱、螢幕畫面輸出、範例代碼。	編輯您的 `.login` 檔案。 使用 `ls -a` 列出所有檔案。 `% You have mail。`
AaBbCc123 (固定間距粗體)	您鍵入的內容，與螢幕畫面輸出相對。	**`% su`** `Password:`
AaBbCc123 (斜體)	指令或路徑名稱中要由真實名稱或值替代的預留位置。	該檔案位於 *install-dir*/bin 目錄中。 這些稱爲**類別**選項。
AaBbCc123	新的字彙或術語、要強調的詞。	請**勿**儲存此檔案。
「AaBbCc123」	用於書名及章節名稱。	請閱讀「使用者指南」中的第 6 章。

相關文件

本小節列出了相關的 Java Platform, Micro Edition (Java ME) 規格。您在某些規格名稱中可看到 Java ME 先前稱爲 Java 2 Platform, Micro Edition (或 J2ME™)。

表格 P-2 相關文件

主題	標題
Sun Java Wireless Toolkit for CLDC	Sun Java Wireless Toolkit for CLDC 使用者指南
自訂	Sun Java Wireless Toolkit for CLDC 進階自訂指南*
統一模擬器介面	統一模擬器介面規格
PDAP Optional Packages - JSR 75	PDA Optional Packages for the J2ME Platform
藍芽與 OBEX - JSR 82	Java APIs for Bluetooth
MIDP 2.1 - JSR 118	Mobile Information Device Profile 2.0
CLDC 1.1 - JSR 139	J2ME Connected Limited Device Configuration
MMAPI - JSR 135	Mobile Media API
J2ME Web Services - JSR 172	J2ME Web Services Specification
SATSA - JSR 177	Security and Trust Services APIs for J2ME
Location API - JSR 179	Location API for J2ME
SIP API - JSR 180	SIP API for J2ME
Mobile 3D Graphics - JSR 184	Mobile 3D Graphics API for J2ME
JTWI - JSR 185	Java Technology for the Wireless Industry
WMA 2.0 - JSR 205	Wireless Messaging API (WMA)
CHAPI 1.0 - JSR 211	Content Handler API
SVG API - JSR 226	Scalable 2D Vector Graphics API for J2ME
Payment API - JSR 229	Payment API
Advanced Multimedia - JSR 234	Advanced Multimedia Supplements
Mobile Internationalization - JSR 238	Mobile Internationalization API
Java Binding for OpenGL® ES API - JSR 239	Java Binding for OpenGL® ES API
MSA - JSR 248	Mobile Service Architecture

* 有原始碼授權即可獲得。

雖然規格是重要的資訊，但有時不易取得。請參閱 [Mobility] 頁面上的其他觀點：

`http://developers.sun.com/techtopics/mobility/`

存取線上文件

下列網站提供有關 Java 技術的技術性文件：

- `http://developers.sun.com/`
- `http://java.sun.com/docs/`

我們歡迎您提出寶貴意見

我們致力於提高文件品質，因此誠心歡迎您提出建議。您可以透過 `developers.sun.com` 電子郵件提供意見回饋。

章 1

簡介

Sun Java Wireless Toolkit for CLDC 提供模擬環境，供開發者開發 MIDP 應用程式。本文件說明如何以兩種有用的方式自訂工具組：

- 建立新的模擬器面板
- 建立混淆器外掛程式

這一章其餘部分會對每一種自訂方式進行簡要描述。

1.1 建立新的模擬器面板

您可以使用以下方法自訂 Sun Java Wireless Toolkit for CLDC 中的模擬器：

1. 下載協力廠商模擬器，並安裝入 Sun Java Wireless Toolkit for CLDC。
2. 以 Sun Java Wireless Toolkit for CLDC 的預設模擬器爲基礎，建立新的模擬器面板。第 2 章會說明此程序。
3. 自訂預設的模擬器實作。若要執行此動作，需具有 Sun Java Wireless Toolkit for CLDC 原始碼的授權，以自訂模擬器實作。

1.2 建立混淆器外掛程式

混淆器 *(Obfuscator)* 是用來縮小 MIDlet 套件可執行檔大小的工具。MIDlet 套件越小，表示下載時間越短，這在目前頻寬不足的無線世界中，意味著等待時間比較少，對使用者而言，通話費也可能比較少。

Sun Java Wireless Toolkit for CLDC 支援 ProGuard 混淆器 (`http://proguard.sourceforge.net/`)，但它具有靈活的架構，可讓您使用任何類型的混淆器。

第 3 章提供技術性詳細資訊。

章 2

設定模擬器面板

這一章說明如何定義模擬器面板。您可以修改現有面板，或爲模擬器建立新的面板。此程序稱爲**設定模擬器面板**。

2.1 面板特性檔

模擬器面板由一個特性檔定義。每個面板特性檔均位於 *toolkit*\wtklib\devices 下其自己的子目錄中，其中 *toolkit* 是 Sun Java Wireless Toolkit for CLDC 的安裝目錄。特性檔的名稱將與目錄名稱相符。

例如，DefaultColorPhone 面板由 *toolkit*\wtklib\devices\DefaultColorPhone 目錄中的 DefaultColorPhone.properties 定義。

面板特性檔會定義模擬器面板的外觀與運作方式。該檔案包含影像與聲音 (不一定儲存於相同目錄內) 指標。例如，DefaultColorPhone 目錄包含電話本身的影像，但 DefaultColorPhone 的圖示與聲音則在 wtklib\devices\Share 中定義。

這一章的其餘部分會描述面板特性檔的內容。特性檔是純文字檔。您可以用任何文字編輯器進行修改。一般而言，特性檔中項目的格式是一個特性名稱後面跟著一個值。名稱與值之間由冒號或等號分隔。以井字標記 (#) 開頭的行則是註解。

建立新面板時，最簡單的方式是複製現有的面板進行修改，例如：

1. **複製 DefaultColorPhone 目錄。**
2. **以新面板的名稱做為新目錄的名稱。**
3. **重新命名特性檔，以符合目錄名稱。**
   如果將目錄命名爲 NewSkin，請將其特性檔重新命名爲 NewSkin.properties。

2.2 面板外觀

模擬器面板的整體外觀由許多因素決定，這一節將分別說明每一項因素：

- 面板影像
- 螢幕界限與可繪製區域
- 螢幕特性
- 圖示
- 字型
- 指令
- 聲音

2.2.1 面板影像

面板大部分的外觀由三個影像決定：

1. **預設**影像會顯示裝置處於非作用中 (正常) 狀態。
2. **高亮度顯示**影像會顯示裝置的所有按鈕都以高亮度顯示時的狀態，如同使用者將滑鼠移至按鈕上方。
3. **按下**的影像則會顯示裝置的所有按鈕都已經按下時的狀態。

這些影像都顯示整個裝置。工具組使用這些影像的不同部分，來呈現高亮度顯示及按下的按鈕。

例如，圖 2-1 顯示 `DefaultColorPhone` 中的三個影像。

圖 2-1 DefaultColorPhone 的正常影像、高亮度顯示影像和按下按鈕時的影像

圖 2-2 中顯示每個小鍵盤的部分放大圖，可讓您看出三個影像的差異。

圖 2-2 模擬器面板的正常影像、高亮度顯示影像以及按下按鈕時的影像的詳細資訊

面板特性檔中以下列特性指定三個影像檔：

- default_image= 影像檔案名稱
- pressed_buttons_image= 影像檔案名稱
- highlighted_image= 影像檔案名稱

影像檔可以是 PNG、GIF 或 JPEG 格式，所有影像檔都要使用同樣尺寸。例如，`DefaultColorPhone.properties` 有下列幾個項目：

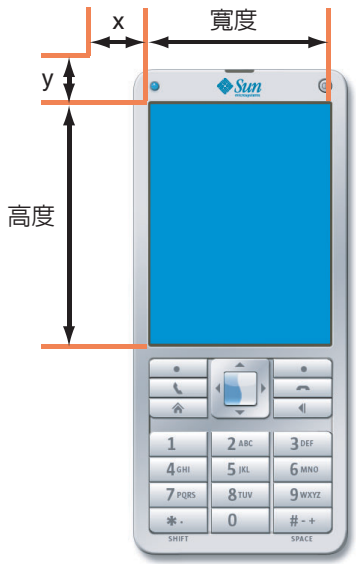
- `default_image=DefaultColorPhoneNormal.png`
- `pressed_buttons_image=DefaultColorPhonePressed.png`
- `highlighted_image=DefaultColorPhoneHiLite.png`

2.2.2 螢幕界限與可繪製區域

該螢幕代表實際裝置的顯示幕。它是由整體的螢幕界限、**可繪製**界限以及其他參數 (用以決定色彩數等因素) 來定義。

整體的螢幕界限是顯示幕的總面積。定義方式爲從左上角影像檔原點 (0,0) 算起的長寬像素來計算。

圖 2-3 螢幕界限



特性檔中以下列方式定義螢幕界限：

- `screen.x=`*x* 座標
- `screen.y=`*y* 座標
- `screen.width=` 寬度
- `screen.height=` 高度

例如：

- `screen.x=37`
- `screen.y=54`
- `screen.width=240`
- `screen.height=320`

大部分裝置不會讓 MIDP 應用程式使用全部的顯示區。螢幕的其餘部分通常保留供不同類型的圖示與指示器使用。同樣的，Sun Java Wireless Toolkit for CLDC 模擬器可讓您定義完整螢幕中，可供 MIDP 應用程式使用的子集，稱爲**可繪製**區域。可繪製區域的原點以相對於顯示幕左上角的座標表示。例如，`DefaultColorPhone` 模擬器面板使用最上面一列顯示圖示，最底下一列顯示軟式標籤和其他圖示，如圖 2-4 所示。

圖 2-4 `DefaultColorPhone` 中的可繪製螢幕區域

在模擬器面板特性檔中，可繪製區域的表示方法如下。

- `screenPaintableRegion.x=`*x* 座標
- `screenPaintableRegion.y=`*y* 座標
- `screenPaintableRegion.width=` 寬度
- `screenPaintableRegion.height=` 高度

請注意，x 和 y 座標依螢幕位置而定。並且 `screenPaintableRegion` 的寬度和高度值不得超出 `screen.width` 和 `screen.height` 的相應值。

例如：

- `screenPaintableRegion.x=0`
- `screenPaintableRegion.y=10`
- `screenPaintableRegion.width=240`
- `screenPaintableRegion.height=290`

備註 – 在全螢幕模式下 (MIDP 2.0 或更高版本中)，模擬器使用的區域從可繪製區域原點開始，一直延伸到螢幕右下角。在 `DefaultColorPhone` 中，這是除了最上面一列以外的完整螢幕區域。

2.2.3 螢幕特性

模擬器面板特性檔決定螢幕支援的色彩數和像素縱橫比。`iscolor` 指定模擬器面板是使用彩色還是灰階。

2.2.3.1 `isColor=`*[true|false]*

指定 *true* 代表彩色，*false* 代表灰階。

2.2.3.2 `colorCount=` 數值

`colorCount` 特性會指定可用的色彩數。對於灰階裝置則指定不同深淺的灰色數：

例如，`DefaultColorPhone` 具有 4096 色的彩色螢幕：

```
isColor=true
colorCount=0x1000
```

2.2.3.3 `enableAlphaChannel=`*[true|false]*

此選項決定模擬器的 alpha (透明度) 處理。

2.2.3.4 `gamma=` 數值

此值決定 Gamma 系數的校正。1 的值表示不做錯誤校正。

2.2.3.5 screenDoubleBuffer=[*true*|*false*]

`true` 可啓用雙重緩衝，`false` 會將其停用。

2.2.3.6 screenBorderColor= 色彩

`screenBorderColor` 指定螢幕可繪製區域以外的背景色彩。例如，`DefaultColorPhone` 使用下列色彩：

```
screenBorderColor=0xb6b6aa
```

2.2.3.7 screenBGColor= 色彩

使用此特性可設定灰階裝置上螢幕的背景色彩。

2.2.4 圖示

Sun Java Wireless Toolkit for CLDC 模擬器支援使用圖示，也就是可向使用者傳達資訊的小影像。通常圖示位在顯示螢幕上之可繪製區域以外的位置。模擬器實作一組固定的圖示，您可以在表格 2-1 中看到說明。

表格 2-1 模擬器圖示

名稱	描述
battery	顯示電池狀態
domain	指示執行中 MIDlet 的保護網域
down	表示可以捲動
inmode	指示輸入模式：小寫、大寫、數字
internet	顯示網際網路活動
left	表示可以捲動
reception	顯示無線訊號強度
right	表示可以捲動
up	表示可以捲動

圖示的定義方式是利用位置 (從螢幕原點算起)、預設狀態以及對應到可能狀態的影像清單。例如，此處爲 `DefaultColorPhone` 中的 `down` 圖示的定義，也就是所顯示的清單或表單高於可用螢幕空間時，會出現的向下箭頭。

```
icon.down: 113, 314, off
icon.down.off:
icon.down.on: ../Share/down.gif
```

第一行指定顯示圖示的位置，這對 DefaultColorPhone 而言，是指可繪製螢幕區域外，最底下一列中央的位置。預設狀態是 off。

沒有影像檔對應到 off 狀態，但 on 狀態使用 wtklib\devices\Share 目錄中的 down.gif 影像。

另一個有趣的範例是 inmode 圖示，它包含 7 個狀態，有 6 個對應的影像檔：

```
icon.inmode: 113, 2, off
icon.inmode.off:
icon.inmode.ABC: ../Share/ABC.gif
icon.inmode.abc: ../Share/abc_lower.gif
icon.inmode.123: ../Share/123.gif
icon. inmode.kana: ../Share/kana.gif
icon.inmode.hira: ../Share/hira.gif
icon.inmode.sym: ../Share/sym.gif
```

模擬器另一個類似圖示的部分在於網路指示器。網路指示器不是位在螢幕上，而是顯示在模擬器面板上。在 DefaultColorPhone 中，網路指示器顯示成模擬器面板左上角的小綠燈。網路指示器的定義使用兩個特性：

- netindicator.image: **影像**
- netindicator.bounds: *x*，*y*，**寬度**，**高度**

例如，在 DefaultColorPhone 中，網路指示器類似：

- netindicator.image: net_indicator.png
- netindicator.bounds: 53, 27, 30, 30

寬度與高度應該符合網路指示器影像的寬度與高度。

2.2.5 字型

模擬器所用的字型定義在面板特性檔中。基本上，您可以爲 MIDP Font 類別中可用的每一種字體、樣式與大小各定義一種字型。格式如下：

font.*face*.*style*.*size*: **字型指定元**

您可以從 MIDP Font API 推測字體、樣式與大小參數，但是在模擬器面板特性檔中，識別碼是小寫。字體是 system、monospace 或 proportional，樣式是 plain、bold 或 italic，大小是 small、medium 或 large。

字型指定元遵循 Java Platform, Standard Edition (Java SE) java.awt.Font 類別程式庫中規定的慣例。下列來自 DefaultColorPhone 的範例中定義了三種大小的比例斜體字型：

```
font.proportional.italic.small: SansSerif-italic-9
font.proportional.italic.medium: SansSerif-italic-11
font.proportional.italic.large: SansSerif-italic-14
```

您必須指定預設字型，以用在沒有任何其他定義的情況下。在 `DefaultColorPhone` 中，預設字型是 10 點的 SansSerif 字型：

```
font.default=SansSerif-plain-10
```

字型也可以加底線。依預設，MIDP 實作會支援此功能，但您可以用下列方式停用特定字型的加底線功能：

`font.`*face*`.`*style*`.`*size*`.underline.enabled=false`

您可以視需要，以下列方式停用所有字型的加底線功能：

```
font.all.underline.enabled=false
```

如果不使用系統字型，您還有一個其他選項，就是使用**點陣**字型。點陣字型只是當成字型的影像，影像中包含字元的形狀。點陣字型影像是單行文字，內容是其中每個字元的形狀。若要定義點陣字型，請使用下列特性：

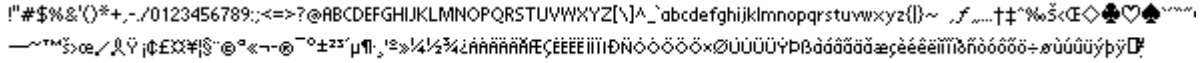
`font.`*name*`=` **字型特性檔**

字型特性檔包含下列特性定義：

- `font_image =` **影像檔案**
- `font_height =` **字型高度**
- `font_ascent =` **字型上升高度**
- `font_descent =` **字型下降高度**
- `font_leading =` **字型行距**

影像檔可以是 PNG、GIF 或 JPEG 格式。它必須包含一列字元。圖 2-5 顯示了兩列，以便容易看到這些字元。

圖 2-5 點陣字型影像



指定高度、上升高度、下降高度與行距時，都以像素爲單位。如果您不熟悉這些字型術語，請參閱 Java SE 文件中的 `java.awt.FontMetrics`。

字型特性檔還必須包含 ASCII 字元碼與影像中水平像素偏移量之間的對映清單。在下列範例中，ASCII 碼 65 對映到水平偏移量 124：

```
ascii_x-65=124
```

定義點陣字型後，它的名稱就可以做爲字型指定元使用。

2.2.6 軟式按鈕標籤

軟式按鈕是沒有固定功能的按鈕。這一章以下部分會詳細介紹這類按鈕。軟式按鈕的標籤顯示在螢幕上。模擬器面板特性檔會決定軟式按鈕標籤的顯示位置與方式。

軟式按鈕標籤的字型是使用字型別名 (您爲字型指定的簡稱) 來定義。每個軟式按鈕標籤均由一個特性來描述：

`softbutton.`*n*`=`*x*`,` *y*`,` *width*`,` *height*`,` *font-alias*`,` *alignment*

alignment 的有效值是 `left`、`right` 與 `center`。

例如，下列特性會告訴工具組，軟式按鈕標籤要使用 Courier 12 號字型：

- `font.softButton=Courier-plain-12`
- `softbutton.0=1,306,78,16,` 軟式按鈕 `,` 左
- `softbutton.1=160,306,78,16,` 軟式按鈕 `,` 右

首先定義字型別名 `softButton`。第一個標籤靠左對齊，第二個標籤則靠右對齊。

2.2.7 聲音

MIDP 警示會有相關聯的聲音。在 Sun Java Wireless Toolkit for CLDC 模擬器中，聲音使用檔案來定義；MIDP `AlertType` 類別中列舉的每種類型各有一個檔案。模擬器可以使用基礎 Java SE 實作支援的任何聲音檔類型。在 Java SE Development Kit 1.5 中，這包括 AIFF、AU、WAV、MIDI 與 RMF。例如，這是 `DefaultColorPhone` 中的定義：

```
alert.alarm.sound:    ../Share/mid_alarm.wav
alert.info.sound:     ../Share/mid_info.wav
alert.warning.sound: ../Share/mid_warn.wav
alert.error.sound:    ../Share/mid_err.wav
alert.confirmation.sound: ../Share/mid_confirm.wav
```

如果特定警示類型未定義任何聲音，就會播放預設聲音：

`alert.confirmation.sound:` **聲音檔案**

此外，您也可以定義模擬電話振動的聲音。在 `DefaultColorPhone` 中，此定義如下：

```
vibrator.sound: ../Share/vibrate.wav
```

2.3 對映使用者輸入

模擬器面板的說明分爲兩部分。第一個部分是外觀，已在上文中說明。第二個部分則會定義如何將使用者輸入對映至模擬器中。

2.3.1 鍵盤處理程式

鍵盤處理程式會辨認使用者按下的按鈕，並在模擬器中執行適當的動作。例如，如果使用滑鼠按一個軟式按鈕，鍵盤處理程式就會負責在模擬器中執行適當的動作。

鍵盤處理程式會定義一組標準按鈕名稱，供您用於定義按鈕。您只須告訴模擬器，按鈕在面板上的位置，鍵盤處理程式就會處理其他工作。

Sun Java Wireless Toolkit for CLDC 模擬器包含兩個鍵盤處理程式，一個用於使用 ITU-T 小鍵盤的電話裝置 (`DefaultKeyboardHandler`)，另一個用於使用全標準鍵盤的裝置。例如，`DefaultColorPhone` 包含以下鍵盤處理程式特性：

```
keyboard.handler = com.sun.kvem.midp.DefaultKeyboardHandler
```

`DefaultKeyboardHandler` 能夠辨識下列標準按鈕名稱：`0` 至 `9`、`POUND`、`ASTERISK`、`POWER`、`SEND`、`END`、`LEFT`、`RIGHT`、`UP`、`DOWN`、`SELECT`、`SOFT1`、`SOFT2`、`SOFT3`、`SOFT4`、`USER1` 至 `USER10`。

在 `QwertyDevice` 中，鍵盤處理程式格式如下：

```
keyboard.handler = com.sun.kvem.midp.QwertyKeyboardHandler
```

`QwertyKeyboardHandler` 支援的按鈕與 `DefaultKeyboardHandler` 相同，而且也包含標準鍵盤上的按鈕，如字母鍵、Shift 與 Alt。

2.3.2 按鈕

按鈕的定義是使用名稱與一組座標。如果提供兩組座標，所定義的就是矩形按鈕。如果出現的座標多於兩組，則按鈕會使用多邊形區域。

按鈕區域以裝置面板影像上的相對位置來定義。當使用者將滑鼠移至定義的按鈕區域上方時，就會顯示面板高亮度顯示影像上對應的區域。如果使用者按下按鈕，就會顯示面板按下按鈕時的影像上對應的區域。

按鈕本身並不是很有趣，只是將按鈕名稱與矩形或多邊形區域連在一起。鍵盤處理程式才會將按鈕名稱對映至模擬器中的某項功能。第 2-12 頁的第 2.3.3 節「指定桌面鍵盤按鍵至按鈕」解釋了桌上型電腦鍵盤上的按鍵如何對映至按鈕。

下列特性示範如何定義按鈕 5 的矩形區域。它的原點是 140, 553，寬度是 84，高度是 37。

```
button.5 = 140, 553, 84, 37
```

以下是星號按鈕的多邊形定義範例：

```
button.ASTERISK = 66, 605, 110, 606, 140, 636, 120, 647, 70, 637
```

此多邊形的定義，是使用多條直線連接列出來的點：

```
66, 605
110, 606
140, 636
120, 647
70, 637
```

2.3.3 指定桌面鍵盤按鍵至按鈕

按鈕可以擁有一個或多個相關聯的桌面鍵盤按鍵。這表示您可以使用桌面鍵盤來控制模擬器，而不必將滑鼠移至裝置面板上，再按滑鼠按鈕。

例如，DefaultColorPhone 可讓您在桌面鍵盤上按下 F1 鍵，模擬左邊軟式按鈕。左邊軟式按鈕在特性檔中定義爲 SOFT1，如下所示：

```
button.SOFT1 = 78, 417, 120, 423, 126, 465, 74, 440
```

桌面鍵盤捷徑定義如下：

```
key.SOFT1 = VK_F1
```

實際的按鍵定義是 *virtual key codes*，要在 Java SE `java.awt.event.KeyEvent` 類別程式庫中定義。如需詳細資訊，請參閱 Java SE 文件。

您可以視需要爲按鈕指定多個桌面鍵盤按鍵。在以下取自 DefaultColorPhone 的範例中，桌面鍵盤上的按鍵 5 或數字鍵台按鍵 5 都定義爲模擬器面板上按鈕 5 的捷徑：

```
key.5 = VK_5 VK_NUMPAD5
```

2.3.4 對映遊戲按鍵

遊戲動作已經在 DefaultKeyboardHandler 中定義，但您可以用 QwertyKeyboardHandler 指定您自己的遊戲動作。請使用此格式：

`game.`*function* `=` **按鈕名稱**

function 可以是 LEFT、RIGHT、UP、DOWN 與 SELECT 其中之一。您可以在這一章前面的部分找到標準按鈕名稱的說明。

預設設定如下所示：

- `game.UP = UP`
- `game.DOWN = DOWN`
- `game.LEFT = LEFT`
- `game.RIGHT = RIGHT`
- `game.SELECT = SELECT`

2.3.5 對映按鍵至字元

您可以使用 `QwertyKeyboardHandler` 定義按下按鈕會產生哪個字元，而且按鈕可以單獨按下，也可以與 Shift 或 Alt 鍵一起按下。

請使用此格式：

`keyboard.handler.qwerty.`*`button`* `= '`*`base-character`*`' '`*`shift-character`*`' '`*`alternate-character`*`'`

base character 是此按鈕通常產生的字元，shift character 是同時按下 Shift 和此按鈕時所產生的字元，而 alternate character 則是同時按下 Alt 和此按鈕時所產生的字元。

您可以使用以下兩種方法模擬 Shift 或 Alt 與某按鈕同時按下的動作：

- 將按鈕對映至鍵盤 (如前一節所述)，並在按下 Shift 或 Alt 鍵時，同時按下與該按鈕相關聯的按鍵。
- 按下 Shift-Lock 或 Alt-Lock，然後按下該按鈕。再按一次 Shift-Lock 或 Alt-Lock，回到初始狀態。

例如：

```
keyboard.handler.qwerty.A = 'a' 'A' '?'
```

2.3.6 對映指令至軟式按鈕

指令是 MIDP 規格的一部分。這種靈活的方式可指定使用者需要的動作，卻不需要要求特定裝置應該如何提供該功能。

一般而言，MIDP 裝置使用軟式按鈕來呼叫指令。指令文字會顯示在顯示幕上靠近軟式按鈕的位置。如果可用的指令多於可用的軟式按鈕，實作會將一個軟式按鈕標籤顯示爲功能表。只要按下功能表軟式按鈕，就會顯示可用指令功能表。

Sun Java Wireless Toolkit for CLDC 模擬器可讓您指定要根據 `javax.microedition.lcdui.Command` 中指定的指令類型，來決定某些指令類型的顯示位置。例如，在有兩個軟式按鈕的模擬器面板上，您可能偏好讓 `BACK` 與 `EXIT` 指令始終顯示在左邊軟式按鈕上，並讓 `OK` 指令顯示在右邊軟式按鈕上。

您可以使用下列格式，在模擬器面板特性檔中指定這些偏好類型：

```
command.keys.command-type= 按鈕
```

例如，DefaultColorPhone 以下列方式定義指令喜好設定：

```
command.keys.BACK = SOFT1
command.keys.EXIT = SOFT1
command.keys.CANCEL = SOFT1
command.keys.STOP = SOFT1

command.keys.OK = SOFT2
command.keys.SCREEN = SOFT2
command.keys.ITEM = SOFT2
command.keys.HELP = SOFT2
```

藉由指定其他按鈕名稱，您可以爲特定指令類型指定其他偏好的按鈕。例如，這一行會告訴模擬器，如果可以，將 BACK 指令對映至 END，否則對映至 SOFT1。

```
command.keys.BACK = END SOFT1
```

最後，您可以視需要，指定只用於特定指令類型的軟式按鈕。下列定義會限制僅可以將 BACK、EXIT、CANCEL 和 STOP 指令類型對映至 SOFT1 鍵。

```
command.exclusive.SOFT1 = BACK EXIT CANCEL STOP
```

2.3.7 指令功能表

如果可用的軟式按鈕少於指令，指令就會放置在功能表中。Sun Java Wireless Toolkit for CLDC 模擬器提供指令功能表的控制方式。您可以選擇用來顯示功能表的按鈕、用來在功能表項目中移動的按鈕，以及顯示功能表時所用的文字標籤。

下列取自 DefaultColorPhone 的特性，會告訴模擬器面板使用第二個軟式按鈕來顯示或隱藏功能表。

```
command.menu.activate = SOFT2
```

預設狀況下使用 UP 與 DOWN 按鈕來移動功能表項目，並使用 SELECT 來選擇指令。您可以用下列特性變更這些指定方式：

- command.menu.select = 按鈕
- command.menu.up = 按鈕
- command.menu.down = 按鈕

2.3.8 暫停與重新繼續

MIDP 規格可讓應用程式 (MIDlet) 隨時暫停，例如接聽來電這類的其他電話事件。

您可以用模擬器面板特性檔來定義暫停與重新繼續 `MIDlet` 的桌面鍵盤捷徑。例如，`DefaultColorPhone` 使用 F6 代表暫停，使用 F7 代表重新繼續：

```
midlet.SUSPEND_ALL = VK_F6
midlet.RESUME_ALL = VK_F7
```

2.3.9 指標事件

模擬器面板是否有觸控式螢幕由一個特性決定，如下所示：

```
touch_screen=[true|false]
```

如果模擬器面板具有觸控式螢幕，則指標事件會傳送至 `Canvases`。

2.4 語言環境與字元編碼

語言環境是指具有相同語言、習俗或文化傳統的地理或政治區域或團體。在軟體中，語言環境是指包含必要資訊，使軟體適用於特定地區的檔案、資料與程式碼的集合。

有些作業僅能用於特定語言環境，需要指定的語言環境才能提供適合使用者的資訊，如下所示：

- 對使用者顯示的訊息
- 文化資訊，例如日期與貨幣格式

在 Sun Java Wireless Toolkit for CLDC 模擬器中，預設語言環境由平台的語言環境來決定。

若要定義特定語言環境，請使用以下定義：

`microedition.locale:` **語言環境名稱**

語言環境名稱由兩部分組成，以破折號 (-) 分隔，例如 `en-US` 是專爲美國英語指定的語言環境，而 `en-AU` 則是專爲澳洲英語指定的語言環境。

第一部分是有效的 ISO 語言碼。這些代碼依 ISO-639 規定，以小寫的二個字母表示。您可以在許多網站找到完整的代碼清單，例如 `http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt`。

第二部分是有效的 ISO 國碼。這些代碼依 ISO-3166 規定，以大寫的二個字母表示。您可以在許多網站找到完整的代碼清單，例如：
`http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html`。

CLDC 中的輸入與輸出 API 使用已命名的字元編碼，在 8 位元字元與 16 位元 Unicode 字元之間進行轉換。特定 MIDP 實作只提供一小組編碼供 MIDlet 使用。

在模擬器中，預設編碼是您正在執行的平台的預設編碼器。您的模擬器可以使用 Java SE 平台可用的其他編碼，例如 UTF-8 和 UTF-16。

若要定義模擬器面板使用的字元編碼，請使用下列定義：

`microedition.encoding:` **編碼**

若要定義所有可用的編碼集，請使用下列定義：

`microedition.encoding.supported:` **編碼清單**

例如：

```
microedition.encoding: UTF-8
microedition.encoding.supported: UTF-8, UTF-16, ISO-8859-1,
ISO-8859-2, Shift_JIS
```

若要支援 Java SE 平台支援的所有編碼，請將
`microedition.encoding.supported` 定義留白，例如：

```
microedition.encoding.supported:
```

備註 – 不論 ISO-8859-1 編碼是否列在 `microedition.encoding.supported` 項目中，一定可用於模擬裝置上執行的應用程式。

章 3

建立混淆器外掛程式

Sun Java Wireless Toolkit for CLDC 可讓您使用位元碼混淆器，來縮小 MIDlet 套件 JAR 檔案的大小。本工具組支援 ProGuard，如「Sun Java Wireless Toolkit for CLDC 使用者指南」所述。

如果要使用其他混淆器，您可以爲 Sun Java Wireless Toolkit for CLDC 撰寫外掛程式。

3.1 撰寫外掛程式

混淆器外掛程式會延伸 `com.sun.kvem.environment.Obfuscator` 介面。此介面本身包含於 *toolkit*`\wtklib\kenv.zip` 中。

`Obfuscator` 介面包含您必須實作的兩個方法：

- `public void createScriptFile(File jadFilename File projectDir);`
- `public void run(File jarFileObfuscated, String wtkBinDir, String wtkLibDir, String jarFilename, String projectDir, String classPath, String emptyAPI) throws IOException;`

若要編譯您的混淆器外掛程式，請確定將 `kenv.zip` 加入您的 `CLASSPATH`。

例如，以下是一個非常簡單的外掛程式原始碼。它不會真的呼叫混淆器，但可以示範如何實作 `Obfuscator` 介面。

```
import java.io.*;
public class NullObfuscator
    implements com.sun.kvem.environment.Obfuscator {
  public void createScriptFile(File jadFilename, File projectDir) {
    System.out.println("NullObfuscator: createScriptFile()");
  }

  public void run(File jarFileObfuscated, String wtkBinDir,
    String wtkLibDir, String jarFilename, String projectDir,
    String classPath, String emptyAPI) throws IOException {
      System.out.println("NullObfuscator: run()");
    }
}
```

假設將此程式碼儲存成 *toolkit*\wtklib\test\NullObfuscator.java，即可在指令行使用類似下列指令進行編譯：

```
set classpath=%classpath%;toolkit\wtklib\kenv.zip
javac NullObfuscator.java
```

3.2 配置工具組

混淆器外掛程式撰寫完成之後，您必須將其位置告訴工具組。若要執行此動作，請編輯 *toolkit*\wtklib\Windows\ktools.properties。編輯混淆器外掛程式的類別名稱，並將類別的位置告訴工具組。如果您要照著範例一起做，請依照下列方式編輯特性：

```
obfuscator.runner.class.name: NullObfuscator
obfuscator.runner.classpath: wtklib\\test
```

重新啓動工具組，並開啓專案。接著選擇 [專案] > [封裝] > [建立模糊化封裝]。主控台會顯示 NullObfuscator 輸出，如下所示：

```
Project settings saved
Building "Tiny"
NullObfuscator: createScriptFile()
NullObfuscator: run()
Wrote C:\WTK252\apps\Tiny\bin\Tiny.jar
Wrote C:\WTK252\apps\Tiny\bin\Tiny.jad
Build complete
```

索引